Team24_Assignment_3
106034061 曾靖渝
106062116 黃晨
106062216 馮謙

Query flow:

由於本次作業是用 **consoleSQLinterpreter** 作為 client 進行測試，因此我們從這個 class 開始 trace code。發現在 **doQuery()** 中，會經由 **RemoteStatementImpl** 產生一個 Planner, 經過 lexer 切割字串、parser 檢查 syntax, verifier 檢查 semantic，再由 **BasicQueryPlanner** create plan tree，回傳給 RemoteStatementImpl 後 call plan.open()，將 resultset 回傳給 client 端。

Implementation:

Lexer

因為需要核對keyword，須在**initkeyword()**中新增”explain“ datatype。

Parser

在**queryCommand()**中最前面新增判斷是否match “explain”的條件，並回傳結果” isExplained“給QueryData。

QueryData

新增”isExplained“參數紀錄是否讀到explain指令，並新增function“**isExplained()”**回傳結果。

BasicQueryPlanner

在Planner tree的最上層新增ExplainPlan，並把之前串接好的樹傳入ExplainPlan

Plan

在 plan interface 中定義新 method **String explain(int tab_cnt)**，並在各種 plan class 中 implement。這個 method 會先 call 下層 plan 的同一個method，得到下層 plan 的 explain message 之後，再與自己的 message concatenate。Estimated blocks, records 則是由 **blocksAccessed()**，**recordsOutput()** 這兩個method 得到。參數 tab_cnt 的作用是為了排版計算 tab 的數量。

ExplainPlan

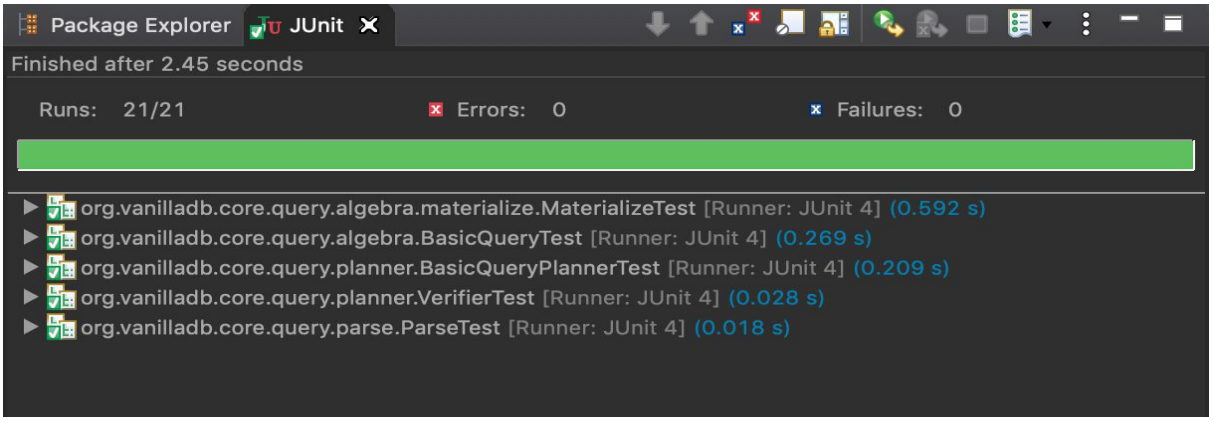
新增 ExplainPlan 來處理”explain” 指令。Construct 時，會傳入下一層的 plan，並產生一個 schema，裡面只有一個 field “query-plan”， type = VarChar(500)。**open()** 會先 call 下一層的 **open()** 得到 scan，再 call **explain()** 得到 explain message，用這兩個 object 以及 schema 作為參數 construct ExplainScan。其他 method 因為不會用到，因此我們只有基本的定義而已。

ExplainScan

為了讓 client 使用起來像是只有一個 record，我們新增一個變數 step，紀錄 client call **beforefirst()** 之後，call 了幾次 **next()** 。Client call **beforefirst()** 時，會先把 step 歸零。再 deligate 給下層的 scan.beforefirst()。接著 call **next()**，會先判斷 step 是否等於 0。如果是，則計算下層的 scan 有多少個 record，並將這個訊息加到 explain message 的後面，並將 step increase。如果 step 不是 0，則直接 return false，避免 client 印出多次 explain message。當 client call **getval()** 時，先用 **isField()** 確認 field name 是否為 "query-plan"，並將 explain message 回傳。

Experiments

1.指令的正確性：如圖所見，修改後的sql指令跑TPCC依然可以正確執行



2.A query accessing single table with WHERE：由圖可見，可正確執行

```
EXPLAIN SELECT d_id FROM district WHERE d_id > 5

query-plan
-------------------------------------------------------------------
->ProjectPlan (#blks=2, #recs=5)
        ->SelectPlan:(d_id>5.0) (#blks=2, #recs=0)
                ->TablePlan on (district) (#blks=2, #recs=10)

Actual #recs: 5
```

3.A query accessing multiple tables with WHERE：可正確執行

```
EXPLAIN SELECT COUNT(d_id) FROM district, warehouse WHERE d_w_id = w_id

query-plan
-------------------------------------------------------------------------
->ProjectPlan (#blks=22, #recs=1)
        ->GroupByPlan (#blks=22, #recs=1)
                ->SelectPlan:(d_w_id=w_id) (#blks=22, #recs=10)
                        ->ProductPlan (#blks=22, #recs=10)
                                ->TablePlan on (district) (#blks=2, #recs=10)
                                ->TablePlan on (warehouse) (#blks=2, #recs=1)

Actual #recs: 1
```

4.A query with ORDER BY：由圖可見，可正確執行

```
SQL> EXPLAIN SELECT COUNT(d_id) FROM district, warehouse WHERE d_w_id = w_id GROUP BY w_id

query-plan
-----------------------------------------------------------------------------------
->ProjectPlan (#blks=2, #recs=1)
       ->GroupByPlan (#blks=2, #recs=1)
               ->SortPlan (#blks=2, #recs=10)
                       ->SelectPlan:(d_w_id=w_id) (#blks=22, #recs=10)
                               ->ProductPlan (#blks=22, #recs=10)
                                       ->TablePlan on (district) (#blks=2, #recs=10)
                                       ->TablePlan on (warehouse) (#blks=2, #recs=1)

Actual #recs: 1
```

5.A query with GROUP BY and at least one aggregation function (MIN, MAX, COUNT, AVG... etc.)

```
EXPLAIN SELECT d_name FROM district, warehouse WHERE d_w_id = w_id ORDER BY d_name DESC

query-plan
-------------------------------------------------------------------------------------------
->SortPlan (#blks=1, #recs=10)
        ->ProjectPlan (#blks=22, #recs=10)
                ->SelectPlan:(d_w_id=w_id) (#blks=22, #recs=10)
                        ->ProductPlan (#blks=22, #recs=10)
                                ->TablePlan on (district) (#blks=2, #recs=10)
                                ->TablePlan on (warehouse) (#blks=2, #recs=1)

Actual #recs: 10
```

6. 當我們想把所有指令合成單一query時：發生錯誤（於Appendix已經 fix bug）

```
SQL> EXPLAIN SELECT COUNT(d_id) FROM district, warehouse WHERE d_w_id = w_id GROUP BY w_id order by count(d_id)

SQL Exception: org.vanilladb.core.query.planner.BadSemanticException: field countofd_id does not exist
java.sql.SQLException: org.vanilladb.core.query.planner.BadSemanticException: field countofd_id does not exist
        at org.vanilladb.core.remote.jdbc.JdbcStatement.executeQuery(JdbcStatement.java:37)
        at org.vanilladb.core.util.ConsoleSQLInterpreter.doQuery(ConsoleSQLInterpreter.java:73)
        at org.vanilladb.core.util.ConsoleSQLInterpreter.main(ConsoleSQLInterpreter.java:54)
```

Appendix

在執行 order by aggregation function 的時候，我們發現系統會跳出 _____ does not exist 的錯誤，於是我們決定做個實驗來看哪裡出了問題。我們把相同的 table 和指令跑在 postgreSQL 上，然後神奇的事情發生了：

這是我們在 core 上建的 table：

```
SQL> select pid, pname, cname, sname from professors

      cname   pid     pname     sname
-------------------------------------
       Algo    21   pGeorge    George
       Algo    22   pGeorge      Mark
         CA    21     pMark       Roy
         CA    21     pMark    George
```

其對應到的指令及結果：

```
SQL> select avg(pid) from professors where pid >= 21 group by cname order by avg(pid)

SQL Exception: org.vanilladb.core.query.planner.BadSemanticException: field avgofpid does not exist
java.sql.SQLException: org.vanilladb.core.query.planner.BadSemanticException: field avgofpid does not exist
        at org.vanilladb.core.remote.jdbc.JdbcStatement.executeQuery(JdbcStatement.java:37)
        at org.vanilladb.core.util.ConsoleSQLInterpreter.doQuery(ConsoleSQLInterpreter.java:73)
        at org.vanilladb.core.util.ConsoleSQLInterpreter.main(ConsoleSQLInterpreter.java:54)
Caused by: org.vanilladb.core.query.planner.BadSemanticException: field avgofpid does not exist
        at org.vanilladb.core.query.planner.Verifier.verifyQueryData(Verifier.java:104)
```

這是 PostgreSQL 上建的 table：

```
postgres=# select pid, pname, cname, sname from professor
postgres-# ;
 pid |  pname   | cname | sname
-----+---------+-------+--------
  21 | pGeorge | Algo  | George
  21 | pGeorge | Algo  | Mark
  22 | pMark   | CA    | Mark
  22 | pMark   | CA    | Roy
(4 rows)
```

這是對應的指令及結果：

```
postgres=# select avg(pid) from professor where pid >= 21 group by cname order by avg(pid)
postgres-# ;
         avg
--------------------
 21.0000000000000000
 22.0000000000000000
(2 rows)
```

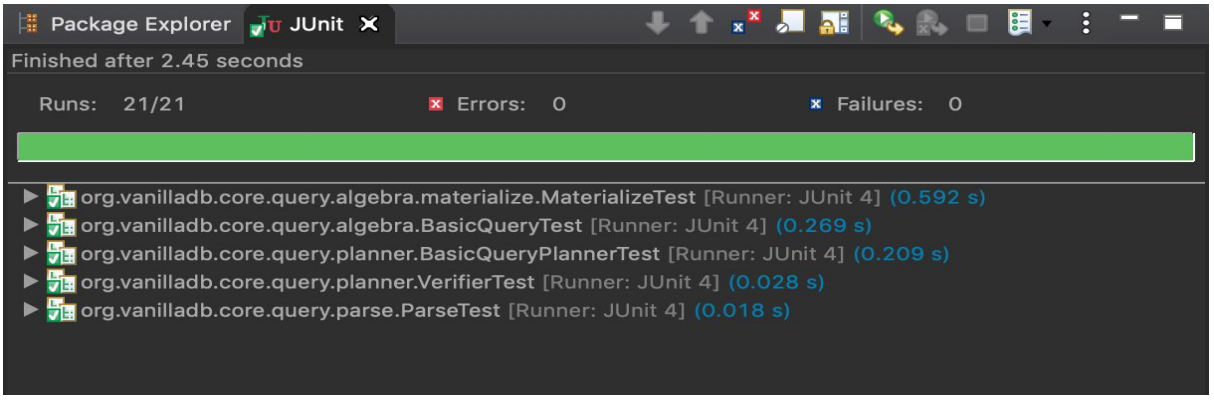
結果顯示，相同的指令在 core 上會有報錯的情形，於是我們把問題發到了討論區上。經過助教的熱心協助後，發現原因出在 sorting verifier 上，於是我們把 sorting field 做了一些改動，讓它跟 projecting field 做到相同的事情。

```
99
100         // Examine the sorting field name
101 //      if (data.sortFields() != null)
102 //          for (String sortFld : data.sortFields()) {
103 //              if (!verifyField(schs, views, sortFld))
104 //                  throw new BadSemanticException("field " + sortFld
105 //                          + " does not exist");
106 //          }
107         if (data.sortFields() != null)
108             for (String sortFld : data.sortFields()) {
109                 boolean isValid = verifyField(schs, views, sortFld);
110                 if (!isValid && data.aggregationFn() != null)
111                     for (AggregationFn aggFn : data.aggregationFn())
112                         if (sortFld.compareTo(aggFn.fieldName()) == 0) {
113                             isValid = true;
114                             break;
115                         }
116                 if (!isValid)
117                     throw new BadSemanticException("field " + sortFld
118                             + " does not exist");
119             }
120     }
```

然後做個小測試看結果是否正確：

```
SQL> EXPLAIN SELECT COUNT(d_id) FROM district, warehouse WHERE d_w_id = w_id GROUP BY w_id order by COUNT(d_id)

query-plan
------------------------------------------------------------------------------------------------------------------------
->SortPlan (#blks=1, #recs=1)
        ->ProjectPlan (#blks=2, #recs=1)
                ->GroupByPlan (#blks=2, #recs=1)
                        ->SortPlan (#blks=2, #recs=10)
                                ->SelectPlan:(d_w_id=w_id) (#blks=22, #recs=10)
                                        ->ProductPlan (#blks=22, #recs=10)
                                                ->TablePlan on (district) (#blks=2, #recs=10)
                                                ->TablePlan on (warehouse) (#blks=2, #recs=1)

Actual #recs: 1

SQL> SELECT COUNT(d_id) FROM district, warehouse WHERE d_w_id = w_id GROUP BY w_id order by COUNT(d_id)

countofd_id
------
    10

SQL>
```

最後在做個 unit test 以防 core-patch 被我們搞壞：



以上就是 sorting field 的奇妙冒險。