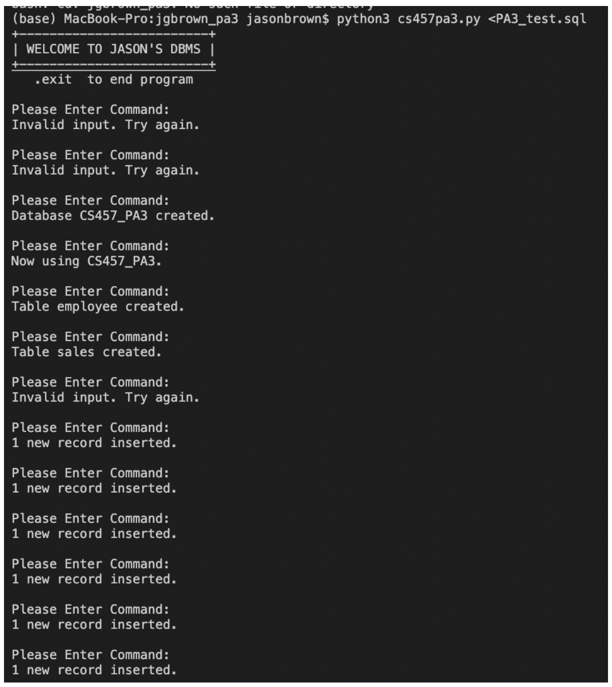
Jason Brown
CS 457 - Project 3
Oct 20, 2021

CS 457 - Project 3 Report

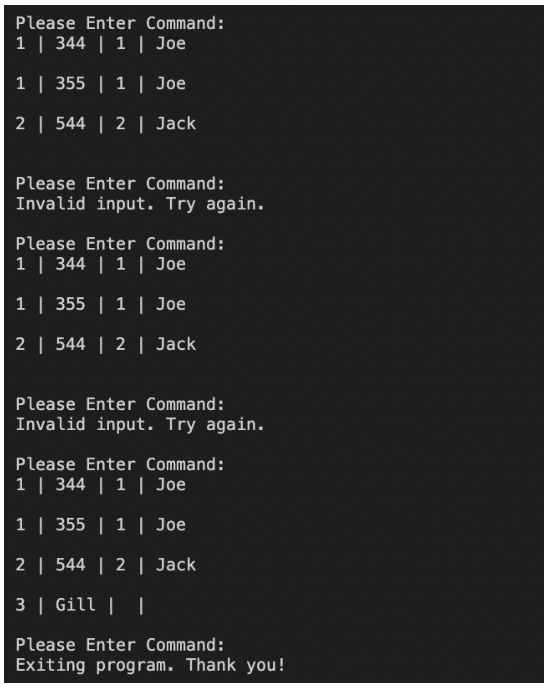
Running Program:

- *If using IDE like VSCode* → Clicking 'run' will run the program. Each line can be manually inputted.
- *Terminal command* → python3 cs457pa3.py <PA3_test.sql

Proof of Working Functionality with above command:



```
(base) MacBook-Pro:jgbrown_pa3 jasonbrown$ python3 cs457pa3.py <PA3_test.sql
+------------------------+
| WELCOME TO JASON'S DBMS |
+------------------------+
   .exit  to end program

Please Enter Command:
Invalid input. Try again.

Please Enter Command:
Invalid input. Try again.

Please Enter Command:
Database CS457_PA3 created.

Please Enter Command:
Now using CS457_PA3.

Please Enter Command:
Table employee created.

Please Enter Command:
Table sales created.

Please Enter Command:
Invalid input. Try again.

Please Enter Command:
1 new record inserted.

Please Enter Command:
1 new record inserted.

Please Enter Command:
1 new record inserted.

Please Enter Command:
1 new record inserted.

Please Enter Command:
1 new record inserted.

Please Enter Command:
1 new record inserted.
```



```
Please Enter Command:
1 | 344 | 1 | Joe

1 | 355 | 1 | Joe

2 | 544 | 2 | Jack


Please Enter Command:
Invalid input. Try again.

Please Enter Command:
1 | 344 | 1 | Joe

1 | 355 | 1 | Joe

2 | 544 | 2 | Jack


Please Enter Command:
Invalid input. Try again.

Please Enter Command:
1 | 344 | 1 | Joe

1 | 355 | 1 | Joe

2 | 544 | 2 | Jack

3 | Gill |  |

Please Enter Command:
Exiting program. Thank you!
```

Join Implementation:

The additional join functionality of this program is a combination of new join_where functions and appendations to previous functions like where and select. There is also a new helper function to ensure that the program can handle multiple files at the same time, ensuring to close each one after their use. The previous functions have added functionality so that they can work with multiple tables simultaneously. The tables can now be created as objects and can have different parameters called from each object. The previous functions are mostly the same other than this. The join function has multiple paths for inner and outer join, where it organizes the new table names and calls the where function on each, however, it still works if only one table is called.

Tuple Organization:

Tuples are stored in my tables in a column and row format, there will be numerous rows of tuples, each separated by " | ", spaces and a vertical line. To keep track of the tuple content, the table file is opened and read using ".readlines()". This command splits a file into an array of strings, separated by line breaks. This means that each iteration of this array will increment the row of the table. Once the row is determined, the element is split into another array, separated by the vertical line character " | ". The column count can be determined by iterating this new array or counting the number of vertical lines in the element.

Essential Tuple Commands:

To Insert a tuple into the table, the user must provide the name of the table to append and the row of the column to append. This table name is appended to the current directory, the table is opened for editing. I then locate the end of the file, add a line break, then add the string which the user inputted. The select function can then be used to print the new table with the addition of the new insertion.

The most important function of this new addition is the "where()" function. This function allows the program to query or edit parts of a table which are specified directly. The function is not complete yet, as in, it only has the capabilities which the test script requires, which is "=" equality and ">" greater than. The function splits the command input into the table name, parameter name which it wants to interact with, and the values it compares in order to interact with the right elements. It then checks for the comparison sign, either equality or greater than, then traverse the table and execute the command which is called: either 'delete', 'select' or 'update'.

Deletion works in a similar way to insertion, with the exception of the implementation of the "where()" function. After traversing the table with the where function, the program will find the correct line to delete, replace the line with whitespace, then delete the line breaks. Tuple modification does the same traversal with the help of the “where()” function, then it iterates through an array of the line which is separated by “ | “ vertical lines. Once the correct column is selected, it will delete the existing value and replace it with the new value. And finally, query, which is the select function which is now capable of selecting certain parameters rather than all of them, or “*”.

How the program organizes multiple databases:

I have created multiple global variables which keep track of the scope of the DBMS. This means it knows the layer of nested directories it is in. Every database or folder is placed in the master folder titled “Jasons_DBMS”. Every database which is created is placed inside the Jasons_DBMS folder and thus everything is organized and contained. After this, there can be multiple layers of nested directories and tables in each. The “Use” function manually traverses the layers so that the user can always access the files which need to be found. There is a “check_current_scope()” function which updates the global directory variables with the current path and scope of the user's progress. It is called numerous times to enter certain directories and make sure the entire program knows where it is.

How the program organizes multiple tables:

The program can easily handle many tables. The biggest obstacle I encountered with tables was keeping track of the current path and directory so that the program can traverse the master folder and find the right files/tables. It can create many tables in a database, it can append data to each one (many times), and it can delete any table in any database, in any order. The global directory variables are responsible for much of this, as they always know where the user wants to traverse, making the user easily find the files they want. When it creates a table, it goes to the correct directory, checks if there is already a file under the same name, creates a file for writing, organizes the parameters, then prints the parameters to the file, separated by “ | “.

How the program works:

Jason Brown
CS 457 - Project 3
Oct 20, 2021

The program begins with a printed title and instructions to exit the program at any time. A constant loop is then implemented which is always searching for a command from the user. If it is one of the expected commands, it will trigger functions which do certain tasks, otherwise it will tell the user about their invalid input, asking to try again. The available commands are to create a database, create a table, drop or delete a database, drop or delete a table, use (manually changes the directory path), alter a table, select (retrieve from a file), and exit the program. CREATE DATABASE calls the global directory variables to update the working directory and ensure every database is added to the folder, “Jasons_DBMS”. It collects the name of the database from the command line, updates the path, checks if the directory was already created, then creates the directory. CREATE TABLE calls the global directory variables to update the working directory and ensure the table is created in the correct place. It creates the file, named after the user input, opens it for writing, then organizes the parameters. It first takes away unnecessary characters and spaces, then counts the parameters by the number of columns then separates them in an array, then prints the contents of the array to the file (separated by vertical lines and spaces). DELETE DATABASE and DELETE TABLE functions virtually complete the same tasks. They first check the scope of the program, update the current working directory, and add the user input (names of the database or table) to the end of the directory paths. Then, after checking if the files and directories exist, they delete the files contained with os.remove() and delete the directories with os.rmdir(), then notify the user the task is completed. USE function is very unique as it manually changes the working directory of the program. It updates the current directory with the check_current_scope() then notifies the user that it has done so. This function is crucial before altering or creating tables so that the program can find the correct files to change. ALTER TABLE adds parameters to the pre-existing tables/files. It checks the current scope and updates the working directory and file path before reading the parameters given. It then organizes the parameters by comma and appends each one to the end of the file, still separated by “ | “. SELECT FROM function opens a pre-existing file, after updating the working directory and file path, and reads the contents to show to the user. It currently only works with SELECT * FROM <blank>

instead of selecting certain features from a file. So, as of now, it reads the entire contents of the file, titled after the word FROM, and outputs this content to the user. I am still in the process of an algorithm which will scan through the file to select certain data from the file instead of merely reading the entirety of the contents.