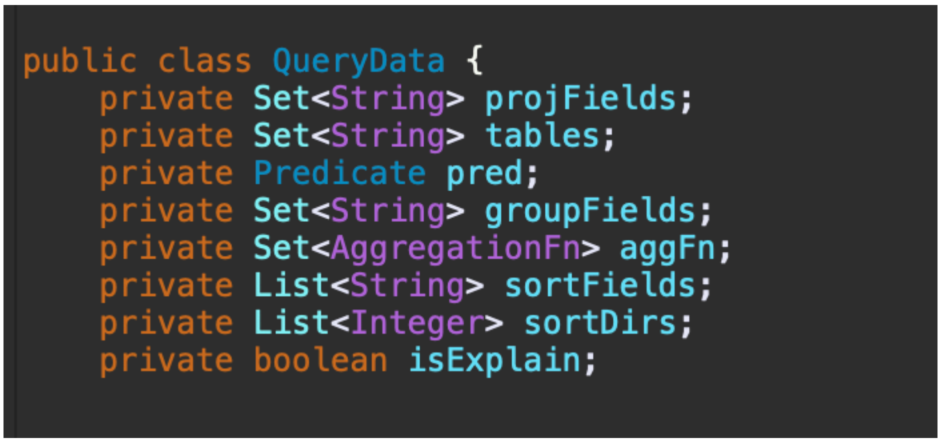
Assignment3 Report

105062206蔡哲維
105062306胡世昕

Modify Querydata:

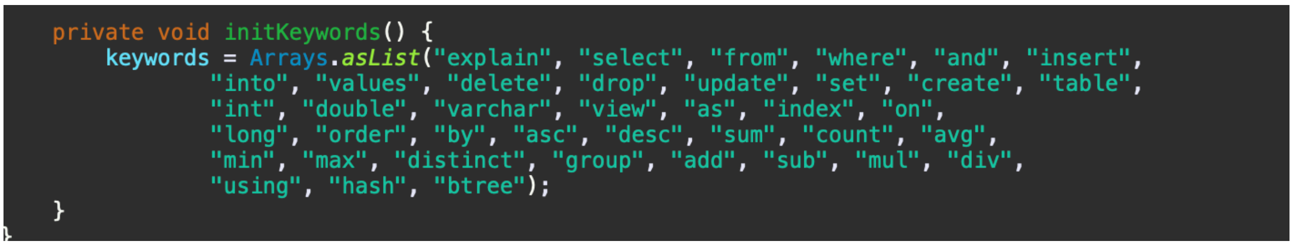
在Querydata中加入新的member isExplain，若其為true，表示user輸入的sql指令為explain，若為false，則非。



```
public class QueryData {
    private Set<String> projFields;
    private Set<String> tables;
    private Predicate pred;
    private Set<String> groupFields;
    private Set<AggregationFn> aggFn;
    private List<String> sortFields;
    private List<Integer> sortDirs;
    private boolean isExplain;
```

Modify Lexer:

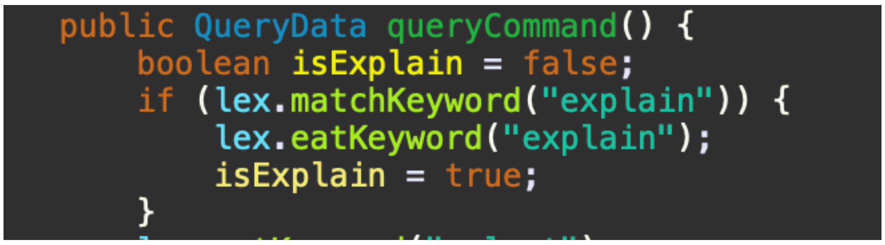
在lexer的keyowrd list中，加入'explain'。



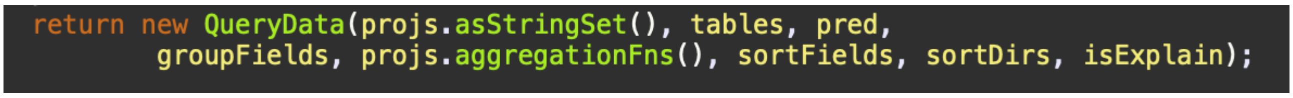
```
    private void initKeywords() {
        keywords = Arrays.asList("explain", "select", "from", "where", "and", "insert",
                "into", "values", "delete", "drop", "update", "set", "create", "table",
                "int", "double", "varchar", "view", "as", "index", "on",
                "long", "order", "by", "asc", "desc", "sum", "count", "avg",
                "min", "max", "distinct", "group", "add", "sub", "mul", "div",
                "using", "hash", "btree");
    }
}
```

Modify Parser:

在queryCommand這個method中，首先判斷傳入的cmd是否為explain開頭，若是，則將local variable isExplain設為true，否則維持false。最後將此variable傳入QueryData的建構子中。



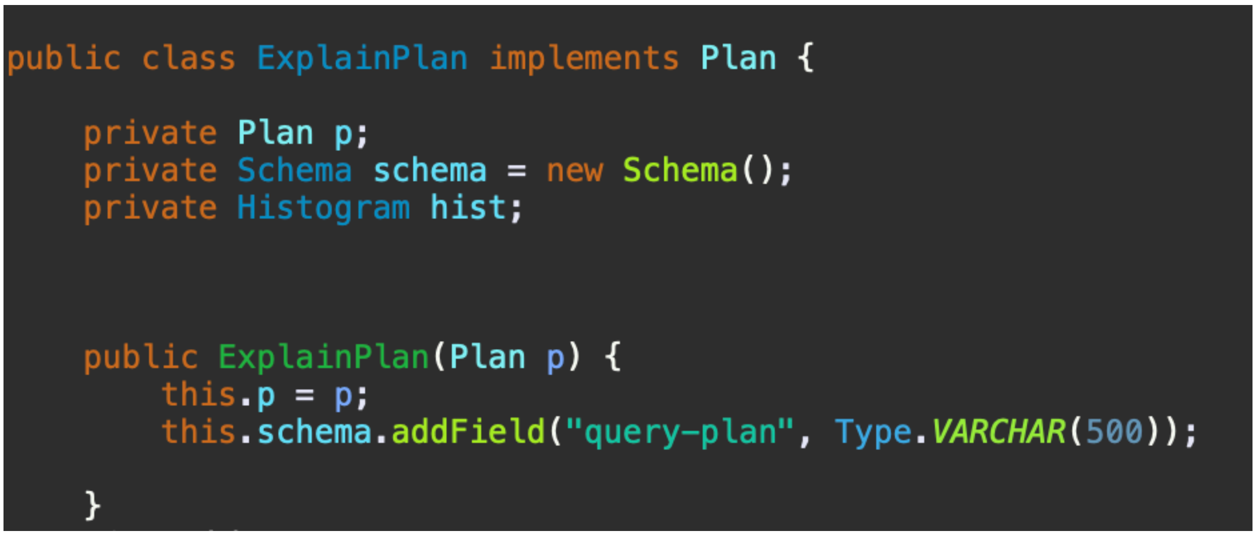
```
public QueryData queryCommand() {
    boolean isExplain = false;
    if (lex.matchKeyword("explain")) {
        lex.eatKeyword("explain");
        isExplain = true;
    }
```



```
return new QueryData(projs.asStringSet(), tables, pred,
        groupFields, projs.aggregationFns(), sortFields, sortDirs, isExplain);
```

Add ExplainPlan:

新增一個叫做ExplainPlan的class，其member包含schema、histogram、以及包住下層plan的p。在建構ExplainPlan的instance時，需要傳入其下層的plan instance，將member p reference到該instance。同時，加入’query-plan’這個field進入schema，其type為varchar(500)。



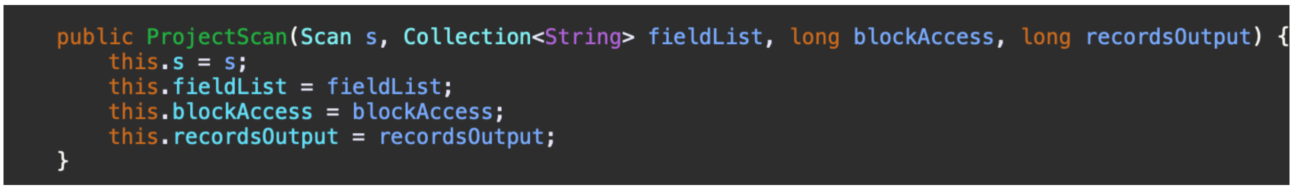
```
public class ExplainPlan implements Plan {

    private Plan p;
    private Schema schema = new Schema();
    private Histogram hist;


    public ExplainPlan(Plan p) {
        this.p = p;
        this.schema.addField("query-plan", Type.VARCHAR(500));

    }
```

另外在open的method中，呼叫p的open，得到其scan，並將其scan傳入ExplainScan的建構子中。

Modify plan classes(projectplan, selectplan…) and scan classes(projectscan, selectscan…)

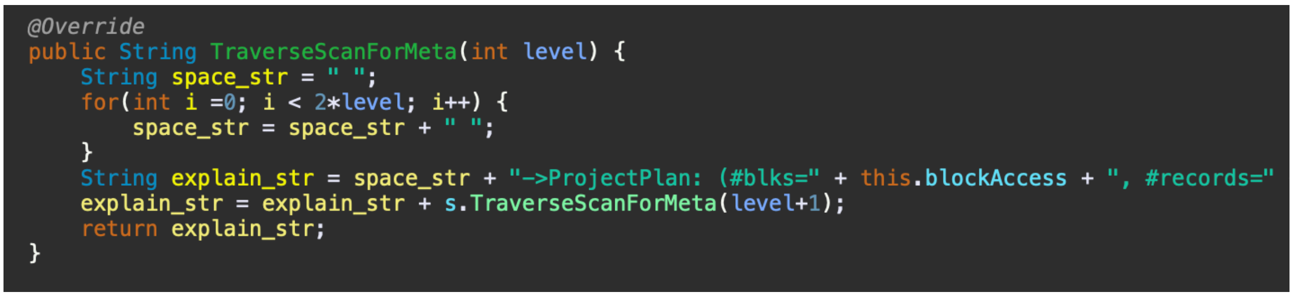
因為Explain指令要去輸出每一層plan估計的block access和records，所以我們必須讓explain scan能夠讀取到每一層的資訊。這讓我們想到要去些微修改既有的plan classes和scan classes。

首先在每一個scan classes中，新增兩個member：blockAccess和recorderOutput，並新增一個建構子，其接收參數多加了block access和record output。在建構子中將參數之值賦予給相對應的member。（在sortscan中，還要多加一個scan member src接收的是下層plan的scan）



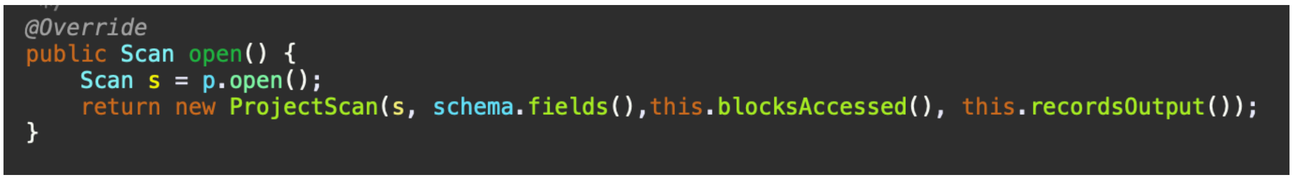
```
public ProjectScan(Scan s, Collection<String> fieldList, long blockAccess, long recordsOutput) {
    this.s = s;
    this.fieldList = fieldList;
    this.blockAccess = blockAccess;
    this.recordsOutput = recordsOutput;
}
```

另外，在scan classes中，新增了一個叫做TraverseScanForMeta的method，其會將當前scan的blockaccess和recordOutput 加入輸出字串中，並呼叫下層scan的TraverScanForMeta，最後將下層回傳的字串append到當前scan的輸出字串後面，再回傳。此method在tablescan時會結束recursion。



```
@Override
public String TraverseScanForMeta(int level) {
    String space_str = " ";
    for(int i =0; i < 2*level; i++) {
        space_str = space_str + " ";
    }
    String explain_str = space_str + "->ProjectPlan: (#blks=" + this.blockAccess + ", #records="
    explain_str = explain_str + s.TraverseScanForMeta(level+1);
    return explain_str;
}
```

接著，修改plan classes。在各個class的open method中，new其scan instance時，傳入的參數要加上plan.blockAccess()和plan.recordOutput()回傳的值。(在sortplan中，還要傳入其下層plan的scan)

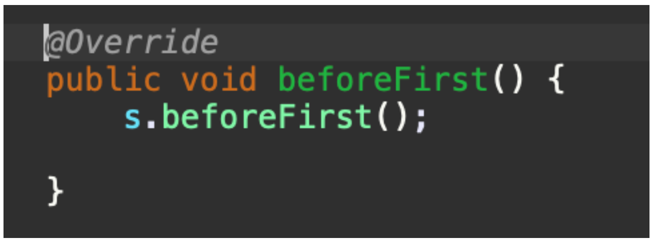


```
@Override
public Scan open() {
    Scan s = p.open();
    return new ProjectScan(s, schema.fields(),this.blocksAccessed(), this.recordsOutput());
}
```

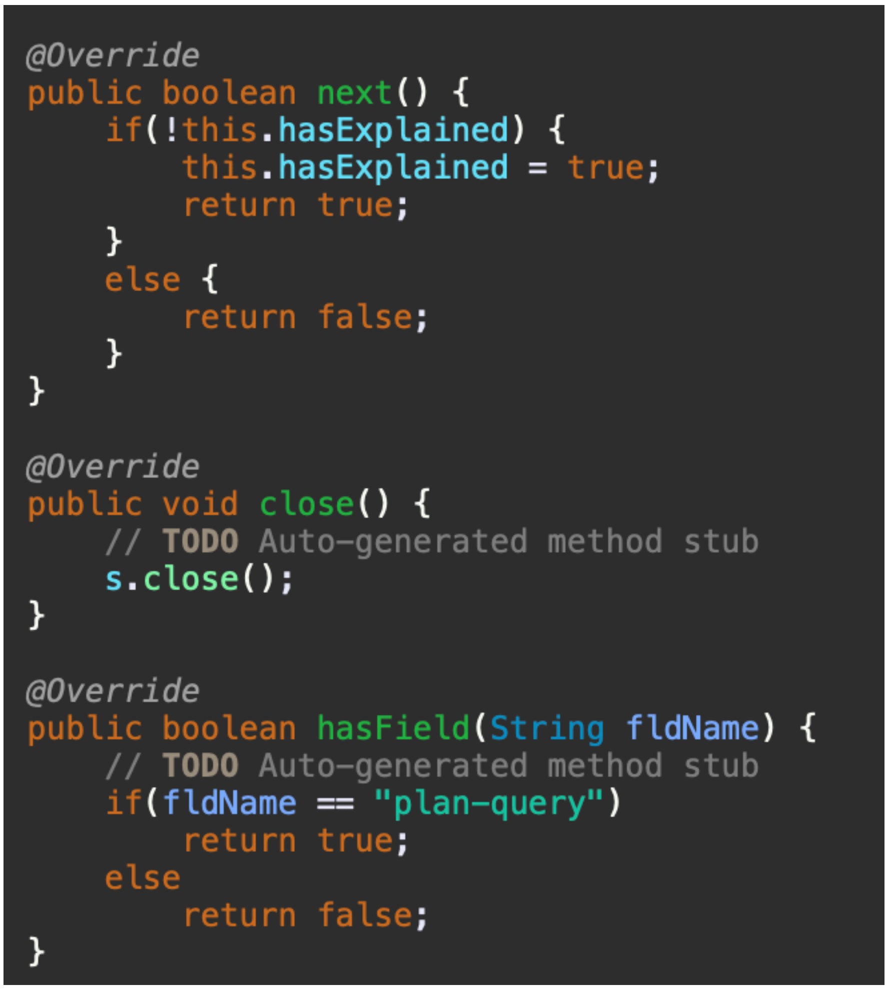
Add ExplainScan:

在ExplainScan中，其beforefirst會呼叫下層scan的beforefirst。其hasfield會去檢查傳入的field是否為'query-plan'，若是則回傳true，否則回傳false。next method則會依據class member hasExplained(初設值為false)來決定是否回傳true，若該變數為false，則將其改為true並回傳true，反之回傳false。

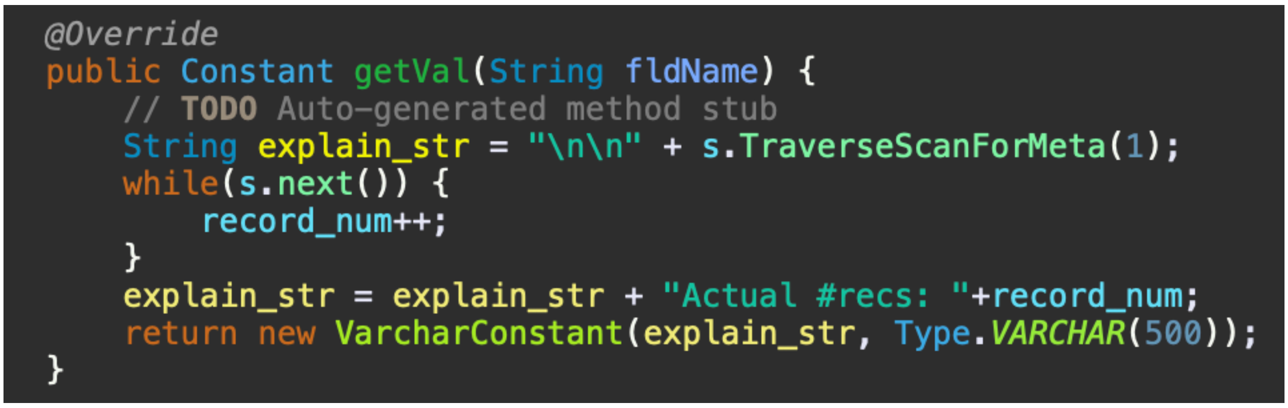
此外在getval的部分則呼叫下層scan的TraverseScanForMeta，並接收其回傳字串，當作plantree的輸出。而actual record access，則透過不斷呼叫s.next，並在迴圈中讓counter不斷加一，直到s.next回傳的事false為止。



```
@Override
public void beforeFirst() {
    s.beforeFirst();

}
```



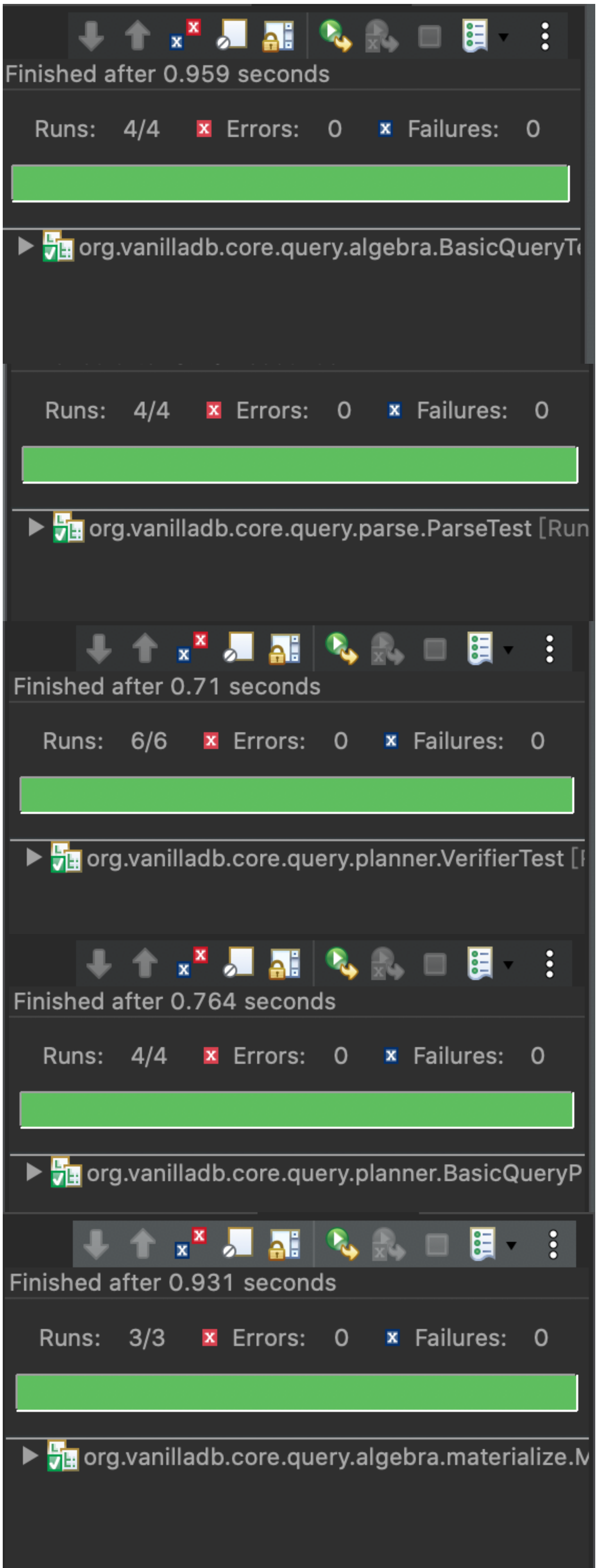
```
@Override
public boolean next() {
    if(!this.hasExplained) {
        this.hasExplained = true;
        return true;
    }
    else {
        return false;
    }
}

@Override
public void close() {
    // TODO Auto-generated method stub
    s.close();
}

@Override
public boolean hasField(String fldName) {
    // TODO Auto-generated method stub
    if(fldName == "plan-query")
        return true;
    else
        return false;
}
```



```
@Override
public Constant getVal(String fldName) {
    // TODO Auto-generated method stub
    String explain_str = "\n\n" + s.TraverseScanForMeta(1);
    while(s.next()) {
        record_num++;
    }
    explain_str = explain_str + "Actual #recs: "+record_num;
    return new VarcharConstant(explain_str, Type.VARCHAR(500));
}
```

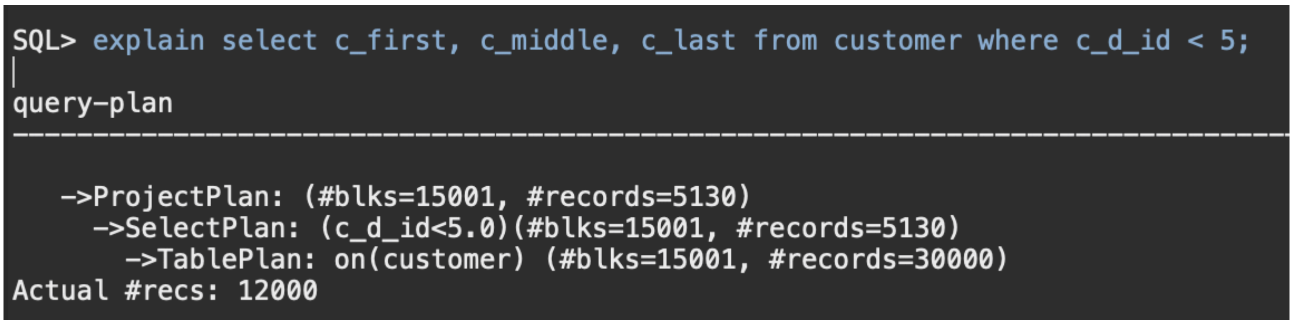
最後將所取得的所有資訊，以VarcharConstant包起來，並令其type為varchar(500)，回傳給console端輸出。

Junit Test:

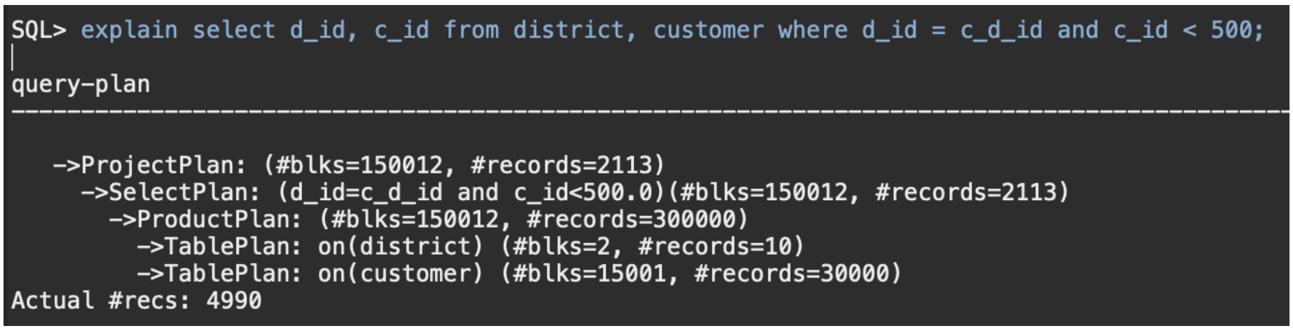


```
Finished after 0.959 seconds
Runs: 4/4   Errors: 0   Failures: 0
▶ org.vanilladb.core.query.algebra.BasicQueryT

Runs: 4/4   Errors: 0   Failures: 0
▶ org.vanilladb.core.query.parse.ParseTest [Run

Finished after 0.71 seconds
Runs: 6/6   Errors: 0   Failures: 0
▶ org.vanilladb.core.query.planner.VerifierTest [

Finished after 0.764 seconds
Runs: 4/4   Errors: 0   Failures: 0
▶ org.vanilladb.core.query.planner.BasicQueryP

Finished after 0.931 seconds
Runs: 3/3   Errors: 0   Failures: 0
▶ org.vanilladb.core.query.algebra.materialize.M
```

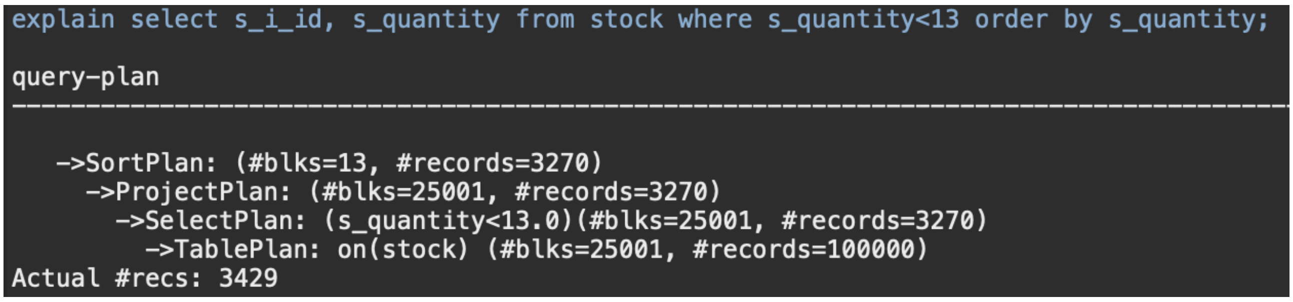
Experiments:



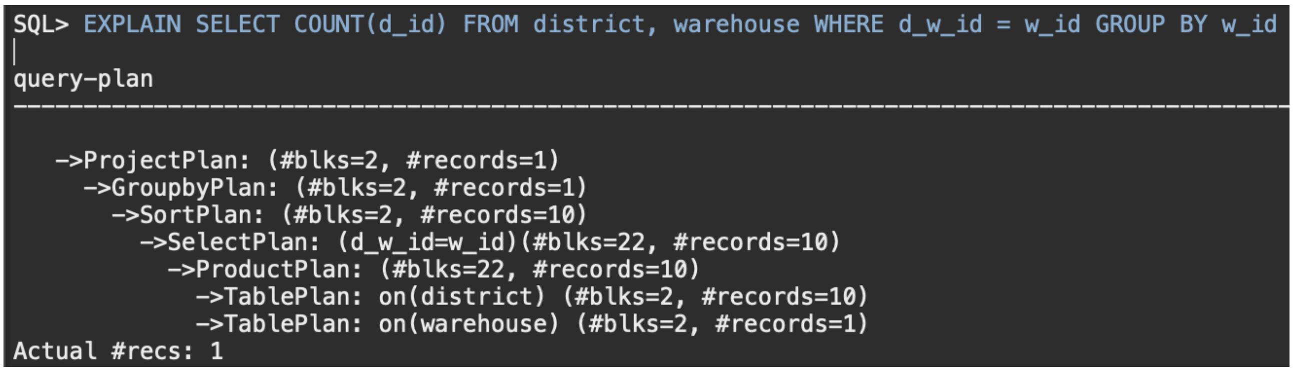
```
SQL> explain select c_first, c_middle, c_last from customer where c_d_id < 5;
|
query-plan
--------------------------------------------------------------------------------

   ->ProjectPlan: (#blks=15001, #records=5130)
     ->SelectPlan: (c_d_id<5.0)(#blks=15001, #records=5130)
       ->TablePlan: on(customer) (#blks=15001, #records=30000)
Actual #recs: 12000
```



```
SQL> explain select d_id, c_id from district, customer where d_id = c_d_id and c_id < 500;
|
query-plan
--------------------------------------------------------------------------------

   ->ProjectPlan: (#blks=150012, #records=2113)
     ->SelectPlan: (d_id=c_d_id and c_id<500.0)(#blks=150012, #records=2113)
       ->ProductPlan: (#blks=150012, #records=300000)
         ->TablePlan: on(district) (#blks=2, #records=10)
         ->TablePlan: on(customer) (#blks=15001, #records=30000)
Actual #recs: 4990
```



```
explain select s_i_id, s_quantity from stock where s_quantity<13 order by s_quantity;

query-plan
--------------------------------------------------------------------------------

   ->SortPlan: (#blks=13, #records=3270)
     ->ProjectPlan: (#blks=25001, #records=3270)
       ->SelectPlan: (s_quantity<13.0)(#blks=25001, #records=3270)
         ->TablePlan: on(stock) (#blks=25001, #records=100000)
Actual #recs: 3429
```



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

   ->ProjectPlan: (#blks=2, #records=1)
     ->GroupbyPlan: (#blks=2, #records=1)
       ->SortPlan: (#blks=2, #records=10)
         ->SelectPlan: (d_w_id=w_id)(#blks=22, #records=10)
           ->ProductPlan: (#blks=22, #records=10)
             ->TablePlan: on(district) (#blks=2, #records=10)
             ->TablePlan: on(warehouse) (#blks=2, #records=1)
Actual #recs: 1
```