

DB22 Assignment3_Report1

108020010 陳毅鴻

108020018 楊宗瀚

Implement Steps :

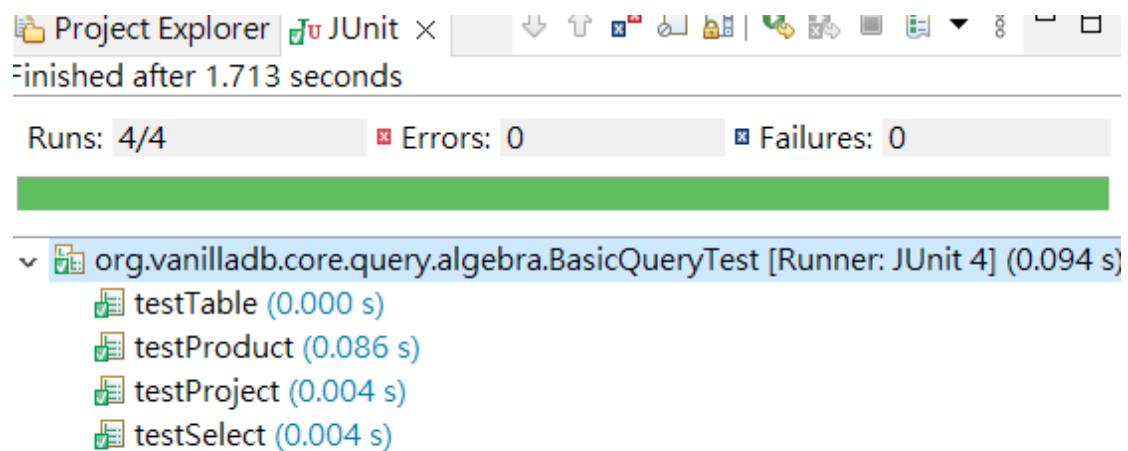
1. We add a key word "explain" at initKeywords() in Lexer.
2. In Parser, we add an if in queryCommand() to get the input command keyword "EXPLAIN", and we have a flag to store the status whether we get explain or not.
3. We add an if to see if the flag we set in Parser is true at createPlan(QueryData data, Transaction tx) in BasicQueryPlanner, if true, we call p=new explainplan(p) to new a explain plan.
4. In ProductPlan, ProjectPlan, SelectPlan, and TablePlan, we add toString() function to have the string we want, including block accessed and records output append to string builder and return to get the output string.
5. We add class ExplainScan which implements Scan, use getVal() function to return the String we want to output, and before return change its type to ConstantVarcahr. Besides, in function next(), we set the restriction that only can return true one time to make sure it output one time.
6. We add class ExplainPlan which implements Plan, we use toString() function to construct a String we want. In that function, we can call plan.toString() to know the next project's String to construct the entire String. And in Scan(), we return toString() to explanscan to let it know the output. The actual rec in output is the the recs of the plan be input in the BasicQueryPlanner.

Environment:

型號: Aspire A315-55G
處理器: Intel(R) Core(TM) i5-8265U CPU @ 1.60GHz 1.80 GHz
已安裝記憶體 (RAM) 8.00 GB (7.85 GB 可用)
系統類型: 64 位元作業系統, x64 型處理器

Windows 10

Testcases Screenshots :



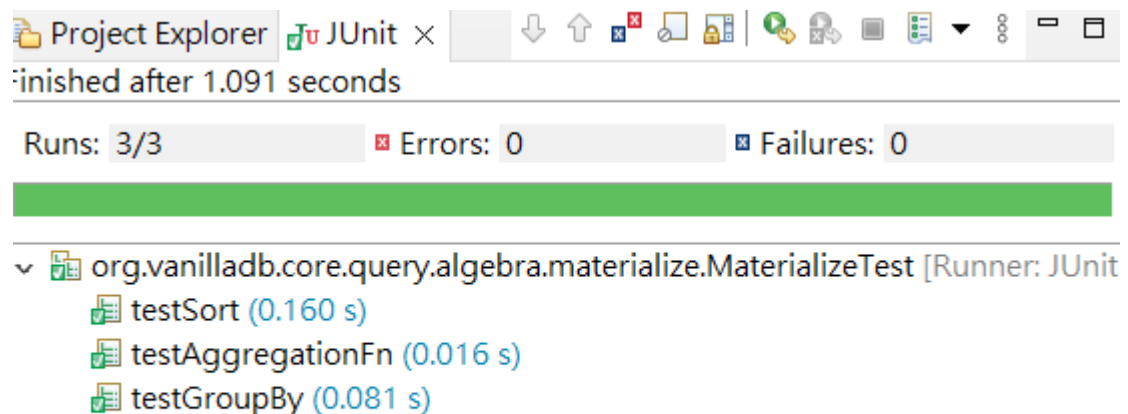
Project Explorer JUnit ×

Finished after 1.713 seconds

Runs: 4/4 Errors: 0 Failures: 0

org.vanilladb.core.query.algebra.BasicQueryTest [Runner: JUnit 4] (0.094 s)

- testTable (0.000 s)
- testProduct (0.086 s)
- testProject (0.004 s)
- testSelect (0.004 s)



Project Explorer JUnit ×

Finished after 1.091 seconds

Runs: 3/3 Errors: 0 Failures: 0

org.vanilladb.core.query.algebra.materialize.MaterializeTest [Runner: JUnit 4] (0.257 s)

- testSort (0.160 s)
- testAggregationFn (0.016 s)
- testGroupBy (0.081 s)

Project Explorer JUnit ×
Finished after 0.239 seconds

Runs: 7/7 Errors: 0 Failures: 0

- ✓ org.vanilladb.core.query.parse.ParserTest [Runner: JUnit 4] (0.055 s)
 - ✓ testIndexCreationWithGivenType (0.055 s)
 - ✓ testMultiKeysIndexCreation (0.000 s)
 - ✓ testIndexCreation (0.000 s)
- ✓ org.vanilladb.core.query.parse.ParseTest [Runner: JUnit 4] (0.007 s)
 - ✓ testParseInsert (0.001 s)
 - ✓ testParseSelect (0.005 s)
 - ✓ testCreateTable (0.001 s)
 - ✓ testParseConstant (0.000 s)

Project Explorer JUnit ×
Finished after 0.984 seconds

Runs: 12/12 Errors: 0 Failures: 0

- ✓ org.vanilladb.core.query.planner.BasicQueryPlannerTest [Runner: JUnit 4] (0.091 s)
 - ✓ testQuery (0.091 s)
 - ✓ testView (0.065 s)
 - ✓ testDelete (0.007 s)
 - ✓ testInsert (0.002 s)
 - ✓ testModify (0.006 s)
- ✓ org.vanilladb.core.query.planner.VerifierTest [Runner: JUnit 4] (0.017 s)
 - ✓ testQueryData (0.002 s)
 - ✓ testInsertData (0.002 s)
 - ✓ testModifyData (0.004 s)
 - ✓ testCreateTableData (0.001 s)
 - ✓ testCreateViewData (0.004 s)
 - ✓ testDeleteData (0.001 s)
 - ✓ testCreateIndexData (0.003 s)

Output example :

```
EXPLAIN SELECT COUNT(id) FROM tb, tb2 WHERE id=1 GROUP BY id
```

```
query-plan
```

```
-----  
->ProjectPlan (#blks=1, #recs=1)  
  ->GroupByPlan: (#blks=1, #recs=1)  
    ->SortPlan (#blks=1, #recs=2)  
      ->SelectPlan pred:(id=1.0) (#blks=6, #recs=2)  
        ->ProductPlan (#blks=6, #recs=4)  
          ->TablePlan on (tb2) (#blks=2, #recs=2)  
          ->TablePlan on (tb) (#blks=2, #recs=2)
```

```
Actual #recs: 1
```

```
SQL>
```