Deep Bodra
5801 1841

REPORT

GROUP MEMBER NAMES

Individual Project

RUNNING THE CODE

Run Test (test.cc)

Make sure that *.bin.meta files are there for each *.bin file

```
make a4-1.out
./a4-1.out [0-11]

Or

To run all testcases
```

./runTestCases.sh

Run gtest (gtest_statistics.cc)

```
make gtest_statistics.out
./gtest_statistics.out
```

IMPLEMENTATION DETAILS

1. `Statistics (Statistics &copyMe);`
    a. It loops through all the relations in copyMe, for each relation it loops through all the attributes and copies data members from copyMe

2. `~Statistics()`
    a. It frees RelationInfo instances for each relation key in the statMap
    b. It also clears the relation key itself

3. `AddRel(char *relName, int numTuples)`
    a. It creates a new RelationInfo instance with numTuples and adds it to the statMap with relName as the key

4. `AddAtt(char *relName, char *attName, int numDistincts)`
    a. It finds the RelationInfo instance in statMap with key=relName
    b. It then adds the new attribute in the attributes map of this instance

5. `CopyRel(char *oldName, char *newName)`
    a. It finds the old relation in the statMap
    b. It creates a new key in the statMap with key as newRelationName.oldAttributeName
    c. It then copies all the required data from old relation into the new relation

6. `Read(char *fromWhere)`
    a. It loops through the statFile until “EOF” is encountered
    b. It then looks for “relation” keyword in the file which indicates the start of the new relation
    c. It reads the numTuples for this relation
    d. It then reads in the attributes line by line until “relation” or “EOF” is encountered when it will repeat the process again

7. `Write(char *fromWhere)`
    a. For each relation it writes out the “relation”, relation name, number of tuples and “attributes” on a separate line.
    b. It then loops through all the attributes and writes out attribute name and number of distinct tuples on a separate line
    c. The end of the file is marked as “EOF”

8. `Apply(struct AndList *parseTree, char *relNames[], int numToJoin)`
    a. Estimate is the function that will actually change the statMap when shouldApply member variable of Statistics class is set to true
    b. Apply will set this variable to true, call Estimate and set this variable back to false

9. `Estimate(struct AndList *parseTree, char **relNames, int numToJoin)`

ORMultiplier and ANDMultiplier

a. The overall cost is a function of the ORMultiplier and the ANDMultiplier which decide the fraction of the tuples in the output
b. It repeatedly traverses the left child of AND (which is OR) and then the right child (if any) which is again an AND
c. If both the operands of OR are of type NAME then the estimation is for join
d. In this case the ORMultiplier is given by `ORMultiplier *= (1 - (1/max(leftNumDistinct, rightNumDistinct)))`
e. If only the left operand is of type NAME then it is estimation for selection
f. If the current attribute is not dependent on the previous attribute then
    i. For GREATER_THAN or LESS_THAN `ORMultiplier *= (2/3)`
    ii. For EQUALS `ORMultiplier *= (1-(1 / number of distinct values for attributes in the join attribute of the left relation)`
g. If the current attribute is dependent on the previous attribute then
    i. For GREATER_THAN or LESS_THAN `ORMultiplier += (1/3)`
    ii. For EQUALS `ORMultiplier +=(1.0/number of distinct values for attributes in the join attribute of the left relation)`
h. Irrespective of the join or selection estimation, the ANDMultiplier is always given by `ANDMultiplier *= ORMultiplier`

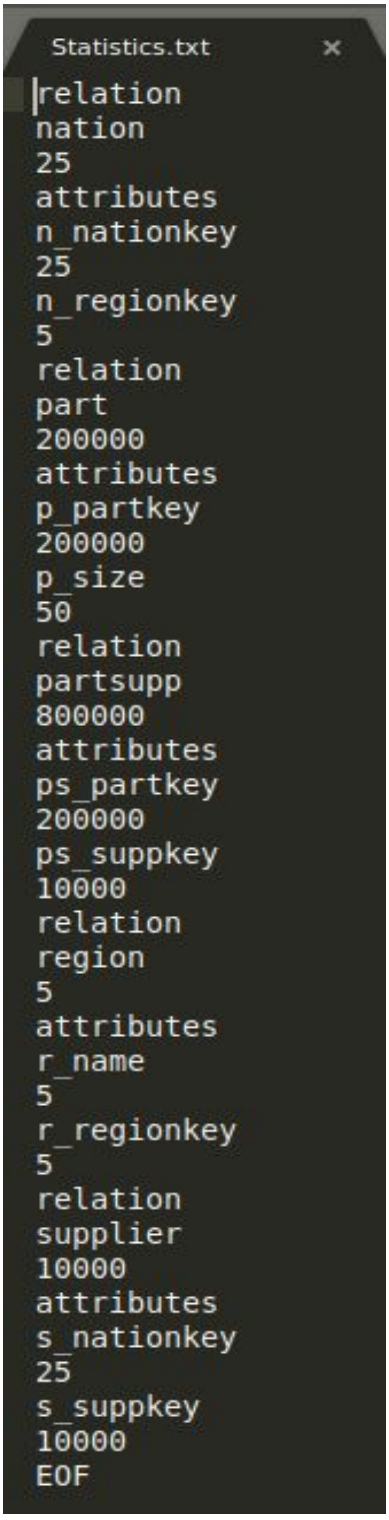
Cost Estimation

i. For Join `cost = number of tuples in left relation * number of tuples in right relation * ANDMultiplier`
j. For Select `cost = number of tuples in left relation * ANDMultiplier`

Apply Estimation

k. For Join
    i. We create a new relation with `key = leftRelation+"_"+rightRelation`
    ii. The attributes other than the join attributes are moved into this new relation
    iii. The original join relations are removed from the statMap
l. For Select
    i. There is no need to create a new relation. We only update current relation

FORMAT OF Statistics.txt



1. relation
    a. This keyword marks the end of the current relation and the beginning of the new relation
    b. The line after relation is the actual name of the relation
    c. The line after that is the number of tuples in the relation

2. attributes
    a. The attributes of this relation start after this keyword
    b. We then have <actual attribute name> followed by <distinct value counts for the attributes> in separate lines until again “relation” keyword is encountered

3. EOF
    a. This keyword indicates the end of the statistics file

OUTPUT

Command Line

```
stuxen@Omen:~/stuxen/Database-System-Implementation/P4.1: Statistics/src$ ./runTestCases.sh


Your estimation Result  857316
 Correct Answer: 8.5732e+5









stuxen@Omen:~/stuxen/Database-System-Implementation/P4.1: Statistics/src$
```

Statistics.txt

Statistics.txt | Makefile

```
1  relation
2  lineitem_part_lineitem_part
3  8
4  attributes
5  l_partkey
6  200000
7  l_shipinstruct
8  4
9  l_shipmode
10 7
11 p_container
12 40
13 p_partkey
14 200000
15 EOF
```

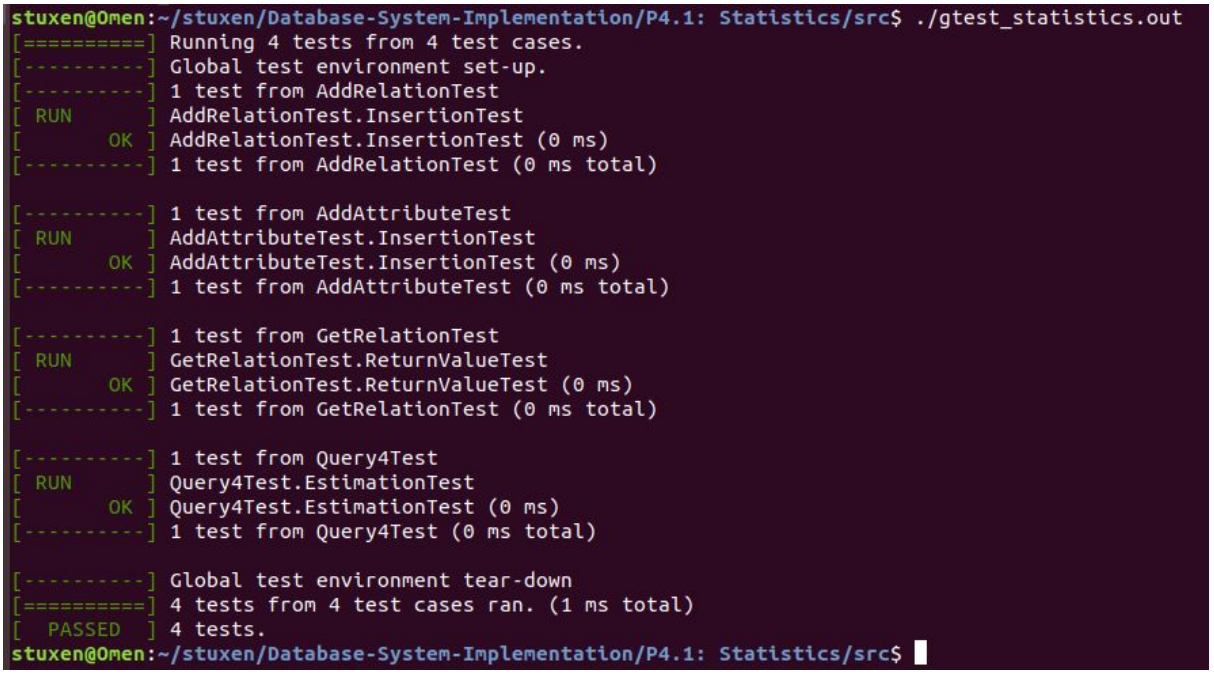
GTEST

1. AddRelationTest
    a. It adds a relation “nation” with 25 tuples and checks if the key exists and also verifies if the value stored is 25

2. AddAttributeTest
    a. It adds a relation “nation” with 25 tuples
    b. It adds the attribute “n_nationkey” with numDistincts = -1
    c. It then checks if the attribute exists in the map and if the value for numDistincts is stored as 25

3. GetRelationTest
    a. It adds the relations nation and region with attributes n_nationkey and r_regionkey respectively
    b. It uses GetRelation helper function (used in Estimate) to check if the it returns nation when n_nationkey is passed and region when r_regionkey is passed to it

4. Query4Test
    a. It tests the Estimate function for its estimated cost on query 4 in test.cc
    b. The difference between cost and 3200 (expected result) should be less than 0.1



CONCLUSION

We have taken a huge step towards designing a query optimizer