Database Assignment3 report

104062203 陳涵宇
104062232 廖子毅
104062234 林士軒

How to implement?

org.vanilladb.core.query.parse

- QueryData
  - Add boolean isExplain (which is mentioned in the comments by TA)
  - Alter the Function *toString()*, which can append "explain" now.
- Lexer
  - Add "explain" into keyword in order to check whether the keywords "exists" or "eat"
- Parser
  - We add a variable, which is a boolean called "isExplained". By the above, we can use the variable to check the whether the world "EXPLAIN" is exists. And if so, we will return true to the QueryData.

org.vanilladb.core.query.planner

- BasicQueryPlanner
  - We mimic the above code and comments. Create the step 7 in this function. And it will create a new ExplainPlain when isExplain is true.

org.vanilladb.core.query.algebra

We construct two classes, which are named as *ExplainPlan* and *ExplainScan*. Besides, We modified the Plan interface and *Plan class.

- Plan Interface
  - Add a function *toString()* to output the info of each plan.
- ExplainPlan
  - Inherit the interface Plan.
  - Modify the following function:

    > 1. Constructor : Pass the previous Plan.
    > 2. Scan : Open previous scan and pass the output data info and schema.
    > 3. schema : new a empty schema and add a field "query-plan VARCHAR(500)"
    > 4. toString : We do not do anything here because this plan will do nothing but output the previous plan info.

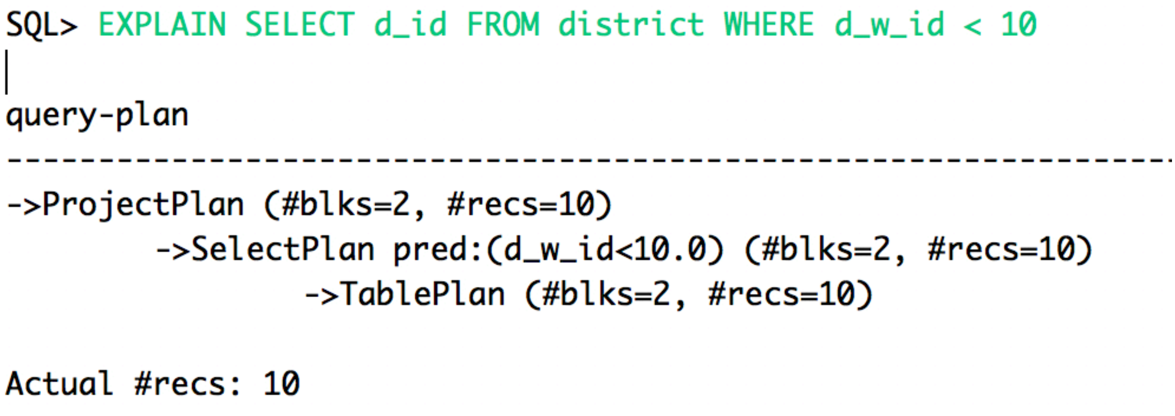
- ExplainScan
  - Modify the following function:

    > 1. Constructor : save the output data info and add one line for total #rec counted by s.next().
    > 2. beforeFirst : because there will be only one data in the field so we use a boolean isFirst to represent it.
    > 3. next : to check whether the boolean isFirst is true or not.
    > 4. getVal : if the field name is "query-plan" return the output data info. Otherwise, throw RuntimeException of not found field name.

- *Plan class
  - Add *toString()* function to output the plan info for explain query.

Use our as3-bench project to load the TPC-C testbed and show the EXPLAIN result for the following queries:

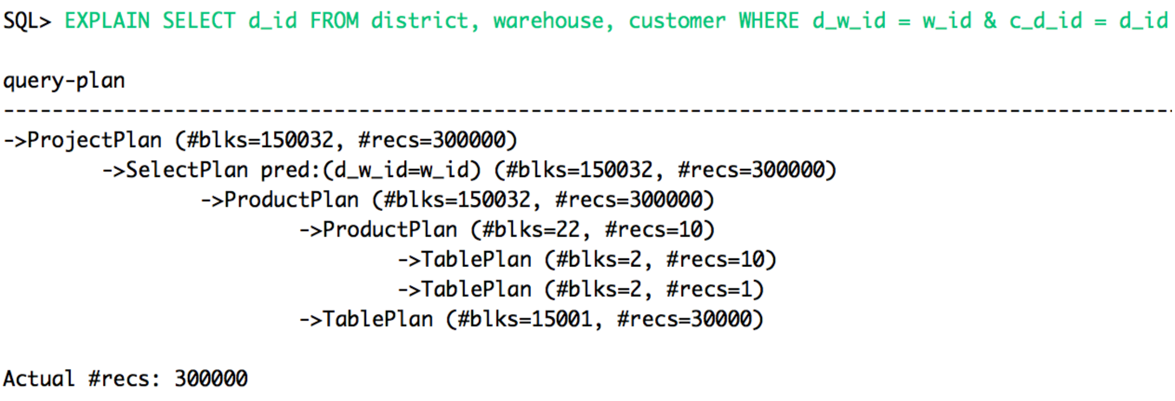
1. A query accessing single table with WHERE



```
SQL> EXPLAIN SELECT d_id FROM district WHERE d_w_id < 10
|
query-plan
------------------------------------------------------------------
->ProjectPlan (#blks=2, #recs=10)
        ->SelectPlan pred:(d_w_id<10.0) (#blks=2, #recs=10)
                ->TablePlan (#blks=2, #recs=10)

Actual #recs: 10
```

2. A query accessing multiple tables with WHERE

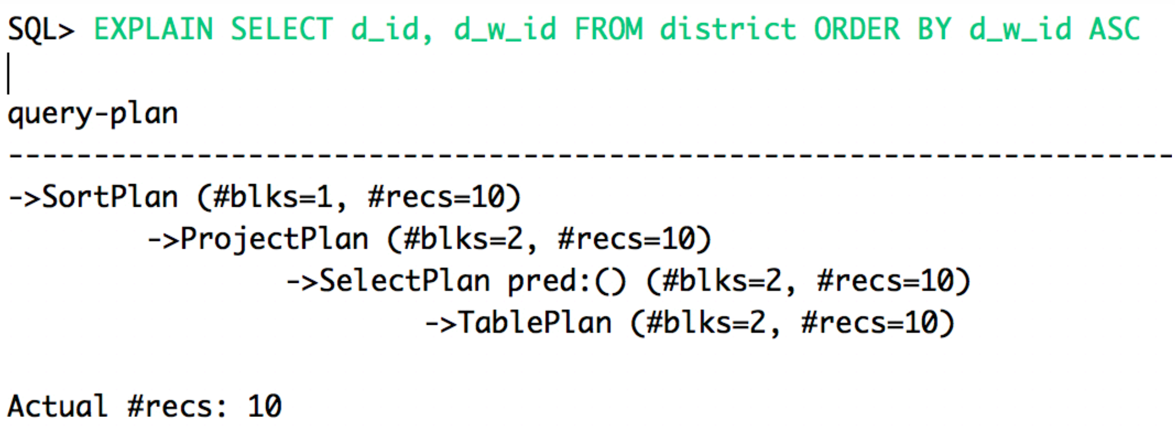


```
SQL> EXPLAIN SELECT d_id FROM district, warehouse, customer WHERE d_w_id = w_id & c_d_id = d_id

query-plan
-----------------------------------------------------------------------------------------------
->ProjectPlan (#blks=150032, #recs=300000)
        ->SelectPlan pred:(d_w_id=w_id) (#blks=150032, #recs=300000)
                ->ProductPlan (#blks=150032, #recs=300000)
                        ->ProductPlan (#blks=22, #recs=10)
                                ->TablePlan (#blks=2, #recs=10)
                                ->TablePlan (#blks=2, #recs=1)
                        ->TablePlan (#blks=15001, #recs=30000)

Actual #recs: 300000
```

3. A query with ORDER BY

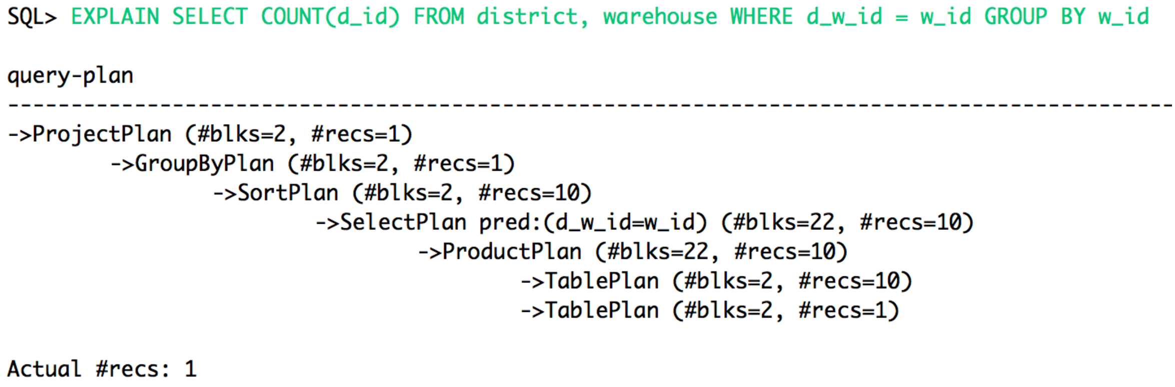


```
SQL> EXPLAIN SELECT d_id, d_w_id FROM district ORDER BY d_w_id ASC

query-plan
----------------------------------------------------------------------
->SortPlan (#blks=1, #recs=10)
        ->ProjectPlan (#blks=2, #recs=10)
                ->SelectPlan pred:() (#blks=2, #recs=10)
                        ->TablePlan (#blks=2, #recs=10)

Actual #recs: 10
```

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



```
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 pred:(d_w_id=w_id) (#blks=22, #recs=10)
                                ->ProductPlan (#blks=22, #recs=10)
                                        ->TablePlan (#blks=2, #recs=10)
                                        ->TablePlan (#blks=2, #recs=1)

Actual #recs: 1
```

Any worth mentioning

在上面的 experiment 中，access multiple table with WHERE，我們嘗試使用四個 table 來檢測 explain 的 plan info，不過 vanilladb 跑了超級無敵久都無法結束，這也推翻了我們上次的假設，”當資料量大的時候，vanilladb 有增進效能的演算法實作“。