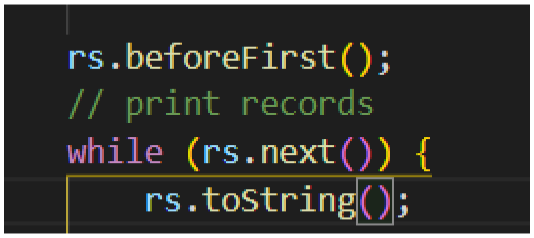
107062240 林柏均
107080006 簡立誠
107080072 朱以箴

# Spring 2022 Database Systems Assignment 3 - Phase 1 Report

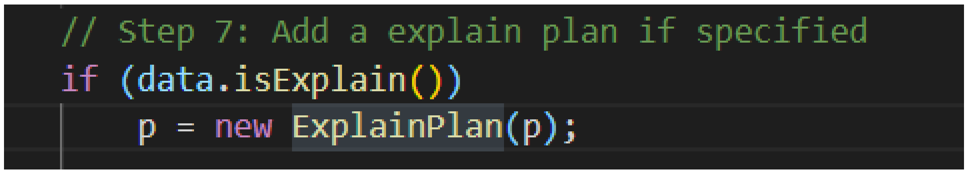
- ***Implementation***
  - consoleSQLInterpreter.java



```
rs.beforeFirst();
// print records
while (rs.next()) {
    rs.toString();
```

In the core patch, we have modified the consoleSQLInterpreter by adding a call to the "toString()" function that we have implemented. This consoleSQLInterpreter class is responsible for taking in queries from console, interpreting the query, executing the query, and printing the resulting records. For each query execution we call the "toString()" function, which is implemented in ProductPlan.java and is responsible for building the result set into a string according to the desired format. We would elaborate on the implementation of this function in more detail below.

  - BasicQueryPlanner.java



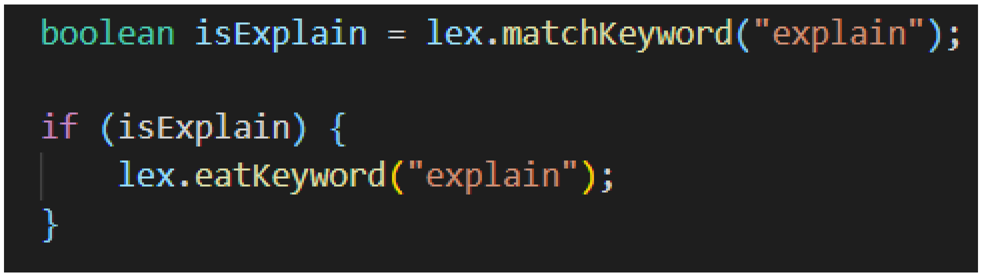
```
// Step 7: Add a explain plan if specified
if (data.isExplain())
    p = new ExplainPlan(p);
```

Since we need to implement the "Explain" command, when creating the planner, we call the ExplainPlan function, which is implemented in ExplainPlan.java, to manage the explanation of the planner algorithm.

  - QueryData.java

In the Querydata.java, we added the boolean to detect whether "Explain" is called.

  - Parser.java

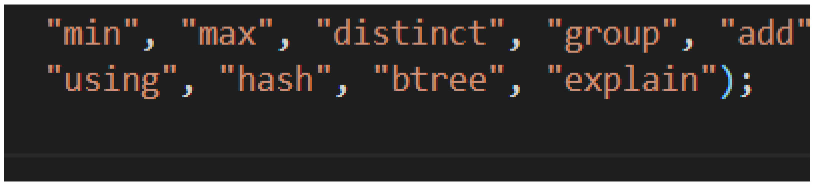


```java
boolean isExplain = lex.matchKeyword("explain");

if (isExplain) {
    lex.eatKeyword("explain");
}
```

Since the "EXPLAIN" keyword should be the first keyword in a new query, we check whether a query contains the keyword for "explain" first thing. If it does, then the lexer eats the keyword to call the corresponding function.

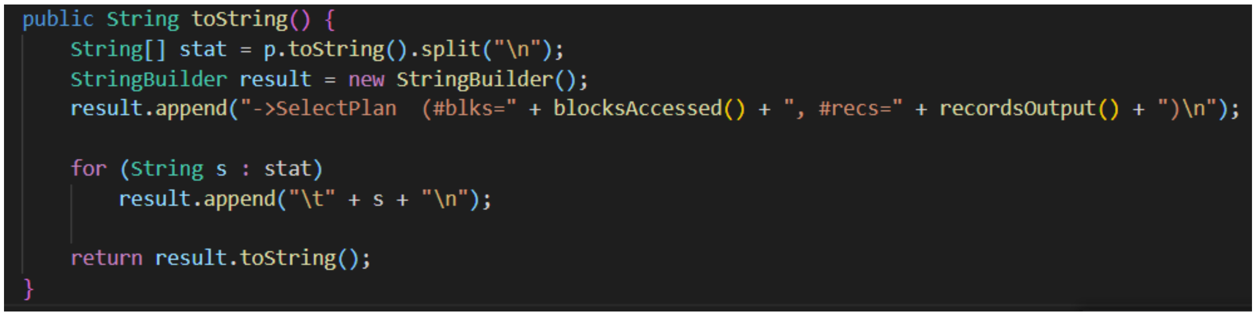
- Lexer.java



```java
 "min", "max", "distinct", "group", "add"
 "using", "hash", "btree", "explain");
```

In lexer, we just need to add the new keyword "explain."

- TablePlan.java / SelectPlan.java / ProjectPlan.java / ProductPlan.java / GroupByPlan.java / MaterializePlan.java / MergePlan.java / SortPlan.java



```java
public String toString() {
    String[] stat = p.toString().split("\n");
    StringBuilder result = new StringBuilder();
    result.append("->SelectPlan  (#blks=" + blocksAccessed() + ", #recs=" + recordsOutput() + ")\n");

    for (String s : stat)
        result.append("\t" + s + "\n");

    return result.toString();
}
```

As the above figure illustrates, we added a similar function "toString()" to all the mentioned files. The details for the implementation of the function is described below:

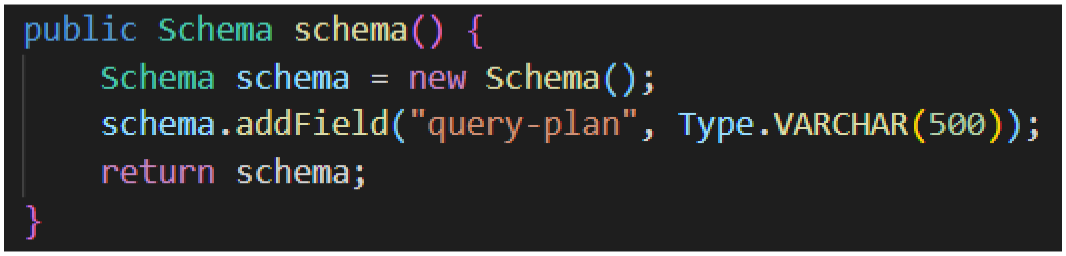
- toString()

  The function converts the two plans, for left-hand and right-hand subqueries, respectively, into strings and splits them at newline characters to create two arrays of strings. These arrays are then reconstructed into a string according to the desired format using a StringBuilder. The number of blocks accessed and the number of records output are also read and appended to the string.

In ProductPlan, this function is called in the constructor of the class, converting each instance of ProductPlan, parameterized by the left-hand and right-hand subqueries, into strings. In ExplainPlan, this function is called in open().

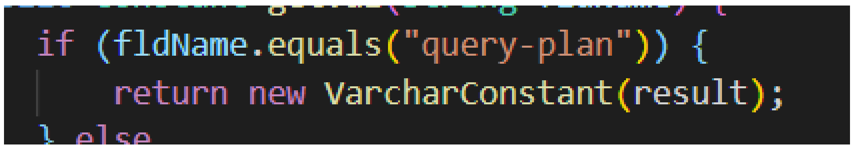
- ExplainScan.java / ExplainPlan.java

  We create two new files of "Explain" for plan and scan. In the ExplainPlan, we also added "query-plan" in the field when calling schema().



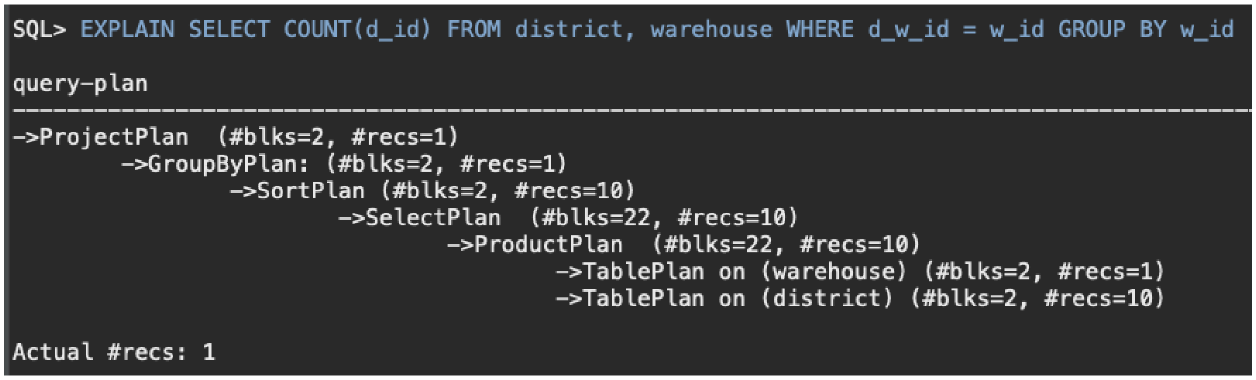
```java
public Schema schema() {
    Schema schema = new Schema();
    schema.addField("query-plan", Type.VARCHAR(500));
    return schema;
}
```

  The ExplainPlan class will call ExplainScan in open(). Then it will measure the actual number of records. It will show the actual number of records when we use the keyword "EXPLAIN" in the query, and the function gets the keyword "query-plan."



```java
if (fldName.equals("query-plan")) {
    return new VarcharConstant(result);
} else
```

- ***Some EXPLAIN Results***

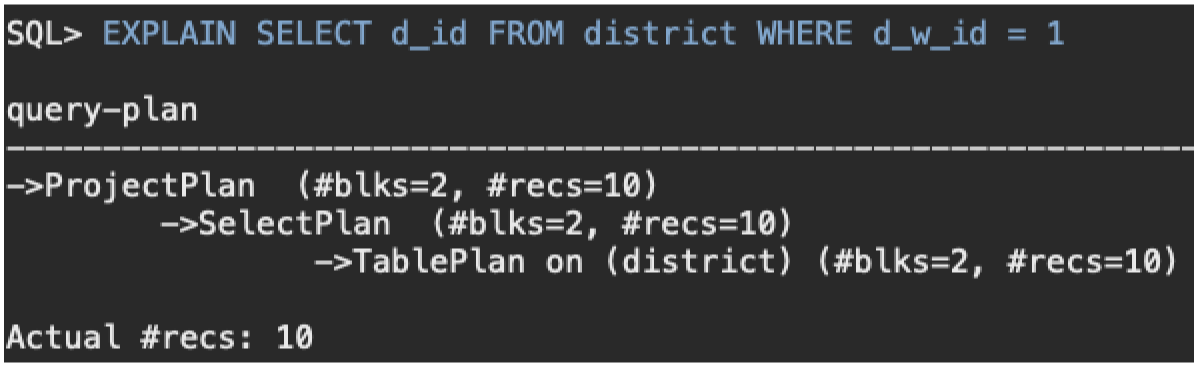


```
SQL> EXPLAIN SELECT COUNT(d_id) FROM district, warehouse WHERE d_w_id = w_id GROUP BY w_id

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

  - A query accessing single table with WHERE

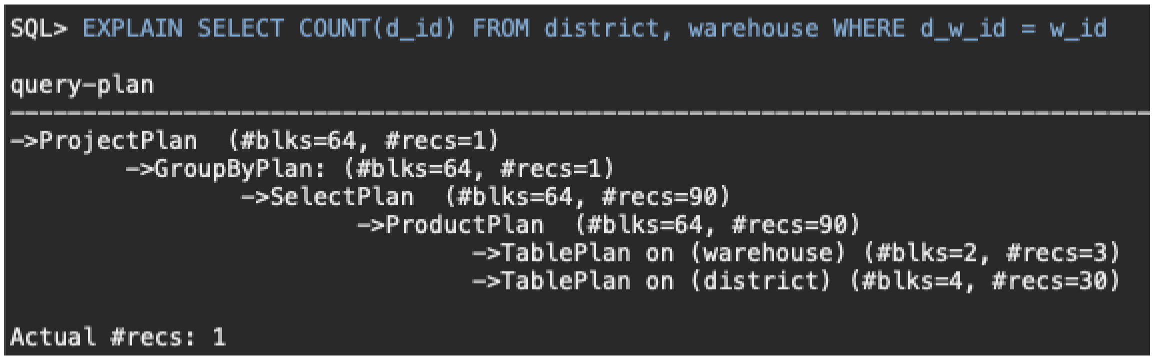


```
SQL> EXPLAIN SELECT d_id FROM district WHERE d_w_id = 1

query-plan
------------------------------------------------------------------
->ProjectPlan  (#blks=2, #recs=10)
        ->SelectPlan  (#blks=2, #recs=10)
                ->TablePlan on (district) (#blks=2, #recs=10)

Actual #recs: 10
```

- A query accessing multiple tables with WHERE

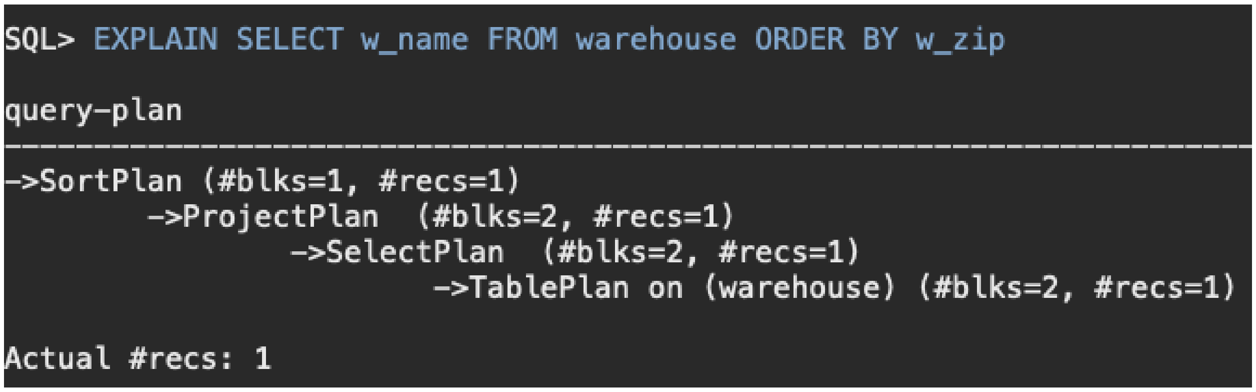


```
SQL> EXPLAIN SELECT COUNT(d_id) FROM district, warehouse WHERE d_w_id = w_id

query-plan
--------------------------------------------------------------------------------
->ProjectPlan  (#blks=64, #recs=1)
        ->GroupByPlan: (#blks=64, #recs=1)
                ->SelectPlan  (#blks=64, #recs=90)
                        ->ProductPlan  (#blks=64, #recs=90)
                                ->TablePlan on (warehouse) (#blks=2, #recs=3)
                                ->TablePlan on (district) (#blks=4, #recs=30)

Actual #recs: 1
```

- A query with ORDER BY

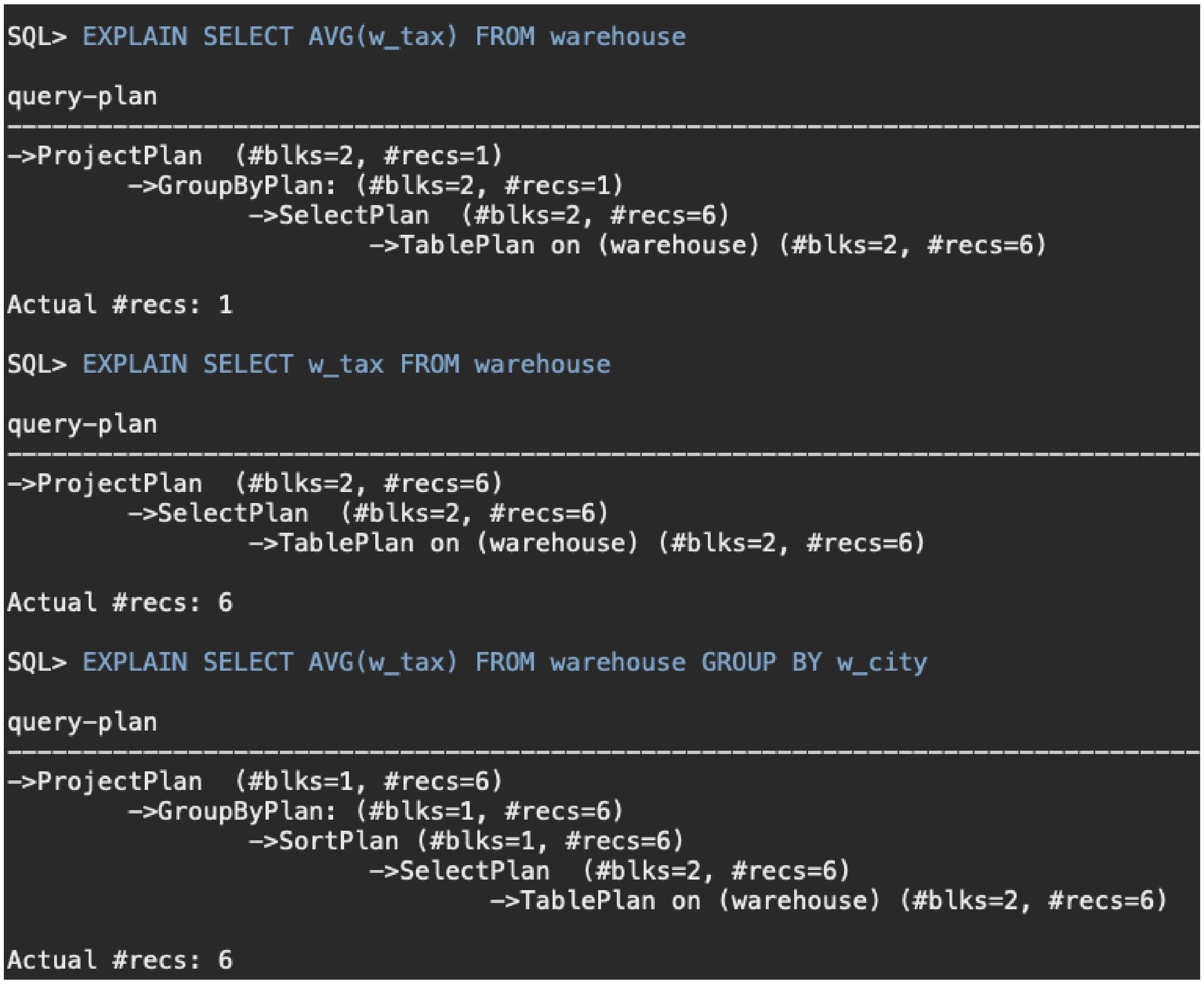


```
SQL> EXPLAIN SELECT w_name FROM warehouse ORDER BY w_zip

query-plan
----------------------------------------------------------------------
->SortPlan (#blks=1, #recs=1)
        ->ProjectPlan  (#blks=2, #recs=1)
                ->SelectPlan  (#blks=2, #recs=1)
                        ->TablePlan on (warehouse) (#blks=2, #recs=1)

Actual #recs: 1
```

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



```
SQL> EXPLAIN SELECT AVG(w_tax) FROM warehouse

query-plan
--------------------------------------------------------------------------------
->ProjectPlan  (#blks=2, #recs=1)
        ->GroupByPlan: (#blks=2, #recs=1)
                ->SelectPlan  (#blks=2, #recs=6)
                        ->TablePlan on (warehouse) (#blks=2, #recs=6)

Actual #recs: 1

SQL> EXPLAIN SELECT w_tax FROM warehouse

query-plan
--------------------------------------------------------------------------------
->ProjectPlan  (#blks=2, #recs=6)
        ->SelectPlan  (#blks=2, #recs=6)
                ->TablePlan on (warehouse) (#blks=2, #recs=6)

Actual #recs: 6

SQL> EXPLAIN SELECT AVG(w_tax) FROM warehouse GROUP BY w_city

query-plan
--------------------------------------------------------------------------------
->ProjectPlan  (#blks=1, #recs=6)
        ->GroupByPlan: (#blks=1, #recs=6)
                ->SortPlan (#blks=1, #recs=6)
                        ->SelectPlan  (#blks=2, #recs=6)
                                ->TablePlan on (warehouse) (#blks=2, #recs=6)

Actual #recs: 6
```