Database Assignment 3 Report1

Team-13： 107062302 楊博丞、108062223 王翊語

Lexer

在keywords中加入 “explain”。

Parser

在 queryCommand 的一開始判斷是否有”explain” 這個 keyword，若是有，最後 return QueryData 要加上最後一個 argument（有無 ”explain”）。

```
boolean explainQuery = false;
if (lex.matchKeyword("explain")) {
  lex.eatKeyword("explain");
  explainQuery = true;
}
```

QueryData

加上一個變數：boolean explain 表是否有此 keyword，並放到 QueryData 的 constructor 中。

```
private boolean explain;
```

另外加一個 public function: isExplain() 回傳是否有 explain。

BasicQueryPlanner

在 createPlan 的最後一步加上判斷，若有 explain 則使用上一步建立的 plan 作為參數 new 一個 ExplainPlan。

```
if (data.isExplain()) {
  p = new ExplainPlan(p);
}
```

ProductPlan

加上 public method: toString()，用來獲取 plan tree 的文字資訊。

此 method 會分別再對 right-hand & left-hand subquery call toString() 得到 children 的 plan tree。最後藉由 StringBuilder 整合當前 plan 本身 blocksAccessed, recordsOutput 的數值以及 right & left children 的 plan tree 作為新的 plan tree 文字資訊輸出。

```
public String toString() {
  String c2 = p2.toString();
  String[] cs2 = c2.split("\n");
  String c1 = p1.toString();
  String[] cs1 = c1.split("\n");
  StringBuilder sb = new StringBuilder();
  sb.append("->");
  sb.append("ProductPlan (#blks=" + blocksAccessed() + ", #recs="
      + recordsOutput() + ")\n");
  // right child
  for (String child : cs2)
    sb.append("\t").append(child).append("\n");
  ;
  // left child
  for (String child : cs1)
    sb.append("\t").append(child).append("\n");
  ;
  return sb.toString();
```

ProjectPlan

一樣加上 public method: toString()，用來獲取 plan tree 的文字資訊。也會 append block, records 資訊以及對 subquery call toString() 得到的 child 的 plan tree。作法類似 ProductPlan。

SelectPlan

一樣加上 public method: toString()，用來獲取 plan tree 的文字資訊。作法同上，也會對 subquery call toString()，但在 StringBuilder 中會在 block, records, child plan tree 資訊前多加上 predicate 的資訊。

TablePlan

一樣加上 public method: toString()，用來獲取 plan tree 的文字資訊。這邊不用再對 subquery call toString()，因為已經是 leaf 了。StringBuilder 中會在 block, records 前多加上 tableName。

ExplainPlan

Explain query 的 plan, 用來建立 plan tree 的文字資訊跟估計值。Schema 是按照 Spec，加上”query-plan" 的 field，type 為 VARCHAR(500)。藉由對傳入的 plan call toString() 來啟動對 plan child call toString 的遞迴過程來建起一棵 plan tree。

Method open() 會建立對應的 Scan，回傳一個 ExplainScan。

```
public ExplainPlan(Plan p) {
  this.p = p;
  schema.addField("query-plan", VARCHAR(500));
  planTree += p.toString();
}

@Override
public Scan open() {
  Scan s = p.open();
  return new ExplainScan(s, planTree);
}
```

ExplainScan

Explain query 的 scan，用來計算實際存取的 records 數跟回傳 explain result。

這裡會根據傳入的 scan 來計算實際的 accessed records，並在 plan tree 最後加上 number of actual accessed records。因為只有一筆資料，beforFirst() 只是讓此資料可以讀取，next() 後讀完了，則讓資料不能讀取。

```
public ExplainScan(Scan s, String planTree) {
  this.s = s;
  this.planTree = planTree;

  // calculate actually accessed records
  s.beforeFirst();
  while (s.next()) {
    actualRecords++;
  }
  s.beforeFirst();
  this.planTree += "\nActual #recs: " + actualRecords;
```

當有人想得到此 plan tree 會 call getVal() 並傳入”query-plan”，若此 field 存在，就直接回傳 plan tree，否則 raise exception。

```
public Constant getVal(String fldName) {
  if (hasField(fldName))
    return new VarcharConstant(planTree, VARCHAR(500));
  else
    throw new RuntimeException("field " + fldName + " not found.");
}
```

A query accessing single table with WHERE

```
SQL> EXPLAIN SELECT d_id, d_name FROM district WHERE d_id <= 5;

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

Actual #recs: 5
```

A query accessing multiple tables with WHERE

```
SQL> EXPLAIN SELECT w_id, d_id, d_name FROM district, warehouse WHERE d_w_id = w_id;

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

Actual #recs: 10
```

A query with ORDER BY

```
SQL> EXPLAIN SELECT c_id, c_first FROM customer WHERE c_id <= 10 AND c_d_id = 1 ORDER BY c_id DESC;

query-plan
------------------------------------------------------------------------------------------------------
->SortPlan (#blks=1, #recs=5)
       ->ProjectPlan (#blks=15001, #recs=5)
               ->SelectPlan pred:(c_id<=10.0 and c_d_id=1.0) (#blks=15001, #recs=5)
                       ->TablePlan on (customer) (#blks=15001, #recs=30000)

Actual #recs: 10
```

A query with GROUP BY and at least one aggregation function

```
SQL> EXPLAIN SELECT w_id, 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 on (warehouse) (#blks=2, #recs=1)
                                       ->TablePlan on (district) (#blks=2, #recs=10)

Actual #recs: 1
```