Project 5: DataBase Demo – Putting It All Together

Group Members:

1. Smrati Pandey, UFID: 4323-9459
2. Wins Goyal, UFID: 7357-1559

Instructions for Execution / Code compilation and to run tests:

I. To extract folders

a) Extract the contents of the folder, named *SmratiPandey_WinsGoyal_p5.zip*.
b) Open the terminal.
c) Navigate into the extracted folder to go to the 'project' folder.

II. To run the tests

Perform following these commands:

```
$ make clean
$ make a5.out
$ ./a5.out
```

a) A Database session gets fired up, as these commands are executed.
b) After the last command, the console will print ***'Enter a Query: (when done press ctrl-D):'***.
c) Copy/paste the Query that we want to test for this databases, and press ***'ctrl-D'***.
d) The result for the corresponding query will display in the console.
e) Keep up doing step (c), till we are testing the database.
f) To stop testing and to shutdown the database, write ***STOP;*** *(with 'semi-colon')* and press Enter.

Summarized Documentation of the implemented methods and functions:

By implementing the requirements of this project, we are putting together all of the work we have done till now in our previous assignments and finally, we will have a working database system that is able to be 'fired up' as a session and be 'shutdown' after processing any changes or queries that are fed to this database. This database is able to support basic functionalities like "CREATE TABLE, INSERT, DROP TABLE, SELECT TABLE, SET" through queries. We already had the Query-plans generated in assignment 4.2 In this project, those Query-plans will be finally executed. The code for this written in *QueryRunner.cc*, *QueryRunner.h* while many other code changes have also been in relevant files. All operations will be performed when the database session is running.

Following is the Query-type format for the Database:

1. CREATE TABLE →
   CREATE TABLE mytable (att1 INTEGER, att2 DOUBLE, att3 STRING) AS HEAP;
   As the support for 'sorted file type' is optional. We have not included the related syntax in this project.
2. INSERT INTO →
   INSERT 'myfile' INTO mytable;
3. DROP TABLE →
   DROP TABLE 'mytable';
4. SET OUTPUT → *## to write the result either to 'Screen'/'to File'/'None'*

SET OUTPUT STDOUT; *(OR)*
SET OUTPUT 'myfile'; *(OR)*
SET OUTPUT NONE;

5. SELECT ….→
   This could contain various combinations of different Relational Operators like *Joins, GroupBy, Select, Distinct, Project, Aggregates like Sum (distinct or general)*

6. STOP; →
   This will shutdown the database.

We have created a 'Case by case' Query Execution. The following 'Flow Pipeline' shows a brief implementation of the methods showing the support for above mentioned Queries:

1. First, we need the Database Session is already turned on. And, the use prints a Query to the console.
2. A *switch* case checks for cases like: Create, Insert, Select, Set, Drop.
   i. '*Create*': A schema is loaded/created and written into a file. *DBFile* is created by calling a function for the same.
   ii. '*Insert*': The schema that was written into file or was already there in the file is loaded. *DBFile* is also loaded with it. The file is inserted into the *DBFile*.
   iii. '*Select*':
      a) This will open all the *DBFiles* (related to tables mentioned in the query).
      b) Query Plan is created by reading the Statistics from the statistics file.
      c) Query Runner will executed this plan by checking if the output_type is '*no_out*'. If so, the Query Plan is printed immediately and all tables are closed. If not, the Query is executed.
      d) If Query is being executed, the executor checks whether the output_type is '*file_out*'. If not, the Query Plan is printed on the screen/console immediately and all tables are closed.
      e) If the output type is '*file_out*', then the output is written in the file. And, all tables are closed after this.
   iv. '*Set*': An output type and output file is set for this query.
   v. '*Drop*': The schema file and corresponding *DBFile* are removed. Also, the related metadata file is removed.
3. When the user don't want to use the Database, it can be shutdown afterward by the *STOP;* query.

Screenshots of Running All Test Case Results: (from *output41.txt*)

I. **Query-1:**

```
Enter a Query: (when done press ctrl-D):
SELECT n.n_nationkey
FROM nation AS n
WHERE (n.n_name = 'UNITED STATES')

n.n_nationkey: [24]

Query ran.
```

II. **Query-2:**

```
Enter a Query: (when done press ctrl-D):
SELECT n.n_name
FROM nation AS n, region AS r
WHERE (n.n_regionkey = r.r_regionkey) AND (n.n_nationkey > 5)

n.n_name: [KENYA]
n.n_name: [MOROCCO]
n.n_name: [MOZAMBIQUE]
n.n_name: [PERU]
n.n_name: [UNITED STATES]
n.n_name: [JAPAN]
n.n_name: [CHINA]
n.n_name: [INDIA]
n.n_name: [INDONESIA]
n.n_name: [VIETNAM]
n.n_name: [FRANCE]
n.n_name: [ROMANIA]
n.n_name: [RUSSIA]
n.n_name: [GERMANY]
n.n_name: [UNITED KINGDOM]
n.n_name: [JORDAN]
n.n_name: [IRAQ]
n.n_name: [IRAN]
n.n_name: [SAUDI ARABIA]

Query ran.
```

III. **Query-3:**

```
Enter a Query: (when done press ctrl-D):
SELECT SUM (n.n_nationkey)
FROM nation AS n, region AS r
WHERE (n.n_regionkey = r.r_regionkey) AND (n.n_name = 'UNITED STATES')

SUM: [24]

Query ran.
```

IV. **Query-4:**

```
Enter a Query: (when done press ctrl-D):
SELECT SUM (n.n_regionkey)
FROM nation AS n, region AS r
WHERE (n.n_regionkey = r.r_regionkey) AND (n.n_name = 'UNITED STATES')
GROUP BY n.n_regionkey

SUM: [1]

Query ran.
```

V. **Query-5:**

```
Enter a Query: (when done press ctrl-D):
SELECT SUM DISTINCT (n.n_nationkey + r.r_regionkey)
FROM nation AS n, region AS r, customer AS c
WHERE (n.n_regionkey = r.r_regionkey) AND (n.n_nationkey = c.c_nationkey) AND (n.n_nationkey > 10)
GROUP BY r.r_regionkey

SUM: [45]
SUM: [43]
SUM: [57]
SUM: [73]
SUM: [56]

Query ran.
```

GTests:

- Perform following commands to execute the GTests:

```
$ make clean
$ make gTestDB.out
$ ./gTestDB.out
```

a) ***ExecuteQueryStatusTest***

This test validates the ExecuteQuery status for different queries like 'CREATE, SET, SELECT, DROP, INSERT, STOP'. Except for STOP, the status should return 0 for all the queries. For STOP, the status should return 1. If the status are returned as expected, this means that the queries are being executed properly.

b) ***OutputTypeTest***

This tests the output type that is required for the "SET" query. So, the output type should be *STDOUT* if the output should be printed on console. The output type should be *FILE* if the output to be written into a file. It should be *NONE* if nothing is mentioned.

Following is the screenshot of these two *Gtest-cases* in execution (terminal):

```
g++ -O2 -Wno-deprecated -o myGtest.out Record.o Comparison
DBMgmt.o MyGtest.o y.tab.o lex.yy.o -ll -lpthread -lgtest
[==========] Running 2 tests from 1 test case.
[----------] Global test environment set-up.
[----------] 2 tests from DBMgmt
[ RUN      ] DBMgmt.ExecuteQueryStatusTest

Output mode changed.
[       OK ] DBMgmt.ExecuteQueryStatusTest (0 ms)
[ RUN      ] DBMgmt.OutputTypeTest

Output mode changed.

Output mode changed.

Output mode changed.
[       OK ] DBMgmt.OutputTypeTest (0 ms)
[----------] 2 tests from DBMgmt (0 ms total)

[----------] Global test environment tear-down
[==========] 2 tests from 1 test case ran. (0 ms total)
[  PASSED  ] 2 tests.
Backups-MBP:Project backup$
```