Summer 2023 CSE 480
Project 3: DBMS - UPDATE, WHERE, JOIN

May 15th, 2023

---

1 A Word of Warning

Make sure you read through the submission instructions thoroughly before you submit your project. Failure to comply with the submission instructions will result in a zero for this project.

Projects 4, and 5 will build off of the code you develop in project 3. So I highly recommend that you do not leave this project to the last day. I also highly recommend that you comment your code thoroughly so you can remember how your code works in the future.

2 Project Due Date

This project is due before June 1st at 11:59pm. Submission instructions for this project can be found at the end of this document.

3 Project Overview

This project will emulate the behaviour of the built-in python module, "sqlite3". Your module will be able to execute SQL statements corresponding to: creating tables, inserting rows, and selecting rows. Additionally, Project 3 will introduce, among some smaller changes, WHERE clauses, UPDATE statements, and JOINS, specifically a LEFT OUTER JOIN.

4 What You are Given

On D2L we have released some test cases, and a script to run the test cases. We have also released two 'regression' tests, which you can use to determine if your updates for Project 3 have broken and Project 2 functionality. You are responsible for creating your own test cases as well to test the robustness of your solution. There are edge cases that we did not release test cases for, and it is up to you to consider all of these and make sure your solution will work in all conditions.

5 What You Need To Do

You need to build the functionality for Project 3 on top of your Project 2. I highly recommend that you make a copy of your Project 2 in case things go wrong.

All of the code for this project must be in 'project.py'. There is no need to write any database output to any files. Each test will be done on a fresh database.

Your program is allowed to use any built-in modules of python, with the exception of sqlite3. sqlite3 is the reference implementation for this project, your output will be compared against it for correctness. Importing the sqlite3 module in project.py will be considered academic dishonesty for these project, which will lead to an automatic failure of this course.

You cannot use any modules of python that are not built-in, meaning you cannot use things like Numpy or Pandas.

If you use any resources that did not come directly from the class, you must specify and cite them in your code file. At the top of the starter code there will be a section to paste links to anywhere you got code or ideas from. This is to make sure that you are not plagiarising.

6 SQL Statements

All SQL keywords will be in ALL CAPS. For this project, the SQL keywords you need to handle are as follows (red are new for this project:)

CREATE TABLE
INSERT INTO VALUES
SELECT FROM ORDER BY
UPDATE SET DELETE
WHERE DISTINCT
LEFT OUTER JOIN ON

You can assume that all SQL keywords will be in all capital letters, as is seen above.

7 New SQL Syntax

7.1 Extensions over Project 2

Single quoted strings may have single quotes in them (escaped with a preceding single quote), you need to store and return the python string after removing this escaping. Example: 'My dog''s name' -> "My dog's name"

Any time a column name is used it may be qualified with a table name ("id" is the same thing as "students.id" if the table being SELECTed is "students").

The "*" column name may be qualified or included as one of multiple columns (SELECT *, name ..., SELECT student.*, name ..., etc).

7.2 INSERT INTO

The INSERT INTO statement can specify the columns and order of the VALUES being inserted: INSERT INTO (id, name) VALUES ... If not all the columns of a table are specified, the absent columns will have NULL inserted. Multiple rows can be inserted with a single INSERT INTO statement: INSERT INTO students (name, grade) VALUES ('James', 2.4), ('Yaxin', 3.9), ('Li', 3.87);

7.3 WHERE

SELECT (as well as UPDATE and DELETE) statements can have a WHERE clause that specifies what rows should be processes. SELECT * FROM students WHERE id > 4; To make things easier, all WHERE clauses will be in this form: WHERE column_name operator value. The column_name may be qualified. The operator will be one of: >, <, =, !=, IS NOT, IS. The value will be a constant (not a different column or expression). There won't be any parenthesis, ANDs or ORs in the projects.

7.4 UPDATE

You will need to add the UPDATE statement: UPDATE table_name SET col1 = value, col2 = value 2; An UPDATE statement changes the associated columns to the value. For simplicity, the value will always be a constant. If a WHERE clause is added, only those rows will be updated. UPDATE student SET grade = 4.0 WHERE name = 'James';

7.5 DELETE

DELETE works much like UPDATE, but instead removes all rows from a table, (unless a WHERE clause is added, in that case it only removes the rows which pass the predicate). DELETE FROM students; DELETE FROM students WHERE id > 4;

7.6 DISTINCT

The DISTINCT keyword specified you only want the unique values (no duplicates). For simplicity, the tests will only use the DISTINCT keyword with a single output column. Example: SELECT DISTINCT column_name FROM ... ORDER BY column_name;

7.7 LEFT OUTER JOIN

The LEFT OUTER JOIN is the only join you need to implement for this project. It will be of the form: SELECT columns FROM table_a LEFT OUTER JOIN table_b ON 'column from table_a' = 'column from table_b' ORDER BY columns; As shown above, the ON clause will always match on equality with a column from the left table then the right table. For simplicity, all column names will be qualified in queries involving joins.

8 Testing Your Code

8.1 Testing Yourself

We've provided a few sample sql transactions as well as a python script to test your code.

To run your code with the testing files provided:

```
python3 cli.py <filename>
```

Where the second argument is the .sql file to test. Each .sql file is a set of sql commands that compose one transaction. We also provide a way for you to compare against the ground truth, which is python's sqlite3 module. To run the ground truth, use the following command:

```
python3 cli.py <filename> --sqlite
```

Your output should be identical to the output from the sqlite3 module. In the cli.py file we have a few print statements so you know if you are using your own code or the ground truth sqlite3 module. You can remove this code if you like, in our final testing we will not have these print statements, of course.

You can, and should create your own test cases as well.

8.2 Instructor Testing

When we test your code, we will run it in almost the exact same way.

9 Assumptions and Guarantees

All tests will be legal SQL (no syntax errors, inserting into nonexistent tables or data type violations).
All select statements will have "FROM" and "ORDER BY" clauses
In "CREATE TABLE" statements every attribute will have a data type and no constraints
All table and columns names will start with a letter (or underscore) and be followed by 0 or more letters or numbers or underscores

10 A Few Notes

The instructor solution for this project is 565 lines of code, including a lot of comments. This project can be difficult, but if you understand how what is being asked and your project 2 is complete and reasonable, then the development should not be too difficult.

11 Submission Instructions

You will submit a file named "project.py" to the handin (handin.cse.msu.edu). All of your code should be in the "project.py" file. If you used any external resources (something other than class material), make sure that you cite the resource in a comment at the top of your "project.py" submission file. The top of the starter code also has space to put your name, netid, PID, and an estimation for how long the project took you to complete. Make sure you fill this out fully before submitting your project.