Evan Grill

Jonathan Weatherspoon

CS 457 -- DBMS

Programming Assignment 4

May 8, 2018

How our Program Stores Data

Databases are represented on the disk a folder with the directory name matching the database name. Similarly, tables are stored in the folder as individual files with the extension ".tbl". In each table file, the first line of the file is the schema, delimited by a "|" character. Subsequent lines in the table file are the individual rows stored with columns delimited by a "|" character as well. The program opens and transfers the table to memory on table access (any query on the table), and saves on any query when outside of a transaction or on commit of a transaction.

High Level Explanation of Implementation

Tuple Insertion

To insert data into a table, we parse the INSERT query into two main components: the table name and the values to insert. We first check to see that there is a value for each attribute in the target table, and if so, add a row into the object that represents that table using the values supplied in the statement.

Tuple Deletion

In order to delete rows in a table, we first parse the DELETE statement into three main components: the table name, a list of attribute names, and a conditional statement. We first get a handle to the table object representing the table, then get a list of rows that corresponds to the condition supplied. We then remove that list of rows from the rows stored in the table object. If a condition was not supplied, we delete all rows from the table.

Tuple Modification

To handle data modification, we first parse the UPDATE statement into three parts: the table name, a dictionary that maps column names to values, and a conditional statement. To update the table when a WHERE clause was supplied, we iterate through the rows stored in the table and if the condition matches for that row, we use the dictionary to change the values of the columns. If no WHERE clause is specified, all rows in the table are updated with the new values.

Query

For the select statement, we use the same function as the DELETE statement to get a list of rows from the table that correspond to the WHERE clause. We then display a header showing the schema for only the rows supplied, and the corresponding data. If no WHERE clause was supplied, the SELECT statement will act as it did in PA1, but will now also display the data in the table.

Inner Joins

To implement inner joins, we implemented parsing for both explicit inner joins (using the "inner join" keywords) as well as implicit inner joins (using a where conditional). The algorithm operates the same as the pseudocode outlined in the assignment document. For each tuple in the left hand table, iterate through the tuples in the second table to find the matching value in the specified attribute, when found, both tuples are added to a new temporary table which is then printed.

Outer Joins

To implement outer joins, we kept track of what type of join was specified in the SELECT statement when we parsed it using special constants. Once we did that, we grabbed the tables that were being joined from our database object (see above), and then merged the tables into one large temporary table using a nested loop outer join algorithm. Once we had the merged table, we gathered the data from it as we would in a normal select statement without joins and displayed that to the user.

Aliasing

To implement aliasing functionality in select statements, we created a dictionary which mapped alias names to table names when we parsed the select statement. When determining which tables to select from, we reference the alias map if an alias was present in the select statement.

Transactions

Transactions are implemented in this version of the project. On a "begin transaction" query, the database (all tables) is locked and lock files are created with the process ID. (tablename.pid) When a query is run that alters a table (insert, update, delete, and alter) locks are checked. If the process running the query is the one that put the lock in place, then the query is run. If not, an error is returned. When a "commit" query is run, if the process running it owns the locks, then the tables are all saved. If the process does not own the locks, another error is reported.

How To Use This Code

This project is written in python3. To run the code, type the following:

```
$ python3 main.py
```

Stdin/Stdout are utilized so passing in files with a list of SQL commands works:

```
$ python3 main.py < PA3_test.sql
```

This code was designed to take advantage of features in python 3.6. Using versions earlier than this may result in undefined behavior. One example of this would be that table schema is stored out of order in versions of python earlier than 3.6.