Database AS3 Phase 1 Report

Implementation of `EXPLAIN`

Parsing `EXPLAIN`

- First, we added the `EXPLAIN` keyword into the lexer by adding it in `Lexer.initKeywords`.
- Then, in `Parser.queryCommand`, we make it so an `EXPLAIN` token is checked at the beginning of each statement. If one is found, it is eaten by the parser, and a flag indicating that the query is an explain query is set. That flag is passed into the `QueryData` returned by the parser.
- Afterwards, in `BasicQueryPlanner.createPlan`, we check the `QueryData` argument to see if it is an explain query (using a getter method, `QueryData.isExplainQuery`), and if so, add an `ExplainPlan` over the plan tree such that the tree is rooted in the `ExplainPlan`.

`ExplainPlan`

- `ExplainPlan` implements the `Plan` interface, with a fixed schema that contains one field ("query-plan" of type `VARCHAR(500)`). It has just one child plan.
- To support "explanation" of each plan node, we added **two** methods to the `Plan` interface:
  - `Plan.explain(StringBuilder sb, int numIndents)`: Used to let each plan define how it should format its subtree's explain string. (Plans with one subplan and plans with multiple subplans would define this differently) The string is appended to the string builder.
  - `Plan.addOptionalInfo(StringBuilder sb)`: Let each plan node define its own "optional info" string, which will be inserted into the explain string by the static method `ExplainPlan.explainNode`.
- Furthermore, the static method `ExplainPlan.explainNode(Plan p, StringBuilder sb, int numIndents)` is provided as a helper method to be used in `Plan.explain`, providing the required string formatting for each node.
- `numIndents` are passed down to provide proper indenting to the explain string to see parent-child relationships.
- As a special case, `ExplainPlan.explain` is the only `Plan.explain` implementation that doesn't invoke `ExplainPlan.explainNode`, as the `ExplainPlan` node itself isn't to be included in the explain string.
- Every other method in `ExplainPlan` simply delegates to the child plan's respective methods, as `ExplainPlan` should be transparent and have no effects on the underlying query.

`ExplainScan`

- When the `ExplainPlan` is opened, the underlying plan is opened first into a scan, and then a scan tree rooted in `ExplainScan` is created from the underlying scan. A reference to the `ExplainPlan` is also passed into the constructor, so that `ExplainScan` can access the explain string.
- What's special about `ExplainScan` is that `ExplainScan.next` will return true exactly once after calling `ExplainScan.beforeFirst`. On that first invocation of `ExplainScan.next`, the method uses a while-loop to exhaustively count the number of records in the underlying scan. It then invokes the original `ExplainPlan`'s `explain` method to get the plan tree's explain string, and then append the actual number of records accessed to that string.

- Afterwards, `ExplainScan.getVal` will convert the explain string into a `VARCHAR(500)` constant and return it as the only record in the query's result set.
- This completes the implementation of `EXPLAIN` queries.

Experiments

A query accessing single table with WHERE

- Query :

```
EXPLAIN SELECT d_id , d_name , d_street_1 , d_zip FROM district WHERE d_tax <
0.09;
```

- Result

```
query-plan
-----------------------------------------------------
->ProjectPlan  (#blks=2, #recs=0)
    ->SelectPlan pred:(d_tax<0.09) (#blks=2, #recs=0)
        ->TablePlan on (district) (#blks=2, #recs=10)
Actual #recs: 4
```

A query accessing multiple tables with WHERE

- Query :

```
 EXPLAIN SELECT c_id FROM customer,district,warehouse WHERE c_d_id=d_id and
c_w_id=w_id;
```

- Result

```
query-plan
-------------------------------------------------------------------------
->ProjectPlan  (#blks=150032, #recs=2932)
    ->SelectPlan pred:(c_d_id=d_id and c_w_id=w_id) (#blks=150032, #recs=2932)
        ->ProductPlan  (#blks=150032, #recs=300000)
            ->ProductPlan  (#blks=22, #recs=10)
                ->TablePlan on (district) (#blks=2, #recs=10)
                ->TablePlan on (warehouse) (#blks=2, #recs=1)
            ->TablePlan on (customer) (#blks=15001, #recs=30000)
Actual #recs: 30000
```

A query with ORDER BY

- Query :

```
EXPLAIN SELECT d_id , d_tax FROM district ORDER BY d_tax DESC
```

- Result :

```
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
```

A query with GROUP BY and at least one aggregation function

- Query :

```
EXPLAIN SELECT c_id,c_d_id,c_w_id,AVG(c_discount),d_name FROM
customer,district,warehouse WHERE c_d_id=d_id and c_w_id=w_id GROUP BY
c_id,c_d_id,c_w_id,d_name;
```

- Result

```
query-plan
-----------------------------------------------------------------------------
->ProjectPlan  (#blks=2994, #recs=0)
    ->GroupByPlan  (#blks=2994, #recs=896)
        ->SortPlan  (#blks=2994, #recs=2994)
            ->SelectPlan pred:(c_d_id=d_id and c_w_id=w_id) (#blks=150032,
#recs=2994)
                ->ProductPlan  (#blks=150032, #recs=300000)
                    ->ProductPlan  (#blks=22, #recs=10)
                        ->TablePlan on (district) (#blks=2, #recs=10)
                        ->TablePlan on (warehouse) (#blks=2, #recs=1)
                    ->TablePlan on (customer) (#blks=15001, #recs=30000)
Actual #recs: 3000
```

A query accessing multiple tables with WHERE , GROUP BY and ORDER BY

- Query :

```
EXPLAIN SELECT c_id,c_d_id,c_w_id,AVG(c_discount),d_name,w_name FROM
customer,district,warehouse WHERE c_d_id=d_id and c_w_id=w_id GROUP BY
c_id,c_d_id,c_w_id,d_name,w_name ORDER BY d_name DESC
```

- Result :

```
query-plan
------------------------------------------------------------------------------------
------------------------------------------------------------------------------------
------------------------------------------------------------------------------------
------------------------------------------------------------------------------------
------------------------------------------------------------------------------------
------------------------------------------------------------------------------------
--------
->SortPlan  (#blks=66, #recs=2889)
    ->ProjectPlan  (#blks=2889, #recs=2889)
        ->GroupByPlan  (#blks=2889, #recs=2889)
            ->SortPlan  (#blks=2889, #recs=2889)
                ->SelectPlan pred:(c_d_id=d_id and c_w_id=w_id) (#blks=150032,
#recs=2889)
                    ->ProductPlan  (#blks=150032, #recs=300000)
                        ->ProductPlan  (#blks=22, #recs=10)
                            ->TablePlan on (district) (#blks=2, #recs=10)
                            ->TablePlan on (warehouse) (#blks=2, #recs=1)
                        ->TablePlan on (customer) (#blks=15001, #recs=30000)
Actual #recs: 30000
```