

Database System Implementation

Project 5 – Complete Database System

Developed by –

Ghodkari Chowdary, Raghunatha Rao - UFID: 6218-1051
Mullapudi, Aseesh - UFID: 9175-1971

Aim of this project is to complete our Database System end to end, we support CREATE TABLE, INSERT INTO, DROP TABLE and SELECT operations. To support it we have made required changes to the parser and lexer.

We have extended the support for our database to execute optimized query plans resulted from previous milestone. We have added support for query tree to invoke relational operations depending on the query plan and pipelines are built to pass information among multiple operations. Our Database System now handles the queries to create and drop tables, insert tuples into tables and handle predicate queries.

YOUTUBE LINK FOR THE DEMO:

<https://youtu.be/JW68O6g5ong>

Implementation Details:

Main.cc Code Structure:

- Initially the program reads the test.cat file to know the location of the tbl and bin files
- Program now asks for query the input query is processed using the lexer and parser to fill the required data structures.
- Depending on the input we execute the required methods.
- ***int QueryTree::createTable(CreateTable *createTable);***
 - Method is invoked if the query is create table. Using the fields in the create table data structure, DBfile is create. According to the fields a heap or sorted file and metadata is created using create function.
- ***int QueryTree::insertFile(InsertFile *insertFile);***
 - Method is invoked to load tuples into DBfile. According to type of dbfile from metadata, load function from heap or sorted file is called accordingly.
- ***int QueryTree::dropTable(char *dropTable);***
 - Method is invoked to drop the table. DBfie and the associated metadata file is deleted.
- ***int QueryTree::executeQueryTree();***
 - Method is invoked to execute the query, before this is done we load all the statistics required using ***void Statistics::loadStatistics();***
 - An optimized query tree built using ***QueryTree* Optimizer::getOptimizedQueryTree();***
 - Now the query tree is executed using the Post Order traversal on query tree.

- Each node now calls the required relational operations to execute the query.

(GTests) gtest.cc:

This file implements google tests for 3 testcases using the functions for Create table, load the file into table and drop table.

Instructions to Run the Code:

To run the overall project, follow the below steps:

Compile and run main.cc:

Ensure there is Catalog file and the bin files, tbl files required present in the root folder.

1. make
2. ./a5.out

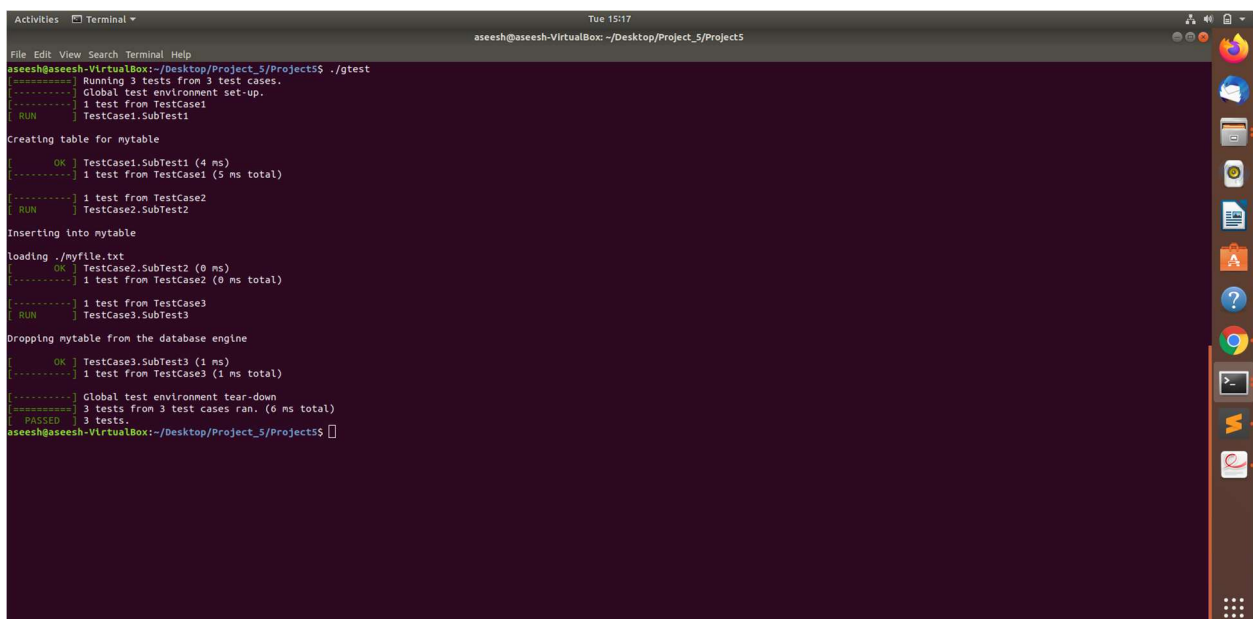
Every time a query needs to be given to the database; we need to give ./a5.out to run the database.

Instructions to run gtests:

Compile and run code gtest.cc:

1. make gtest
2. ./gtest

The screenshot of ./gtest is below



```
aseesh@aseesh-VirtualBox: ~/Desktop/Project_5/Project5
aseesh@aseesh-VirtualBox:~/Desktop/Project_5/Project5$ ./gtest
[*****] Running 3 tests from 3 test cases.
[*****] Global test environment set-up.
[*****] 1 test from TestCase1
[ RUN      ] TestCase1.SubTest1
Creating table for mytable
[ OK      ] TestCase1.SubTest1 (4 ms)
[*****] 1 test from TestCase1 (5 ms total)
[*****] 1 test from TestCase2
[ RUN      ] TestCase2.SubTest2
Inserting into mytable
loading ./myfile.txt
[ OK      ] TestCase2.SubTest2 (0 ms)
[*****] 1 test from TestCase2 (0 ms total)
[*****] 1 test from TestCase3
[ RUN      ] TestCase3.SubTest3
Dropping mytable from the database engine
[ OK      ] TestCase3.SubTest3 (1 ms)
[*****] 1 test from TestCase3 (1 ms total)
[*****] Global test environment tear-down
[*****] 3 tests from 3 test cases ran. (6 ms total)
[ PASSED  ] 3 tests.
aseesh@aseesh-VirtualBox:~/Desktop/Project_5/Project5$
```