Project 4 (part 1): Statistical Estimation

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_p41.zip*.
b) Open the terminal.
c) Navigate to the extracted folder.

II. To run the tests

Perform following these commands:

```
            $ make clean
            $ make a4-1.out
            $ ./a4-1.out {x}  ## To run any test case b/w 1 to 11 replacing “x”
            $ ./runTestCases.sh  ## Runs test cases: 1,2,5,10 & 11 in the script
```

Last command will produce “***output41.txt***” file

Summarized Documentation of the implemented methods and functions:

This project required us to implement from scratch the Statistics class for ‘Statistical Estimation’, which will be used by the Query Optimizer later so as to make optimal decisions on different plans in order to execute a query. Not only this class will store statistical information about database’s attributes and relation, but this information will be used its *Apply* and *Estimate* methods for simulation the query operation and estimating best results.

The source codes for this assignment are written in *Statistics.h* and *Statistics.cc*. A *Statistics.txt* file is also generated by the class, that contains the statistical information, which is re-read by the class methods to do the estimation.

I. Class Data members (of *Statistics* class, in *Statistics.h*)**:**

- ***rel_Map*** (type: *map <string, relOfStructure>*): Maps structural information of a relation to that relation’s name. ***relOfStructure*** (a container class, type: *struct*) stores the structural information:
  - *map <string, unsigned long int>* ***differentAttributes***: Number of distinct values of an attribute.
  - *unsigned long int* ***NumberOfTuple***: Total number of tuples in a relation.
- ***relAtts*** (type: *map <string, vector<string>>*): Maps an attribute to a list of relations that it occurs in.
- ***infoVector*** (type: *map <int, vector<string>>*): Maps *PartitionNum* to the relation vector (*rel_vector*).
- ***AttributeStructure*** (type: *struct*): It has two data members storing information for each attribute object.
  - int ***AttributeCount*** & long double ***estAtt***;

II. **Class Operational Methods**: (i. Operations on Container Class, ii. Operations for Estimation simulation)

1. *Statistics****(Statistics &copyMe)****;* It is a copy constructor of the Statistics class with input parameter as the statistics object (&*copyMe*) that is to be copied. This performs a deep copy of all the hashmap based data structures in this object to a new copy of the object. Following helper methods (declared and defined in *Statistics.h*) are used to retrieve those structures and values:
    - *map<string, relOfStructure>* **** GetRelMap()***
    - *map<string, vector<string>>* **** GetRelAtts()***
    - *int* ***GetPartNum()***
    - *map<int, vector<string>>* **** GetPartInfo()***

2. *void* ***CopyRel****(char *o_Name, char *n_Name);* This operation (or method) copies the relation with old name (that includes all its attributes and statistical information ← in the hashmaps) and stores all that data to a newly created relation with the new name.

3. *void* ***AddAtt****(char *relName, char *attributeName, int diff_num);* This function adds an attribute to the *Statistics* object. If it's the first time the attribute is being added to the object, the map *rel_map* updates and adds to its attribute information. If the attribute already exists in the object, *diff_num* is updated for that attribute in that relation. This is done in *relAtts* map with *attributeName* as the key. If the *diff_num* is -1, the number of distinct values for that attribute in the relation is same as the number of tuples in that relation.

4. *void* ***AddRel****(char *relName, int numOfTuples);* This function adds relation to the *Statistics* object with its number of tuples. If present already in the object structure, the method simply updates the *numOfTuples* for that relation. The information is stored in *rel_Map* with *relName* as the key.

5. *void* ***Read****(char *source);* This operation enables the *Statistics* object to read the data from *source* file (which is the '*Statistics.txt*' file) and store all that information to the current object (i.e. itself). If the file doesn't exist, the error is handled by interpreting the text file as empty.

6. *void* ***Write****(char *source);* This operation enables the *Statistics* object write to it's current stored data (all hashmaps and statistical values) into the *source* file (basically, creating '*Statistics.txt*').

7. *void* ***Apply****(struct AndList *parseTree, char *name_relations[], int numbersToJoin);* This function uses the statistical estimates to simulate a join of all the relations in the *name_relations* list. It doesn't actually does the join, but uses the predicates listed in the *parseTree*. All relation names in the *name_relations* list are their own singleton set or list. If there are multiple relation names in their individual set (subset of the *name_relations* list), first the join is estimated and simulated on relations in their own subsets. After simulating their individual joins, these relations are considered as one combined relation, then after an overall join is simulated for the *name_relations* list. Similar is done for the 'selection' operation. After these simulations are done, corresponding statistics object information is updated, using *infoVector* map with *PartitionNum* as the key.

8. *double* ***Estimate****(struct AndList *parseTree, char *name_relations[], int numbersToJoin);* This function does simulations of join / selection operators but doesn't modify or update the current statistics object.

Format of ‘*Statistics.txt*’:

The *Statistics* class object will serialize its stored data structure values as String into the ‘*Statistics.txt*’. This object contains hashmaps for Relation Names, Relation Attribute Names and Information about Relations. The file is easily readable by any human and easy to be re-read by the object to load itself with the information present in the file. Following is the explained format about the *Statistics.txt* file.

The file writes the information about each relation table that are present in the *name_relations* list. For now, we consider the each individual element in that list is a singleton subset, i.e. only one relation in each subset of the list. So, the combined relation after Joins / Selection operation is the relation obtained from all those relations mentioned in the list. This combined relation can be mapped to a group name. In this case, number of groups is same as the number of combined or resulted relations.

File starts with a ‘*\n’ character* before starting to write about any relation. For formatting and easy reading purposes, ‘----------’ is printed at the start and ‘__________’ is printed at the end of each relation information.

The information is written for all relations mentioned in the *name_relations* list.

- *\n’ character* is printed for every relation in the file.
- After ‘----------’, first line mentions the “*Relation Name*” and “*total number of tuples*” in that relation separated by tabs. And, as the *Statistics* object re-reads the file to simulate joins / selections, as all the relations are to be joined, the number of tuples is updated in the text file with that of the resulting joined relation.
- After the first line, the next lines contain the information about the attribute names in that relation. Only those attribute names are mentioned which were added to the *Statistics* object for that relation and that many lines follow after the first line. This ends with ‘__________’
- The number of lines that follow after the first line mention the ‘*Attribute Name*’ in that relation and ‘*number of distinct values*’ for that attribute in that relation separated by tabs. The attribute name is written as the <*first letter of the Relation Name*> + “_” + <*Attribute Name*>.
- This process follows for all the relations that need to be mentioned in the text file.

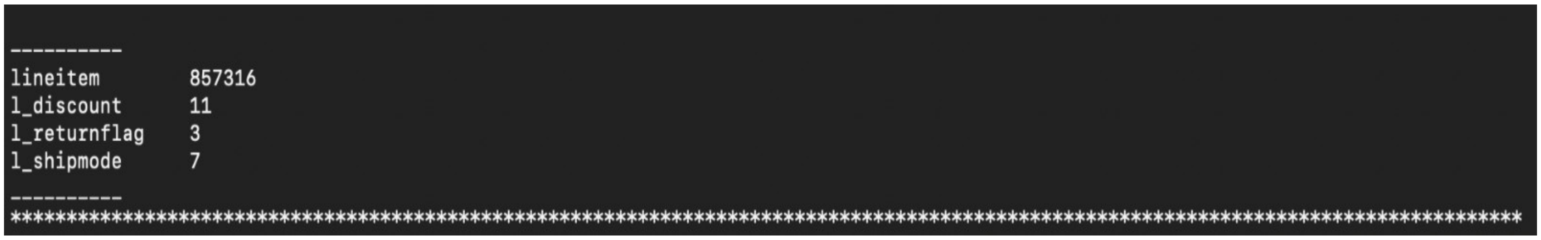
For example,
Let us assume we have 3 relations (say, *rrel*, *srel*, *trel*) with their respective attributes & information all added to the *Statistics* class object. We are assuming that the all the tables are in singleton sublist in the *name_relations* list. So, no table is pre-joined and thus, all the relations are basic.

The Statistics.txt file for this should look like as follow:

```

----------
rrel        20000
r_attr1     2000
r_attr2     100
__________

----------
srel        20000
s_attr1     2400
s_attr2     100
__________

----------
trel        20000
t_attr1     25
__________
```

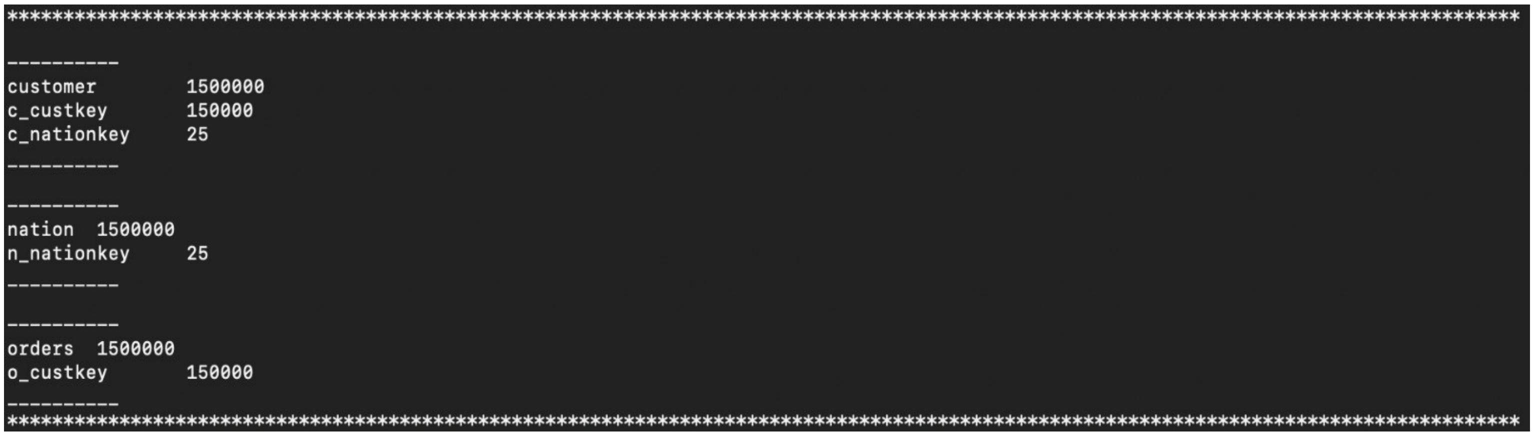
Screenshots of Running All Test Case Results: (from *output41.txt*)

I. **Query 1:**



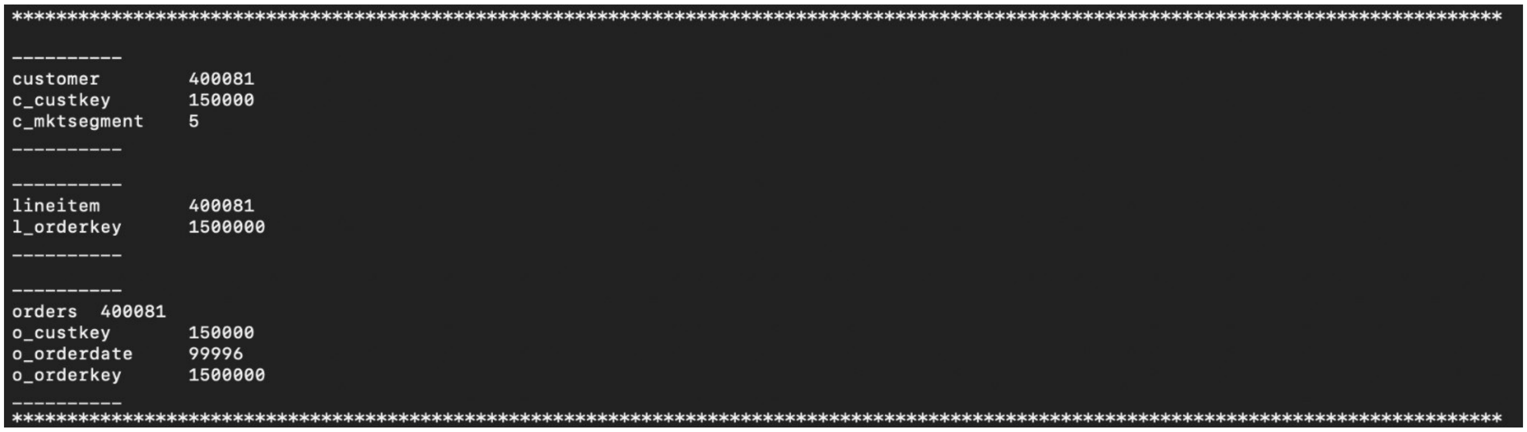
```
----------
lineitem        857316
l_discount      11
l_returnflag    3
l_shipmode      7
----------
**********************************************************************************************************************************************
```

II. **Query 2:**



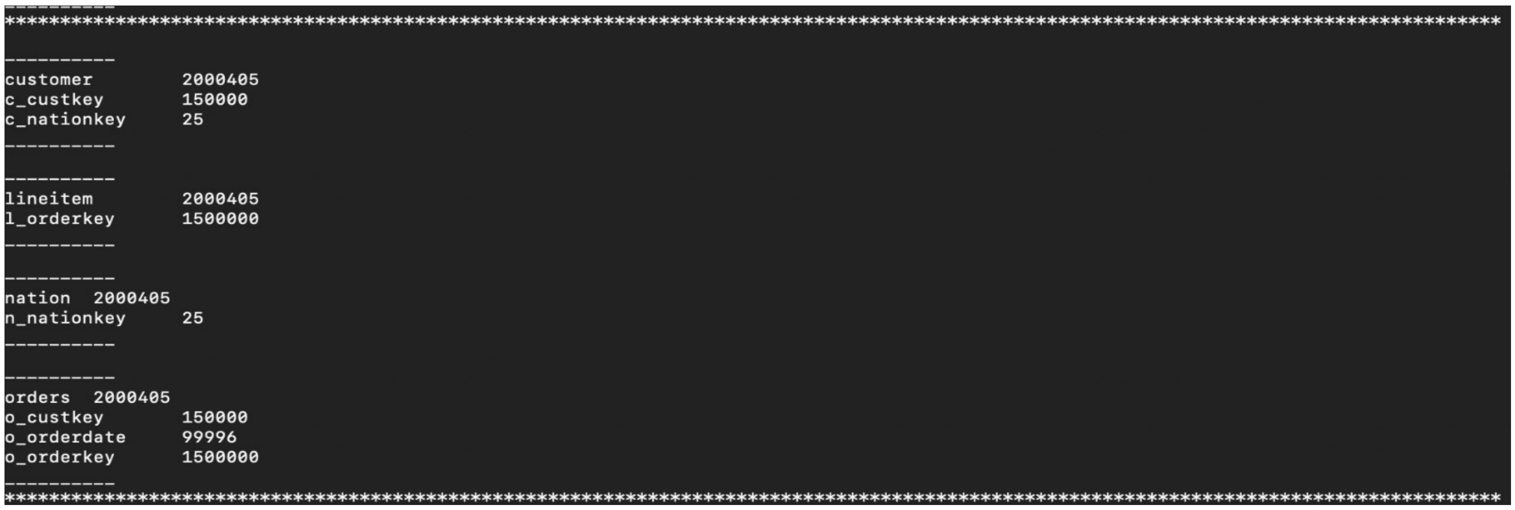
```
**********************************************************************************************************************************************

----------
customer        1500000
c_custkey       150000
c_nationkey     25
----------

----------
nation  1500000
n_nationkey     25
----------

----------
orders  1500000
o_custkey       150000
----------
**********************************************************************************************************************************************
```

III. **Query 5:**

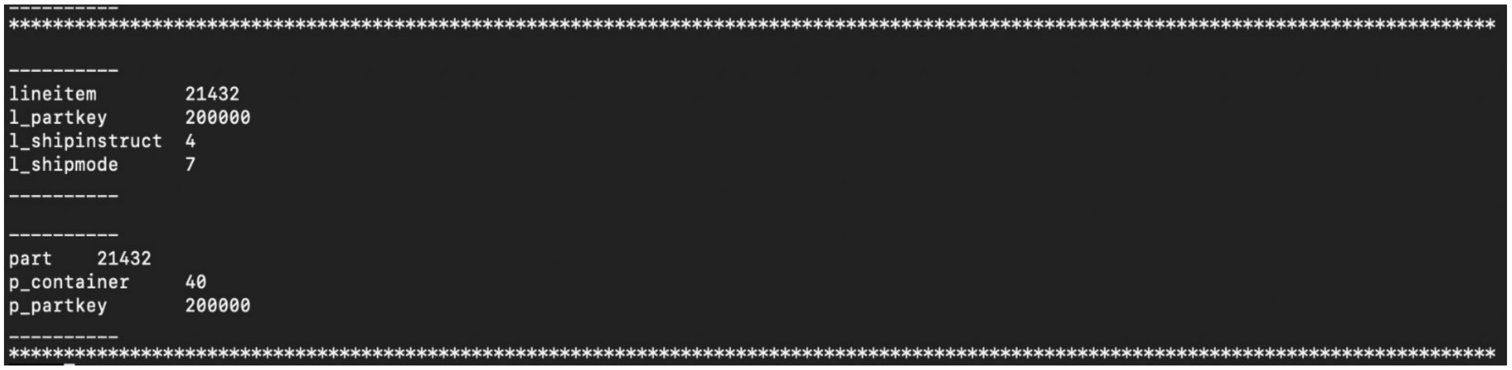


```
**********************************************************************************************************************************************

----------
customer        400081
c_custkey       150000
c_mktsegment    5
----------

----------
lineitem        400081
l_orderkey      1500000
----------

----------
orders  400081
o_custkey       150000
o_orderdate     99996
o_orderkey      1500000
----------
**********************************************************************************************************************************************
```

IV. **Query 10:**


```
----------
****************************************************************************************************************************************

----------
customer        2000405
c_custkey       150000
c_nationkey     25
----------

----------
lineitem        2000405
l_orderkey      1500000
----------

----------
nation  2000405
n_nationkey     25
----------

----------
orders  2000405
o_custkey       150000
o_orderdate     99996
o_orderkey      1500000
----------
****************************************************************************************************************************************
```

V. **Query 11:**


```
----------
****************************************************************************************************************************************

----------
lineitem        21432
l_partkey       200000
l_shipinstruct  4
l_shipmode      7
----------

----------
part    21432
p_container     40
p_partkey       200000
----------
****************************************************************************************************************************************
```

Note:

- No attribute added “o_orderdate” for relation “orders” in q5.
  Added: s.AddAtt(relName[1], "o_orderdate",99996);
- No attribute added “l_receiptdate” for relation “lineitem” in q7.
  Added: s.AddAtt(relName[1], "l_receiptdate",198455);
- No attribute added “o_orderdate” for relation “orders” in q10.
  Added: s.AddAtt(relName[1], "o_orderdate",99996);

GTests:

- Perform following commands to execute the GTests:

```
$ make clean
$ make myGtest.out
$ ./myGtest.out
```

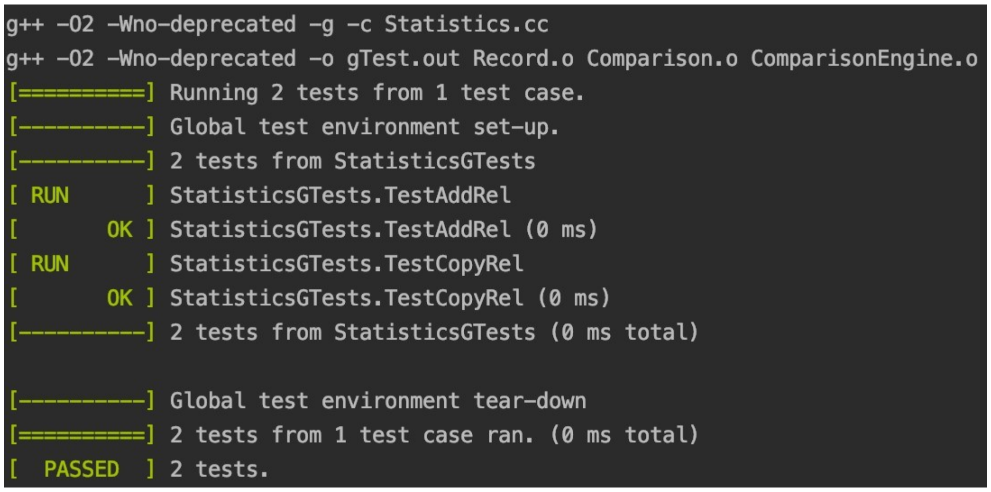
- Note: Before executing the above commands, generate *tpch* files (1 GB) using the *dbgen* program. If needed, also update *the catalog*, *the dbfile output* and *tpch directories*, in the *myGtest.cc* file.

a) *Test for AddRel:* → ***TestAddRel***
This tests whether the relation is getting added correctly to the *Statistics* object and whether the number of tuples mentioned for that relation is also correctly mapped to the relation. *GetRelMap()* method of the

statistics object is called to get the mapping of the relation added. The number of tuples is compared with the retrieved value from that mapping.

b) *Test for CopyRel:* → ***TestCopyRel***
This tests whether the relation with old name is properly getting copied to the relation with new name. Dummy relation names and attribute names with number of distinct values for that attribute as well as the number of tuples are also stored and mapped in the *Statistics* object and *CopyRel* is called. Next, *GetRelMap()* is called to retrieve this mapped structure and all the retrieved values are compared with the dummy names and values.

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



```
g++ -O2 -Wno-deprecated -g -c Statistics.cc
g++ -O2 -Wno-deprecated -o gTest.out Record.o Comparison.o ComparisonEngine.o
[==========] Running 2 tests from 1 test case.
[----------] Global test environment set-up.
[----------] 2 tests from StatisticsGTests
[ RUN      ] StatisticsGTests.TestAddRel
[       OK ] StatisticsGTests.TestAddRel (0 ms)
[ RUN      ] StatisticsGTests.TestCopyRel
[       OK ] StatisticsGTests.TestCopyRel (0 ms)
[----------] 2 tests from StatisticsGTests (0 ms total)

[----------] Global test environment tear-down
[==========] 2 tests from 1 test case ran. (0 ms total)
[  PASSED  ] 2 tests.
```