Jayam Sutariya

Dr. Zhao

CS 457.1001

13 April 2021

Design Document

For this project, I decided to the implement the functionalities using C++ as I am the most proficient in that language. The approach I took for the implementation of this project is to use standard input to get the command. In the main function, depending on the command, the user is then directed to the corresponding piece of code that will accomplish the command. This is mostly done in the main function using mostly if statements. Helper functions were used along the way to assist with certain functionalities in the main function. At all times, however, it was made sure that the code can be tested with individual inputs as well as a whole input file through the compiler.

The structure of the program is as follows: In the program directory, the user may create databases, and in those databases, there shall exist tables attached to that certain database. This model is exactly the same as the one shown in the demo by the TA. To manage these databases, try-catch blocks were used to make sure if a certain database already exists or if it does not exist at all. This prevented unprecedented use of a certain database while also preventing the misplacement of data.

The management and organization of the tables is also exactly the same as the demo provided by the TA. One try-catch was again used to declare the existence or the lack thereof for a certain table when it came to table creation or deletion. Another major try-catch was also used regarding the tables which made sure no tables may be altered, created, or deleted if no database is currently in use. This again preventing the misplacement of data and such. Overall, the management and organization are basically full proof at this point.

There are several functionalities present in this program. The first noticeable functionality is actual reading of the input and parsing it. For that, I used vectors and the sstream object called istringstream to split the input into several strings and then storing those words into the vector of strings. It was at this moment that I removed the semicolons and such. The first command functionality is CREATE DATABASE. For this, I used the system() function and passed in the string “mkdir db\_name” to execute that command in the console resulting in the creation of a database. In fact, for all creation and deletion functionalities, I used the system() function. To DROP DATABASE, I used the same method previously mentioned involving the system() function except this time I changed the string to “rm -r db\_name”. To USE a table, I changed a bool flag to true as soon as that command was passed in the console. I set the string curr\_dir to that database name. Similarly, for CREATE and DROP TABLE, I used the system() function except for the creation of the table, the string was “touch curr\_dir/tbl\_name”. Deleting the database and a table are the same in terms of code. For creating the table, I appended the data to the table using file i/o. For the SELECT command, I simply read from the table, and used cout to print the data to the terminal making minor tweaks to make it look how it should. For the ALTER command, I used the same implementation as CREATE TABLE where I append the data to the mentioned table. Lastly, for the .EXIT command, I simply used it as the condition for the do-while loop to exit the loop.

There were some changes for the new functionalities added for Project 2. For example, inserting tuples to the table is a little different now. I removed the parentheses like and applied that to the CREATE TABLE as well. I further replaced all quotation marks with empty spaces. Moreover, I also replaced the commas with vertical bar. I modified the implementation of the select command, where I made it print line by line instead of the previous functionality. Updating and deleting the tuples is where I changed the most. I inherited the table and stored the values, skipping the first line with attribute names, into 3 different vectors. To check for updates and deletes, I simply iterate through the vectors and apply the update. Since, I am using vectors, instance deletion was fairly simple as well. Afterwards, I would simply overwrite the new vector data into the old file. I also kept a counter to see how many updates or deletes were made as required by the assignment. For the last select command with the condition, I again used the vectors data as conditions and output that instead of straight out the file.

For Project 3, the functionalities for the inner and outer join were added. To differentiate between each of the joins, the input line was parsed into vector of strings and that vector of strings of analyzed to see if it is an inner or outer join. If it is an inner join, you create a nested for loop, one loop traverses through the Employee table and other traverses through the Sales table. Then I check if the id ith element of the first loop equals the Employee ID of the second loop. If that is the case, I output the first table tuple and then the second table tuple. However, if it is an outer join, there are slight changes. Same procedure except a count is kept. If the ID from the first table equals the Employee ID from the second table, then increase the count by 1. After this check if the count is 0, if it is 0, output, the left tuple followed by empty string. Then simply set count to 0 at the end of the loop and continue the process.

For the transaction functionalities such as begin transaction and commit, new code was added. When the user says begin transaction, the program checks if a file called “lock” exists in the current directory. If it does, set a bool variable called locked to true which was originally false. This variable is checked during update and the commit functions, if the value of locked is true, no updates are made, and the transaction is aborted. However, if the “lock” file does not exist, it starts the transaction and creates a file called “lock” while changing the value of the bool variable to true. The update function resumes as normal except the updates are not written to file yet. The updated data is written to file in the commit function. Since, the transaction is complete, the “lock” file is deleted while changing the value of the locked variable back to its original value of false.