Carli DeCapito, Sanya Gupta, Eugene Nelson
CS 457: Database Management Systems
Programming Assignment 1 - Documentation
February 22, 2018

Functionality

This programming assignment required the group to implement the basic functionality of a SQLite Database Management System. The program designed by the group implements basic functionality of both databases and tables. The program can create and drop databases and tables. Databases are also able to utilize a “use” function, which allow it to “create”, “drop”, and “alter” tables within the database that calls the use function. As previously mentioned, tables within databases are also able to be created, deleted, and altered.

The program works similar to a terminal (similar to SQLite), where it is continuously requesting input. Once input is typed and entered into the terminal, the program parses that command and performs the necessary actions. The program will continue to ask for input until the command “.EXIT” is inputted.

Organization

The program uses both files systems and abstract data structures to organize and keep track of the existing databases and tables.

File System Organization and Use

The program stores all file information on the computer disk. For each database created, a directory with an equivalent name is created in the current working directory. For each table created, a file with an equivalent name is created in the appropriate database directory. For each database and table that calls the “drop” function, the appropriate file or directory is removed from the disk. Each file contains its corresponding attributes, such as a1 int or a2 varchar(20). These attributes are written to the appropriate table file and can be altered or dropped. By using the disk to store database and table information, this ensures that after the program ends, the database information is still available to the user.

Class Organization and Use

The program stores all the file and directory names into a vector of databases, where each database contains a vector of tables. For example, if database db_1 contains tables tbl_1 and tbl_2, and database db_2 contains table tbl_1, the physical representation would look like the drawing in Fig. 1. The vector of databases was only

designed to keep track of which files and directories exist. No further information besides the directory and filenames is store in these abstract data types.

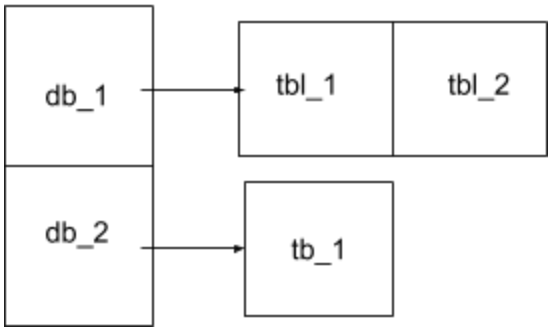

Fig. 1: A physical representation of the layout of the database and table information storage. Each database is an element of a vector, and each table is an element of a vector. (Note: The arrows are not pointers. They simply indicate vectors within vectors.)

Since at the end of every program, the data in these abstract data types are removed from memory, there is not way to ensure that when ".EXIT" is called the database information is saved saved for future use. On the startup of every program run through, the disk is checked to see if any database or table information exists within it. If it does, that data is stored into these vectors and can allow for further modification and manipulation of the existing databases and tables (directories and files).

Database Organization

Create

The Database Create functionality is responsible for creating a new database directory on the disk and for push_back of the database onto the database system vector. For example, to create a database, the user will type the following into the terminal:

CREATE DATABASE <database name>;

Before creating the new database, the program first must make sure that there are no existing databases with the same exact name. For example, if the following input was provided by a user, it would result in an error:

CREATE DATABASE db_1;
CREATE DATABASE db_1;

The terminal layout will result in an error because db_1 already exists after the first input. If there is an existing database with the same name, then an error indication will

occur. Otherwise a new directory will be created with the appropriate name and a new database will be pushed on the database system vector.

Drop

The Database Drop functionality is responsible for removing an existing database directory on the disk and for removing the database from the database system vector. For example, to remove a database, the user will type the following into the terminal:

DROP DATABASE <database name>;

Before removing the database, the program first must make sure that there is an existing database with the same exact name. For example, if the following input was provided by a user, it would result in an error:

CREATE DATABASE db_1;
DROP DATABASE db_1;
DROP DATABASE db_1;

The terminal layout will result in an error after the third input because db_1 does not exist after the second input. If there is not an existing database with the same name, then an error indication will occur. Otherwise the appropriate database will be deleted and removed from the database system vector.

Use

The Database Use functionality is responsible for ensuring that all following table manipulation is from that given database until another database use is specified. This functionality ensures that the database provided already exists and sets the current database variable as the provided input. For the user to specify which database he/she would like to use, the following command is appropriate:

USE <database name>;

Table Organization

Create

Table Create function creates a file that contains the attribute(s) name and attribute type(s) that is read in from the command line. For example, the table create function is called once a user types the following within the terminal program:

CREATE TABLE <Table Name> (<attribute name 1> <attribute type 1>, <attribute name 2> <attribute type 2>);

The program check if the table already exists within the database, if so the program will give an error; otherwise the program will create the table and parse in and store the attributes in the disk. If a user specifies a table with the same name for the attributes more than once, then the table will be deleted and an error will be given. For example, the following input will cause the program to give an error because a1 attribute has been declared more than once, regardless of the type:

CREATE TABLE tbl_1 (a1 int, a1 float);

As a result, the table will fail to be created, and will result in an error display in the command line. Likewise, if the user does not input the attribute without parentheses while creating the table, then an error will be displayed. However, if the user creates the table with the command below:

CREATE TABLE tbl_2();

an empty table without any columns will be created.

Drop

Table Drop function will delete an existing table that belongs to the database in use. The program first checks if the given table exists in the current database. If so, then it will go ahead and delete the table from the disk by deleting the file that stores that table's contents in the database directory. This is done through the command displayed before:

DROP TABLE <table name>;

However, if the user tries to delete a table that does not exist in the directory, the program will notify the user through an error.

Alter

Table Alter function will add an attribute to an existing table in the database. The program will check to see if the table specified by the user exists in the database and if so, will add the new attribute to the table. This is done through the command below:

ALTER TABLE <table name> ADD <attribute name> <attribute type>;

If the user wants to add more than one attribute to the existing table, it can be done by separating each attribute by a comma. For example:

ALTER TABLE <table name> ADD <attribute name 1> <attribute type 1>,
<attribute name 2> <attribute type 2> ;

However, if the user tries to alter the table by inputting the same attribute name or adding to a table that does not exist, then an error will be displayed.

Select

Table Select function displays what attributes are stored in the table which is specified from the user. This done through the command displayed below:

SELECT * FROM <table name>;

Then the program will display all the attribute names and types stored in that table separated by "|" symbol. If the user queries from a table that does not exist in the database, then the program will display an error.

How to Run and Compile

The user must first navigate to where the program is stored on the computer through terminal. A makefile is provided to compile this program so all the user has to do is type:

```
make
./main
```

The program should now be running. The user should now be prompted by a ">" symbol, which indicates that it is expecting user input.

Special Circumstances

The program requires that all SQL commands and keywords must be in caps:

eg. CREATE, DROP, USE, ALTER, SELECT * FROM, .EXIT, DATABASE, TABLE, ADD

Likewise, each line should end with a semicolon except for when exiting the program. If the user wishes to terminate the program, .EXIT must be type and does not include a semicolon at the end. Spacing also matters. This program follows the spacing convention that is displayed in the provided SQL test file.