Deep Bodra
5801 1841

REPORT

GROUP MEMBER NAMES

Individual Project

RUNNING THE CODE

Run Test (main.cc)
Make sure that *.bin.meta files are there for each *.bin file
```
make main
./main
```

Note: output will be displayed in `cmd`

`Or`

To run all testcases [1-6]
./`runTestCases42.sh`

Note: output will be displayed in the file `output42.txt.` Also clear the file as the new output will be appended to the end of the file

Run gtest (gtest_statistics.cc)
```
make gtest_q_optimize.out
./gtest_q_optimize.out
```

IMPLEMENTATION DETAILS

1. `QueryPlan`
    a. The data structure of the QueryPlan is a representation of a query tree
    b. It is made up of different types of QueryNode's (Select, Project, Join, GroupBy,..etc)
    c. The parent of any subtree of this tree takes up the tuples from one or more of its children, processes it (based on its own node type) and sends it to its parent (if any)
    d. If a node has no parent then it just spits the output tuples in either the output file or the terminal/cmd

2. `QueryNode`
    a. They are the fundamental component of the QueryPlan/tree
    b. It is kind of an abstract class with very little functionality
    c. All the other types of nodes inherit this class
    d. It has information about the outputSchema, cost of the operation, statistics data, pipe id's, etc

3. `LeafQueryNode`
    a. They are the leaf nodes of the query plan
    b. They represent the Selection operator

4. `OnePipeQueryNode`
    a. They are the internal nodes which take one input pipe only
    b. They represent all the remaining operators (including WriteOut) except the Join
    c. Thus they can have only one child

5. `TwoPipeQueryNode`
    a. They are the internal nodes which take two input pipes
    b. It represents the Join operator
    c. Thus they have exactly two child

6. Operator Specific Node classes
    a. I have a dedicated class for each of the operators to be implemented
    b. They all inherit from either `OnePipeQueryNode` or `TwoPipeQueryNode`

7. Utils
    a. MoveSelectionDown
        i. It traverses the parse tree of the query and checks if the target schema is contained in the subtree (OR list of a node) in which case the SELECTION is moved down

b. EstimateJoinPermutationCost
    i. Given any permutation of the join, it uses Statistics to evaluate the cost of this sub plan of join
c. LoadMinCostJoin
    i. It permutes through all the possible joins using the permutation function of the algorithm library
    ii. For each permutation it uses EstimateJoinPermutationCost to get the cost for that permutation
    iii. After all the permutation have been evaluated it picks the best one

8. Combining Everything
    a. CreateLeafQueryNodes
        i. It loops through all the tables in the query and creates a `LeafQueryNode` for every table in the query

    b. CreateJoinQueryNodes
        i. It uses LoadMinCostJoin to get the best possible sub plan for a join
        ii. Depending on the best plan, it build the nodes accordingly and adds them to the QueryPlan
        iii. It also makes use of the MoveSelectionDown() function to push as many Selection operators down in the query as they are better plans than the ones in which selection is done at the top

    c. CreateSumQueryNodes
        i. It creates the required node and adds to the tree after initializing with the correct function information
        ii. It is also responsible for adding the GroupByQueryNode if there are any grouping attributes in the query

    d. CreateProjectQueryNodes
        i. It adds ProjectQueryNode if there are selection attributes in the query

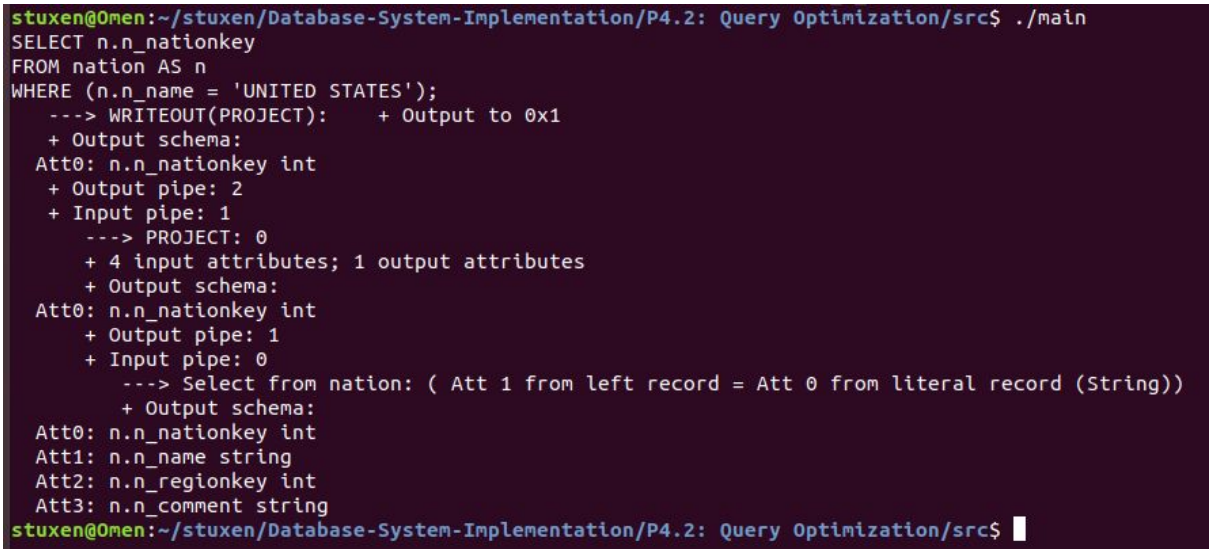
    e. CreateDistinctQueryNodes
        i. It adds DistinctQueryNode if the input query has Distinct

    f. CreateWriteOutQueryNodes
        i. The root of the query plan is always the WriteOutQueryNode
        ii. Generally the output of the Project operation is fed to this node

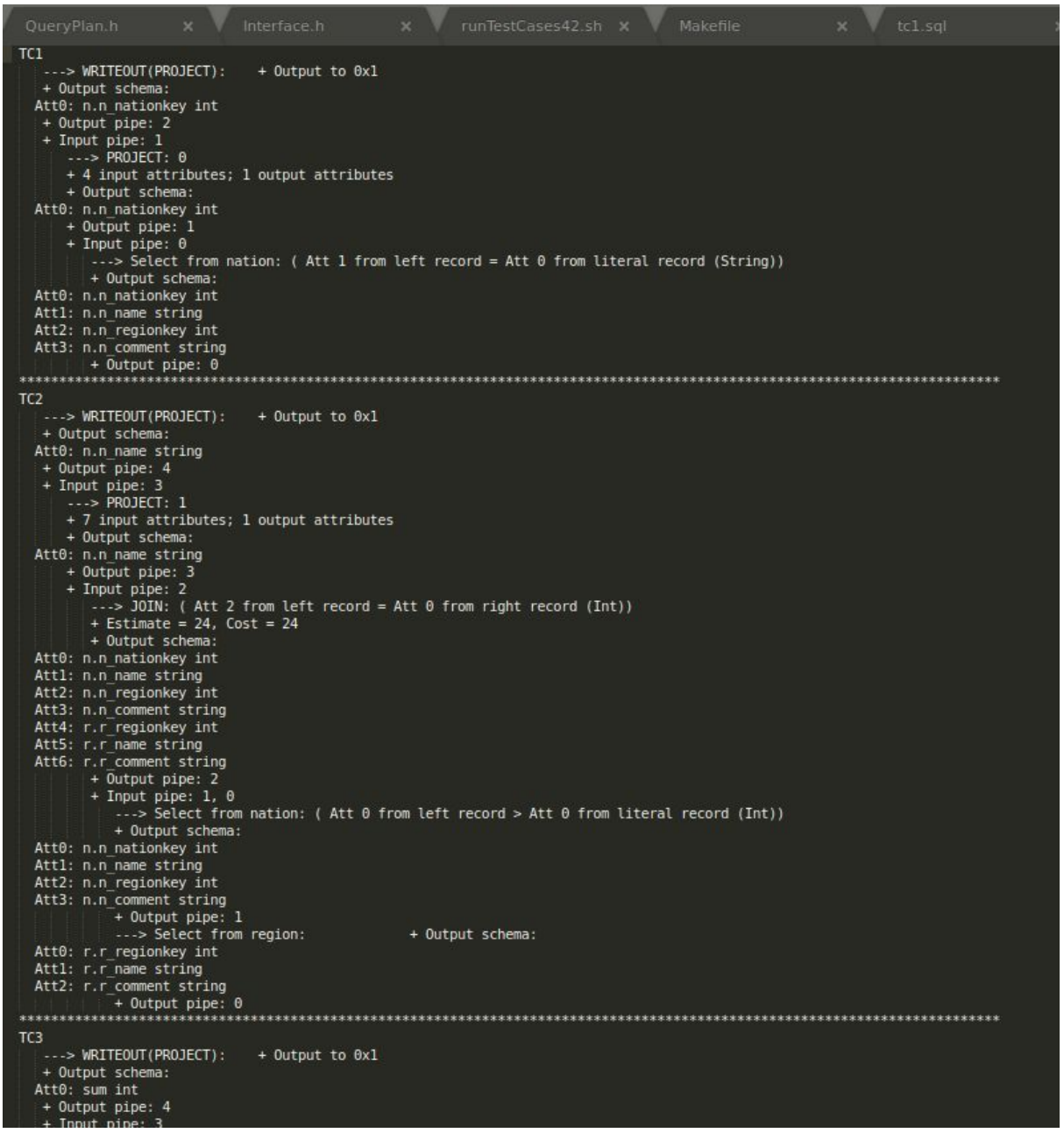
g. Print
    i. It prints the current node information and recursively prints the children
    ii. The format of the printing is same as the of the query tree
    iii. The indentation represents the children of the closest above parent

OUTPUT

Command Line (tc1.sql)



output42.txt



```
QueryPlan.h   Interface.h   runTestCases42.sh   Makefile   tc1.sql

TC1
   ---> WRITEOUT(PROJECT):    + Output to 0x1
   + Output schema:
  Att0: n.n_nationkey int
   + Output pipe: 2
   + Input pipe: 1
      ---> PROJECT: 0
      + 4 input attributes; 1 output attributes
      + Output schema:
  Att0: n.n_nationkey int
      + Output pipe: 1
      + Input pipe: 0
         ---> Select from nation: ( Att 1 from left record = Att 0 from literal record (String))
         + Output schema:
  Att0: n.n_nationkey int
  Att1: n.n_name string
  Att2: n.n_regionkey int
  Att3: n.n_comment string
         + Output pipe: 0
**********************************************************************************************************************************
TC2
   ---> WRITEOUT(PROJECT):    + Output to 0x1
   + Output schema:
  Att0: n.n_name string
   + Output pipe: 4
   + Input pipe: 3
      ---> PROJECT: 1
      + 7 input attributes; 1 output attributes
      + Output schema:
  Att0: n.n_name string
      + Output pipe: 3
      + Input pipe: 2
         ---> JOIN: ( Att 2 from left record = Att 0 from right record (Int))
         + Estimate = 24, Cost = 24
         + Output schema:
  Att0: n.n_nationkey int
  Att1: n.n_name string
  Att2: n.n_regionkey int
  Att3: n.n_comment string
  Att4: r.r_regionkey int
  Att5: r.r_name string
  Att6: r.r_comment string
         + Output pipe: 2
         + Input pipe: 1, 0
            ---> Select from nation: ( Att 0 from left record > Att 0 from literal record (Int))
            + Output schema:
  Att0: n.n_nationkey int
  Att1: n.n_name string
  Att2: n.n_regionkey int
  Att3: n.n_comment string
            + Output pipe: 1
            ---> Select from region:            + Output schema:
  Att0: r.r_regionkey int
  Att1: r.r_name string
  Att2: r.r_comment string
            + Output pipe: 0
**********************************************************************************************************************************
TC3
   ---> WRITEOUT(PROJECT):    + Output to 0x1
   + Output schema:
  Att0: sum int
   + Output pipe: 4
   + Input pipe: 3
```

(See the output42.txt for the rest of the output)

GTEST

1. OutputSchemaNumAttributeTest
    a. It verifies that the number of attributes in the outputSchema generated by the QueryPlan of the query tc6.sql is actually 2 or not
    b. Query6:

```
SELECT SUM (ps.ps_supplycost), s.s_suppkey
FROM part AS p, supplier AS s, partsupp AS ps
WHERE (p.p_partkey = ps.ps_partkey) AND (s.s_suppkey =
ps.ps_suppkey) AND (s.s_acctbal > 2500.0)
GROUP BY s.s_suppkey;
```

2. NumberOfTablesTest
    a. The bottom/leaf nodes of the query tree contain selection from a table
    b. The number of these selection nodes must be equal to the number of tables involved in the query
    c. This test check whether the QueryPlan actually creates the required number of nodes or not (works for any input query)

```
stuxen@Omen:~/stuxen/Database-System-Implementation/P4.2: Query Optimization/src$ ./gtest_q_optimize.out
[==========] Running 2 tests from 1 test case.
[----------] Global test environment set-up.
[----------] 2 tests from QueryPlanTest
[ RUN      ] QueryPlanTest.OutputSchemaNumAttributeTest
SELECT SUM (ps.ps_supplycost), s.s_suppkey
FROM part AS p, supplier AS s, partsupp AS ps
WHERE (p.p_partkey = ps.ps_partkey) AND (s.s_suppkey = ps.ps_suppkey) AND (s.s_acctbal > 2500.0)
GROUP BY s.s_suppkey;
[       OK ] QueryPlanTest.OutputSchemaNumAttributeTest (56141 ms)
[ RUN      ] QueryPlanTest.NumberOfTablesTest
SELECT SUM (ps.ps_supplycost), s.s_suppkey
FROM part AS p, supplier AS s, partsupp AS ps
WHERE (p.p_partkey = ps.ps_partkey) AND (s.s_suppkey = ps.ps_suppkey) AND (s.s_acctbal > 2500.0)
GROUP BY s.s_suppkey;
[       OK ] QueryPlanTest.NumberOfTablesTest (9876 ms)
[----------] 2 tests from QueryPlanTest (66017 ms total)

[----------] Global test environment tear-down
[==========] 2 tests from 1 test case ran. (66017 ms total)
[  PASSED  ] 2 tests.
stuxen@Omen:~/stuxen/Database-System-Implementation/P4.2: Query Optimization/src$
```

CONCLUSION

We have designed out query optimizer that will assist the Query Execution of Project 5 (Complete DB)