

## Project 4 (part 2): Query Compilation and Optimization

### 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\_p42.zip*.
- b) Open the terminal.
- c) Navigate to the extracted folder.

#### II. To update “*test.cat*”

- a) In *test.cat* file, update: *catalog\_path*
- b) This should be updated in the first line.

#### III. To run the tests

Perform following these commands:

```
$ make clean
$ make a42.out
$ ./a42.out          ## Enter the Query, then push ctrl-D
$ ./a42.out< {fileName} ## The file feeds the Query to it.
$ ./runTestCases42.sh
```

Last command will generate the Query plan for files *tc1.sql*, *tc2.sql*, *tc3.sql*, *tc4.sql* and *tc5.sql*. These Query plans are saved in “*output42.txt*” file afterwards.

### Summarized Documentation of the implemented methods and functions:

We are required to print the *Optimized Query Plan* for a processed *SQL* query, that will also use *Statistics* implemented in the last assignment. The main files where the code is used, written and implemented are *ParseTree.h*, *PrintParseTree.h*, *PrintParseTree.cc* and *main.cc*

A *QueryPlan* is basically a *Tree* and it's *nodes* represent the *QueryPlanNode*. Also, as mentioned in the assignment, there shall be different types of these nodes, each subclass corresponding to the respective relational operators that were all implemented in *Assignment 3*. These nodes could be ‘*Select File (SF)*’, ‘*Select Pipe (SP)*’, ‘*Project (P)*’, ‘*Group By (GB)*’, ‘*Sum (S)*’, ‘*Distinct (D)*’, ‘*Join (J)*’ and ‘*Write (W)*’

The following ‘Flow Pipeline’ shows a brief implementation of the methods corresponding to different *QueryNodes* being called in the *Main* method in *main.cc*:

1. First, create a ‘Select File’ node for every relation or table calling the class ‘*SelectFileNode*’. Segregate Select & Join operations from the list given.
2. Then, create ‘Select Pipe’ node for every selected table using the class ‘*SelectPipeNode*’. As we want to minimize the number of generating intermediate tuples, compute the *Optimal Order of Joins*.
3. Create ‘Join Node’ for what is directed by the computed ‘Optimal Order of Joins’, using the class ‘*JoinNode*’.

4. Now, an order of Precedence follows for upcoming relational operations to be performed. That is,
  - a) If there is an attribute for 'Group By' → Use class '*GroupByNode*' to create 'Group By' node.
  - b) If there is any aggregate present in the attributes → Use class '*SumNode*' to create 'Sum' node. The aggregate function is nested in the *else* condition of 'Group By'.
  - c) If Query asked for 'Projection', Use class '*ProjectNode*' to create 'Project' node. Or use '*NewProjNode*'.
  - d) If Query asked for 'Distinct' tuples in the result, Use class '*DistinctNode*' to create 'Distinct' node.

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

### 1. TC1:

```
≡ output42.txt
1  TC1
2  enter:
3  Number of selects: 1
4  Number of joins: 0
5
6  IN ORDER TRAVERSAL
7  |
8  | *****
9  | SELECT FILE
10 | Input Pipe 0
11 | Output Pipe 1
12 | Output Schema:
13 |     Att n.n_nationkey : Int
14 |     Att n.n_name : String
15 |     Att n.n_regionkey : Int
16 |     Att n.n_comment : String
17 |
18 | *****
19 | SELECT PIPE
20 | Input Pipe 1
21 | Output Pipe 2
22 | Output Schema:
23 |     Att n.n_nationkey : Int
24 |     Att n.n_name : String
25 |     Att n.n_regionkey : Int
26 |     Att n.n_comment : String
27 | SELECTION CNF :
28 | ((n.n_name = UNITED STATES))
29 | *****
30 | PROJECT
31 | Input Pipe 2
32 | Output Pipe 3
33 | Output Schema:
34 |     Att n.n_nationkey : Int
35 | *****
```

## II. TC2:

≡ output42.txt

```
36 TC2
37 enter:
38 Number of selects: 1
39 Number of joins: 1
40
41 IN ORDER TRAVERSAL
42
43 | *****
44 | SELECT FILE
45 | Input Pipe 0
46 | Output Pipe 1
47 | Output Schema:
48 |     Att n.n_nationkey : Int
49 |     Att n.n_name : String
50 |     Att n.n_regionkey : Int
51 |     Att n.n_comment : String
52 |
53 | *****
54 | SELECT PIPE
55 | Input Pipe 1
56 | Output Pipe 2
57 | Output Schema:
58 |     Att n.n_nationkey : Int
59 |     Att n.n_name : String
60 |     Att n.n_regionkey : Int
61 |     Att n.n_comment : String
62 | SELECTION CNF :
63 | ((n.n_nationkey > 5))
64 | *****
65 | SELECT FILE
66 | Input Pipe 0
67 | Output Pipe 3
68 | Output Schema:
69 |     Att r.r_regionkey : Int
70 |     Att r.r_name : String
71 |     Att r.r_comment : String
72 |
73 | *****
```

```
73 | *****
74 | JOIN
75 | Left Input Pipe 2
76 | Right Input Pipe 3
77 | Output Pipe 4
78 | Output Schema:
79 |     Att n.n_nationkey : Int
80 |     Att n.n_name : String
81 |     Att n.n_regionkey : Int
82 |     Att n.n_comment : String
83 |     Att r.r_regionkey : Int
84 |     Att r.r_name : String
85 |     Att r.r_comment : String
86 | CNF:
87 | (n.n_regionkey = r.r_regionkey)
88 |
89 | *****
90 | PROJECT
91 | Input Pipe 4
92 | Output Pipe 5
93 | Output Schema:
94 |     Att n.n_name : String
95 | *****
```

### III. TC3:

```
output42.txt
96 TC3
97 enter:
98 Number of selects: 1
99 Number of joins: 1
100
101 IN ORDER TRAVERSAL
102
103 *****
104 SELECT FILE
105 Input Pipe 0
106 Output Pipe 1
107 Output Schema:
108     Att n.n_nationkey : Int
109     Att n.n_name : String
110     Att n.n_regionkey : Int
111     Att n.n_comment : String
112
113 *****
114 SELECT PIPE
115 Input Pipe 1
116 Output Pipe 2
117 Output Schema:
118     Att n.n_nationkey : Int
119     Att n.n_name : String
120     Att n.n_regionkey : Int
121     Att n.n_comment : String
122 SELECTION CNF :
123 ((n.n_name = UNITED STATES))
124 *****
125 SELECT FILE
126 Input Pipe 0
127 Output Pipe 3
128 Output Schema:
129     Att r.r_regionkey : Int
130     Att r.r_name : String
131     Att r.r_comment : String
132
133 *****
```

```
132
133 *****
134 JOIN
135 Left Input Pipe 2
136 Right Input Pipe 3
137 Output Pipe 4
138 Output Schema:
139     Att n.n_nationkey : Int
140     Att n.n_name : String
141     Att n.n_regionkey : Int
142     Att n.n_comment : String
143     Att r.r_regionkey : Int
144     Att r.r_name : String
145     Att r.r_comment : String
146 CNF:
147 (n.n_regionkey = r.r_regionkey)
148
149 *****
150 SUM
151 Input Pipe ID : 4
152 Output Pipe ID : 5
153 Output Schema:
154     Att sum : Double
155 Function :
156 (n.n_nationkey)
157 distinctFunc: 0
158 *****
```



#### IV. TC4:

```
≡ output42.txt
159 TC4
160 enter:
161 Number of selects: 1
162 Number of joins: 1
163 GROUPING ON n.n_regionkey
164 | Att n.n_regionkey: String
165 IN ORDER TRAVERSAL
166
167 *****
168 SELECT FILE
169 Input Pipe 0
170 Output Pipe 1
171 Output Schema:
172 | Att n.n_nationkey : Int
173 | Att n.n_name : String
174 | Att n.n_regionkey : Int
175 | Att n.n_comment : String
176
177 *****
178 SELECT PIPE
179 Input Pipe 1
180 Output Pipe 2
181 Output Schema:
182 | Att n.n_nationkey : Int
183 | Att n.n_name : String
184 | Att n.n_regionkey : Int
185 | Att n.n_comment : String
186 SELECTION CNF :
187 | ((n.n_name = UNITED STATES))
188 *****
189 SELECT FILE
190 Input Pipe 0
191 Output Pipe 3
192 Output Schema:
193 | Att r.r_regionkey : Int
194 | Att r.r_name : String
195 | Att r.r_comment : String
196
```

```
≡ output42.txt
196
197 *****
198 JOIN
199 Left Input Pipe 2
200 Right Input Pipe 3
201 Output Pipe 4
202 Output Schema:
203 | Att n.n_nationkey : Int
204 | Att n.n_name : String
205 | Att n.n_regionkey : Int
206 | Att n.n_comment : String
207 | Att r.r_regionkey : Int
208 | Att r.r_name : String
209 | Att r.r_comment : String
210 CNF:
211 | (n.n_regionkey = r.r_regionkey)
212
213 *****
214 GROUP BY
215 Left Input Pipe 4
216 Output Pipe 5
217 Output Schema:
218 | Att sum : Double
219 | Att n.n_regionkey : Int
220 OrderMaker :
221 NumAtts = 1
222 | 0: 2 Int
223 Function :
224 | (n.n_regionkey)
225 distinctFunc: 0
226
227 *****
228 PROJECT
229 Input Pipe 5
230 Output Pipe 6
231 Output Schema:
232 | Att sum : Double
233 *****
```

## V. TC5:

≡ output42.txt

```
234 TC5
235 enter:
236 Number of selects: 1
237 Number of joins: 2
238 GROUPING ON r.r_regionkey
239 | Att r.r_regionkey: String
240 IN ORDER TRAVERSAL
241
242 *****
243 SELECT FILE
244 Input Pipe 0
245 Output Pipe 1
246 Output Schema:
247 | Att n.n_nationkey : Int
248 | Att n.n_name : String
249 | Att n.n_regionkey : Int
250 | Att n.n_comment : String
251
252 *****
253 SELECT PIPE
254 Input Pipe 1
255 Output Pipe 2
256 Output Schema:
257 | Att n.n_nationkey : Int
258 | Att n.n_name : String
259 | Att n.n_regionkey : Int
260 | Att n.n_comment : String
261 SELECTION CNF :
262 | ((n.n_nationkey > 10))
263 | *****
264 SELECT FILE
265 Input Pipe 0
266 Output Pipe 3
267 Output Schema:
268 | Att r.r_regionkey : Int
269 | Att r.r_name : String
270 | Att r.r_comment : String
271
```

≡ output42.txt

```
272 | *****
273 JOIN
274 Left Input Pipe 2
275 Right Input Pipe 3
276 Output Pipe 4
277 Output Schema:
278 | Att n.n_nationkey : Int
279 | Att n.n_name : String
280 | Att n.n_regionkey : Int
281 | Att n.n_comment : String
282 | Att r.r_regionkey : Int
283 | Att r.r_name : String
284 | Att r.r_comment : String
285 CNF:
286 | (n.n_regionkey = r.r_regionkey)
287
288 | *****
289 SELECT FILE
290 Input Pipe 0
291 Output Pipe 5
292 Output Schema:
293 | Att c.c_custkey : Int
294 | Att c.c_name : String
295 | Att c.c_address : String
296 | Att c.c_nationkey : Int
297 | Att c.c_phone : String
298 | Att c.c_acctbal : Double
299 | Att c.c_mktsegment : String
300 | Att c.c_comment : String
301
302 | *****
303 JOIN
304 Left Input Pipe 4
305 Right Input Pipe 5
306 Output Pipe 6
307 Output Schema:
308 | Att n.n_nationkey : Int
309 | Att n.n_name : String
```

≡ output42.txt

```
310     Att n.n_regionkey : Int
311     Att n.n_comment : String
312     Att r.r_regionkey : Int
313     Att r.r_name : String
314     Att r.r_comment : String
315     Att c.c_custkey : Int
316     Att c.c_name : String
317     Att c.c_address : String
318     Att c.c_nationkey : Int
319     Att c.c_phone : String
320     Att c.c_acctbal : Double
321     Att c.c_mktsegment : String
322     Att c.c_comment : String
323 CNF:
324 (n.n_nationkey = c.c_nationkey)
325
326     *****
327 GROUP BY
328 Left Input Pipe 6
329 Output Pipe 7
330 Output Schema:
331     Att sum : Double
332     Att r.r_regionkey : Int
333 OrderMaker :
334 NumAtts =      1
335 0:      4 Int
336 Function :
337 ((n.n_nationkey) + (r.r_regionkey))
338 distinctFunc: 1
339
340     *****
341 PROJECT
342 Input Pipe 7
343 Output Pipe 8
344 Output Schema:
345     Att sum : Double
346     *****
```



## GTests:

- Perform following commands to execute the GTests:

```
$ make clean
$ make gTest.out
$ ./gTest.out
```

a) *QueryOptimizationTest for Join Node*:→ **JoinNode**

This tests whether the Join Order is generated if 'And' is passed. Boolean expression handles this in the structure. The result is compared with the expected outcome.

b) *QueryOptimizationTest for Project Node*:→ **ProjectNode**

This tests whether if the Projection is required, the node is generated in Query Plan or not. This is done by passing the 'Comparison Pointed' in the structure. The result is compared with the expected outcome.

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

```
g++ -O2 -Wno-deprecated -g -c gTest.cc
g++ -O2 -Wno-deprecated -o gtest.out Record.o Comparison.o ComparisonEngine.o
.o gtest.o -l pthread -lgtest
[=====] Running 2 tests from 1 test case.
[-----] Global test environment set-up.
[-----] 2 tests from QueryOptimizationTest
[ RUN      ] QueryOptimizationTest.JoinNode
([ OK ] QueryOptimizationTest.JoinNode (0 ms)
[ RUN      ] QueryOptimizationTest.ProjectNode
unknown code 97184248[ OK ] QueryOptimizationTest.ProjectNode (0 ms)
[-----] 2 tests from QueryOptimizationTest (0 ms total)

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