

COP-6726 DATABASE SYSTEM IMPLEMENTATION

Project - 4.2 Report

Group Member Names:

K Jaswanth Reddy - (UFID: 22719671)

M Dheeraj Kumar - (UFID: 82690922)

Running the Code:

1. Source code in the folder contains all the tbl and respective bin files for all tables.
2. Unit Testing is done and are presented in gtest_statistics.cc
3. To run a42.cc, from project root execute command "make a42" and then execute "./a42.out"
4. Google test framework has been added in the directory "googletest". This framework is needed to run gtest unit test cases.
5. The output is stored into output42.txt when runTestCases42.sh is triggered.
6. All the gtest unit test cases have been added in gtest_statistics.cc file. To run gtest_statistics.cc, from project root execute command "make gtestest_q_optimize.out" and then execute "./gtestest_q_optimize.out [0-6]"

Steps to run Tests(a42.cc):

- Execute the "make a42" command from the project root. Then execute "./a42.out"
- Enter the Transactional Queries into the console and it prints the necessary output.
- To run all the test cases at once we run by using ./runTestCases.sh [1-6]

Steps to run gtests(gtest_statistics.cc):

- Execute the "make gtest_q_optimize.out" command from the project root. Then execute "./gtest_q_optimize.out".

Methods Description:

QueryPlan:

1. QueryPlan is a data structure representing a query tree. It's made up of various QueryNode types (Select, Project, Join, GroupBy, etc.).
2. Any subtree's parent collects tuples from one or more of its children, processes them (using its own node type), and sends them to its parent (if any).
3. If a node has no parent, it just gives out the output tuples in either the output file or in the terminal.

QueryNode:

1. QueryNodes are the most important part of the QueryPlan/tree. It's a sort of abstract class with limited capabilities. This class is inherited by all other types of nodes.
2. It contains information about the outputSchema, the cost of the operation, statistical data, pipe ids, and so on.

LeafQueryNode:

1. LeafQueryNodes representing Selection operators are the leaf nodes of the query plan.

OnePipeQueryNode:

1. They represent all the remaining operators (including WriteOut) except the Join. They are internal nodes that only accept one input pipe. They can only have one child.

TwoPipeQueryNode:

1. They are internal nodes that accept two input pipes and represent the Join operator. As a result, they have exactly two child.

Operator Specific Node classes

1. We had also developed a class for each of the operators implemented in this project. All these inherit from either OnePipeQueryNode or TwoPipeQueryNode.

Utils:

1. Several functions such as MoveSelectionDown, EstimateJoinPermutationCost, LoadMinCostJoin functions are used to perform the basic operations in order to pick up the best possible options.

Also, several combining all of the functionalities are implemented using functions like: createLeafQueryNodes, CreateJoinQueryNodes, CreateSumQueryNodes, CreateProjectQueryNodes, CreateDistinctQueryNodes, CreateWriteOutQueryNodes, Print functions in order to perform the combining functionalities across the data to achieve the optimization of the transactional queries on the data.

Output:

Below are the Screenshots attached from results. Transactional queries are passed in individually to test out.

Testcase1(tc1):

```
jaswanth — ubuntu@primary: ~/Home/Desktop/P4_2_Query_Optimization/src — multipass
[ubuntu@primary:~/Home/Desktop/P4_2_Query_Optimization/src$ ./a42.out
SELECT n.n_nationkey
FROM nation AS n
[WHERE (n.n_name = 'UNITED STATES');
----> WRITEOUT(PROJECT):    + Output to 0x561b105f53c0
+ 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
ubuntu@primary:~/Home/Desktop/P4_2_Query_Optimization/src$ ]
```

Testcase2(tc2):

```
[ubuntu@primary:~/Home/Desktop/P4_2_Query_Optimization/src$ ./a42.out
SELECT n.n_name
FROM nation AS n, region AS r
[WHERE (n.n_regionkey = r.r_regionkey) AND (n.n_nationkey > 5);
----> WRITEOUT(PROJECT):    + Output to 0x55c89d0453c0
+ 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
ubuntu@primary:~/Home/Desktop/P4_2_Query_Optimization/src$ ]
```

Testcase3(tc3):

```
jaswanth — ubuntu@primary: ~/Home/Desktop/P4_2_Query_Optimization/src — multipass < mu
[ubuntu@primary:~/Home/Desktop/P4_2_Query_Optimization/src$ ./a42.out
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');
----> WRITEOUT(PROJECT):      + Output to 0x559c0287e3c0
+ Output schema:
Att0: sum int
+ Output pipe: 4
+ Input pipe: 3
----> SUM:      + Function:      + Output schema:
Att0: sum int
+ 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 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: 1
----> Select from region:      + Output schema:
Att0: r.r_regionkey int
Att1: r.r_name string
Att2: r.r_comment string
+ Output pipe: 0
ubuntu@primary:~/Home/Desktop/P4_2_Query_Optimization/src$
```

Testcase4(tc4):

```
jaswanth — ubuntu@primary: ~/Home/Desktop/P4_2_Query_Optimization/src — multipass < mu
[ubuntu@primary:~/Home/Desktop/P4_2_Query_Optimization/src$ ./a42.out
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;
----> WRITEOUT(PROJECT):      + Output to 0x55c6b65e23c0
+ Output schema:
Att0: sum int
Att1: n.n_regionkey int
+ Output pipe: 4
+ Input pipe: 3
----> GROUPBY:      + OrderMaker: NumAtts = 1
0: 2 Int
+ Function:      + Output schema:
Att0: sum int
Att1: n.n_regionkey int
+ 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 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: 1
----> Select from region:      + Output schema:
Att0: r.r_regionkey int
Att1: r.r_name string
Att2: r.r_comment string
+ Output pipe: 0
ubuntu@primary:~/Home/Desktop/P4_2_Query_Optimization/src$
```

Testcase5(tc5):

```

jaswanth — ubuntu@primary: ~/Home/Desktop/P4_2_Query_Optimization/src — multipass - multipa
[ubuntu@primary:~/Home/Desktop/P4_2_Query_Optimization/src$ ./a42.out
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;
----> WRITEOUT(PROJECT):      + Output to 0x56434ef6d3c0
+ Output schema:
Att0: sum int
Att1: r.r_regionkey int
+ Output pipe: 7
+ Input pipe: 6
----> GROUPBY:      + OrderMaker: NumAtts =      1
0:      4 Int
+ Function:      + Output schema:
Att0: sum int
Att1: r.r_regionkey int
+ Output pipe: 6
+ Input pipe: 5
----> DISTINCT:      + 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
Att7: c.c_custkey int
Att8: c.c_name string
Att9: c.c_address string
Att10: c.c_nationkey int
Att11: c.c_phone string
Att12: c.c_acctbal double
Att13: c.c_mktsegment string
Att14: c.c_comment string
+ Output pipe: 5
+ Input pipe: 4
----> JOIN: ( Att 0 from left record = Att 3 from right record (Int))
+ Estimate = 150000, Cost = 150024
+ 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
Att7: c.c_custkey int
Att8: c.c_name string
Att9: c.c_address string
Att10: c.c_nationkey int
Att11: c.c_phone string
Att12: c.c_acctbal double
Att13: c.c_mktsegment string
Att14: c.c_comment string
+ Output pipe: 4
+ Input pipe: 3, 0
----> 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: 3
+ Input pipe: 2, 1
----> 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: 3
+ Input pipe: 2, 1
----> 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: 2
----> Select from region:      + Output schema:
Att0: r.r_regionkey int
Att1: r.r_name string
Att2: r.r_comment string
+ Output pipe: 1
----> Select from customer:      + Output schema:
Att0: c.c_custkey int
Att1: c.c_name string
Att2: c.c_address string
Att3: c.c_nationkey int
Att4: c.c_phone string
Att5: c.c_acctbal double
Att6: c.c_mktsegment string
Att7: c.c_comment string
+ Output pipe: 0
ubuntu@primary:~/Home/Desktop/P4_2_Query_Optimization/src$

```

Testcase6(tc6):

```

jaswanth — ubuntu@primary: ~/Home/Desktop/P4_2_Query_Optimization/src — multipass • mu
[ubuntu@primary:~/Home/Desktop/P4_2_Query_Optimization/src$ ./a42.out
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;
----> WRITEOUT(PROJECT):      + Output to 0x55c97f94e3c0
+ Output schema:
Att0: sum double
Att1: s.s_suppkey int
+ Output pipe: 6
+ Input pipe: 5
----> GROUPBY:      + OrderMaker: NumAtts =      1
0:      14 Int
+ Function:      + Output schema:
Att0: sum double
Att1: s.s_suppkey int
+ Output pipe: 5
+ Input pipe: 4
----> JOIN: ( Att 0 from right record = Att 10 from left record (Int))
+ Estimate = 799999, Cost = 1599999
+ Output schema:
Att0: p.p_partkey int
Att1: p.p_name string
Att2: p.p_mfgr string
Att3: p.p_brand string
Att4: p.p_type string
Att5: p.p_size int
Att6: p.p_container string
Att7: p.p_retailprice double
Att8: p.p_comment string
Att9: ps.ps_partkey int
Att10: ps.ps_suppkey int
Att11: ps.ps_availqty int
Att12: ps.ps_supplycost double
Att13: ps.ps_comment string
Att14: s.s_suppkey int
Att15: s.s_name string
Att16: s.s_address string
Att17: s.s_nationkey int
Att18: s.s_phone string
Att19: s.s_acctbal double
Att20: s.s_comment string
+ Output pipe: 4
+ Input pipe: 3, 1
----> JOIN: ( Att 0 from left record = Att 0 from right record (Int))
+ Estimate = 800000, Cost = 800000
+ Output schema:
Att0: p.p_partkey int
Att1: p.p_name string
Att2: p.p_mfgr string
Att3: p.p_brand string
Att4: p.p_type string
Att5: p.p_size int
Att6: p.p_container string
Att7: p.p_retailprice double
Att8: p.p_comment string
Att9: ps.ps_partkey int
Att10: ps.ps_suppkey int
Att11: ps.ps_availqty int
Att12: ps.ps_supplycost double
Att13: ps.ps_comment string
+ Output pipe: 3
+ Input pipe: 2, 0
----> Select from part:      + Output schema:
Att0: p.p_partkey int
Att1: p.p_name string
Att2: p.p_mfgr string
Att3: p.p_brand string
Att4: p.p_type string
Att5: p.p_size int
Att6: p.p_container string
Att7: p.p_retailprice double
Att8: p.p_comment string
+ Output pipe: 2
----> Select from partsupp:      + Output schema:
Att0: ps.ps_partkey int
Att1: ps.ps_suppkey int
Att2: ps.ps_availqty int
Att3: ps.ps_supplycost double
Att4: ps.ps_comment string
+ Output pipe: 0
----> Select from supplier: ( Att 5 from left record > Att 0 from literal record (Double))
+ Output schema:
Att0: s.s_suppkey int
Att1: s.s_name string
Att2: s.s_address string
Att3: s.s_nationkey int
Att4: s.s_phone string
Att5: s.s_acctbal double
Att6: s.s_comment string
+ Output pipe: 1
[ubuntu@primary:~/Home/Desktop/P4_2_Query_Optimization/src$
ubuntu@primary:~/Home/Desktop/P4_2_Query_Optimization/src$

```

Output42.txt:

Below is the screenshot of the output generated by running the script runTestCases42.sh.

```
jaswanth — ubuntu@primary: ~/Home/Desktop/P4_2_Query_Optimization/src — multipass • multipass-gui.bYYYYX
ubuntu@primary:~/Home/Desktop/P4_2_Query_Optimization/src$ sh runTestCases42.sh
ubuntu@primary:~/Home/Desktop/P4_2_Query_Optimization/src$
ubuntu@primary:~/Home/Desktop/P4_2_Query_Optimization/src$
ubuntu@primary:~/Home/Desktop/P4_2_Query_Optimization/src$ cat output42.txt
TC1
----> WRITEOUT(PROJECT):      + Output to 0x55cbd64503c0
+ 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 0x55a7932bc3c0
+ 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 0x55be4158d3c0
+ Output schema:
Att0: sum int
+ Output pipe: 4
+ Input pipe: 3
----> SUM:      + Function:      + Output schema:
Att0: sum int
+ Output pipe: 3
+ Input pipe: 2
----> JOIN: ( Att 2 from left record = Att 0 from right record (Int))
+ Estimate = 24, Cost = 24
```

GTEST RESULTS:

1. OutputSchemaNumAttributeTest:

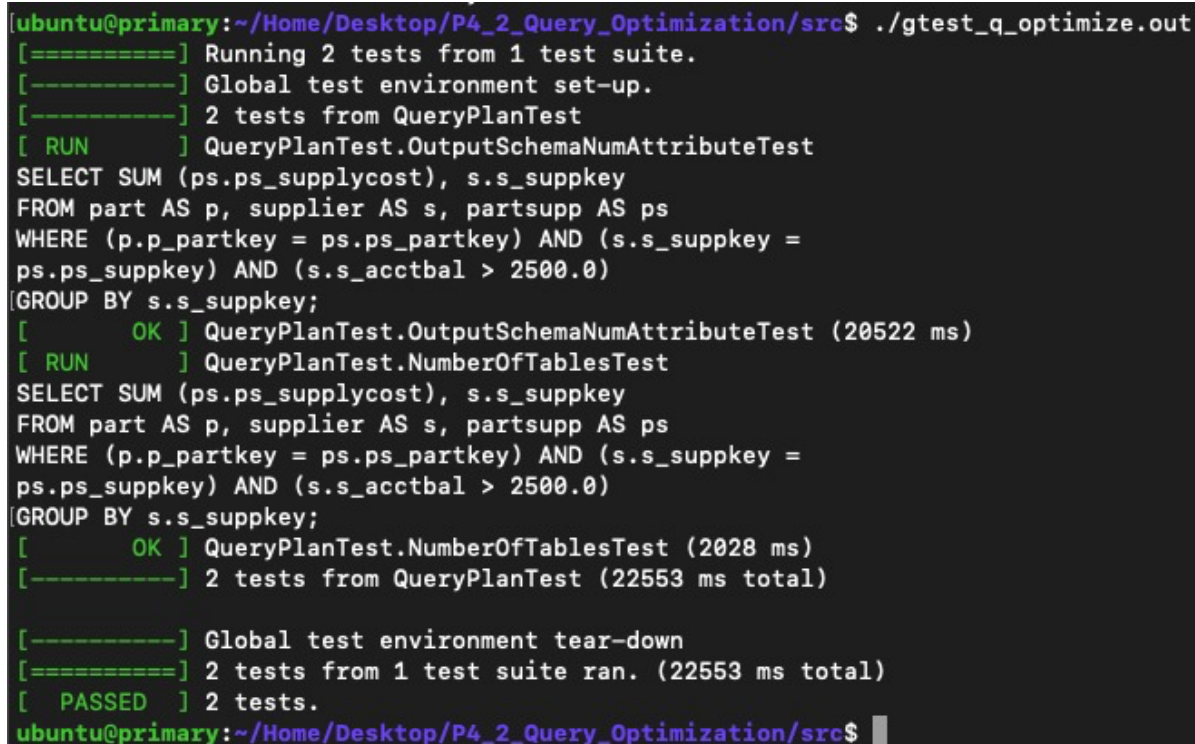
This test is used to verify the number of attributes in outputSchema generated by the QueryPlan for tcl6.sql and the query is

```
"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:

Leaf nodes of the query tree has the tables, and the number of these selections has to be equal to the number of tables for the query used. It is used to verify the Query Plan has created the number of nodes or not.

Execute the "make gtest_q_optimize.out" command from the project root. Then execute "./gtest_q_optimize.out".



```
[ubuntu@primary:~/Home/Desktop/P4_2_Query_Optimization/src$ ./gtest_q_optimize.out
[=====] Running 2 tests from 1 test suite.
[-----] 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 (20522 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 (2028 ms)
[-----] 2 tests from QueryPlanTest (22553 ms total)

[-----] Global test environment tear-down
[=====] 2 tests from 1 test suite ran. (22553 ms total)
[ PASSED  ] 2 tests.
ubuntu@primary:~/Home/Desktop/P4_2_Query_Optimization/src$
```

Result: All the test cases are Passed.

Conclusion: In this project we have successfully implemented a Query Optimizer used for the Query Execution.