Introduction to Database System - Assignment 3 Report

Team 22 - 108020017 黃珮綺, 108020021 賴柏翰

Phase 1 Report

1. Implementations

Lexer

- **Add a keyword `explain`.** This allows `EXPLAIN` to be recognizable by the lexical analyzer and to be runnable by the program.

Parser

- **Determine if the query has `EXPLAIN` and store the result in QueryData.**

QueryData

- **Add an `isExplain` attribute.** Let the `QueryData` record whether the query has `EXPLAIN`.
- **Add a function that returns whether the `QueryData` is `EXPLAIN` or not.**

BasicQueryPlanner

- **Create plan.** The plan is built based on the data information recorded in `QueryData`. If the query has `EXPLAIN`, which is derived from the `QueryData`, an `ExplainPlan` is added at the last step.

Plan

- **Define a virtual function `outputString` to output the query record.**

ExplainPlan

- **This class implements `Plan` for `EXPLAIN`.** It contains a schema with single field `query-plan` of type `VARCHAR(500)`, and a plan of its subtree. On calling open, it will open its plan tree and pass the plan tree to `ExplainScan`, as well the schema and the `ouputString` of the plans. The `outputString` recursively calls the `outptString` function defined in the plans in the subtree and prints all the query information at the end.

ExplainScan

- **This class implements `Scan` for `EXPLAIN`.** It stores the output of `EXPLAIN` which is passed from the `ExplainPlan`. Also, it counts the number of the records read with the given query, and then append the result to the output. When `getVal` is called and the required field name matches `query-plan`, it will return the output result.

- **Define a flag `isBeforeFirst` as class attribute that indicates if the scan pointer is before the first record.** The function `next` will check this flag. By this mean, we can ensure that the result of `EXPLAIN` will be output only once.

Other Plans

- **Implement the virtual function `outputString` from its base class `Plan`.** We have modified classes including `TablePlan`, `ProductPlan`, `SelectPlan`, `SortPlan`, `GroupByPlan`, `ProjectPlan`, `MergeJoinPlan` and `MaterializePlan`. For each plan in the tree, the output record will follow the format `-> ${PLAN_TYPE} [optional information] (#blks=${BLOCKS_ACCESSED}, #recs=${OUTPUT_RECORDS})`. The codes are provided in the link.

2. Test Cases

1. `SQL> EXPLAIN SELECT d_id FROM district WHERE d_id > 5`

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

Actual #recs: 5
```

2. `SQL> EXPLAIN SELECT d_id, w_id 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
```

3. `SQL> EXPLAIN SELECT d_id FROM district ORDER BY d_id`

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

Actual #recs: 10
```

4. `SQL> EXPLAIN SELECT COUNT(d_id) FROM district GROUP BY d_w_id`

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

Actual #recs: 1
```

5. `SQL> EXPLAIN SELECT MIN(d_id), AVG(d_id), MAX(d_id) FROM district GROUP BY d_w_id`

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

Actual #recs: 1
```

Reference:

Source Code