Author = Joshua Insorio

Date = 3/23/21

Project = PA2

Class = CS457

**CHANGELOG:**

* Added temporary class to hold database information
* Creating vector<vector> to hold temporary database for modification
* Allow fstream to read and insert into the vector
* Insert into function works
* function to handle return carriages
* save table function
* Update function
* Universal parser function
* Delete Function
* Select function

**IMPLEMENTATION:**

Tuple insertion = retrieve information from command line to be inserted into the txt file through fstream commands

Tuple deletion = locate the tuple from a vector class, and erase using the vector command

Tuple update = locate the tuple from a vector class and modify the desired data

Tuple query = locate the tuple from a vector class and query the desired data

**COMPILE:**

Use the provided make file and type “make”.

To run do ./main < PA2\_test.sql

**DOCUMENTATION:**

The main difference between PA1 and PA2 is the implementation of a class system. This class system is created after reading the inputs from the txt file, which was previously implemented in PA1. Alongside this addition of the data structure, additional functions were used to help manage(modify/update) the tuples in the database. These functions are “insertTemp”, “saveTable”, “rmReturn”, and “getSet”. These functions will be commented in the code using the formal C style commenting. To keep it short, “insertTemp” is the function essentially converting the database in the txt file into a vector class. Next, “saveTable” essentially saves the modified database within the vector class back into the txt file. “rmReturn” is the function dedicated to remove the return lines from the input, as the “update” command calls for several lines of code. And finally, “getSet” is a modified version of my preexisting “getData”, “getName”, and “getTblName” functions as it allows direct control over which word to parse from the input commands.

The tuple insert function strongly follows and is structured after my create table function. It is similar in which the function is given the command line, and parses it for the data, use fstream commands to open the file and write into it. The main difference is the addition of the “data.erase(remove\_if(data.begin(), data.end(), ::isspace), data.end());” command which is intended to remove all the whitespace from the original input. During manual testing this was not a problem as I was able to control the amount of whitespace. However, within the test file I had difficulty removing the whitespace as some could be considered ‘\t’ or ‘ ’, this solution works as a universal remover of whitespace.

The tuple update and deletion function mainly calls upon the highlighted added feature in PA2, the use of a vector class. It calls upon he “insertTemp” function, which reads all the database tuples from the txt file, insert them into a string array, then convert the string values to its respected int, char, and float versions. This is then inserted into the class via a parameterized constructor, and then pushed into a vector to house all the classes(tuples). From there a for loop is used to locate the desired tuple for modification or deletion, then update the tuple from the desired command that was inputted. In this case, not all the “cases” were made for all possible update and deletion possibilities except those needed for the test file however space was made for additional modifications in the future. And upon completing the modification/deletion, the “saveTable” function is called to save the modifications back into the txt file.

Finally, the tuple query command is a slightly modified version of the already implemented “select \*” command. It reads in the tuples from the txt file after determining the intended selections and locations. There are two if statements to determine if there will be multiple selections desired, and another for loop finding the location of the tuple from the vector class. Then if statements are used for the cases, however similar to the update/delete function, not all the possibilities are implemented but rather only those as required by the test file. And space was left for additional cases in the future as needed.