Team5 Assignment3 Report1

107062313 黃寶萱　　107062117 李采蓉　　107062317 陳怡汝

◈ Part1 : Parse

A. org.vanilladb.core.query.parse.Lexer.java

```
private void initKeywords() {
    keywords = Arrays.asList("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", "explain");
}
```

由於 Lexer 會負責將讀取到的 token 轉換成小寫，為了解析 EXPLAIN，只需在 Keywords 中新增"explain"。

B. org.vanilladb.core.query.parse.Parser.java

```
public QueryData queryCommand() {
    boolean flag = false;
    Set<String> explainFields = new HashSet<String>();
    if(lex.matchKeyword("explain")) {
        flag = true;
        lex.eatKeyword("explain");
        explainFields.add("query-plan");
    }
```

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

新增 EXPLAIN 之後 Query 的文法中修改如下：

<Query> := [EXPLAIN] SELECT <ProjectSet> FROM <TableSet>
[WHERE <Predicate>] [GROUP BY <IdSet>]
[ORDER BY <SortList> [DESC | ASC]]

由於 Parser 負責解析文法，故於 Parser 的 queryCommand()中進行以下修改：

1. 先判斷 lexer 有無找到"explain"這個關鍵字，若有則讓 flag 值設成 true，並且藉由 lexer 讀取"explain"。
2. 按照下方新增 field 的方式，若有 EXPLAIN 則在 explainField 新增一個 query-plan 的 string，若沒有則 explainField 為空 set。
3. Return value 透過 QueryData 新增 flag 和 explainField 兩個參數。

C. org.vanilladb.core.query.parse.QueryData.java

```
public QueryData(boolean isExplain, Set<String> explainFields, Set<String> projFields, Set<String> tables, Predicate pred,
        Set<String> groupFields, Set<AggregationFn> aggFn, List<String> sortFields, List<Integer> sortDirs) {
    this.isExplain = isExplain;
    this.explainFields = explainFields;
    this.projFields = projFields;
    this.tables = tables;
    this.pred = pred;
    this.groupFields = groupFields;
    this.aggFn = aggFn;
    this.sortFields = sortFields;
    this.sortDirs = sortDirs;
}
```

```
public boolean isExplain() {
    if(isExplain==true) return true;
    else return false;
}

public Set<String> explainFields() {
    return explainFields;
}
```

```
public String toString() {
    StringBuilder result = new StringBuilder();;
    if (isExplain==true)
        result.append("explain ");
```

由於在 Parser 需要 QueryData 新增 isExplain 和 explainFields 兩個參數，故於此中進行以下修改：

1. 修改 constructor 中變數。其中 isExplain 代表下的 SQL 中有"EXPLAIN"，而 explainFields 則代表若有 EXPLAIN，則 explain output 的結果會存在 explainFields 裡新增的 query-plan 中。
2. 新增回傳 isExplain 和 explainFields 的 functions。
3. 在 toString 中最初判斷，若有 EXPLAIN，則讓回傳的 result 新增 "explain "

◈ Part2 : ExplainPlan and ExplainScan

A. org.vanilladb.core.query.algebra.ExplainPlan.java

我們參考 ProjectPlan 的方式新增一個 ExplainPlan() class，負責處理 explain plan 的行爲，並對以下 function 進行修改。

1. Constructor：在這邊我們建立 Explain plan 的 schema，fldNames 為只有一個”query-plan”的 set，由此添加類型為 VARCHAR(500)的 query plan field 到 schema 內。

```
public ExplainPlan(Plan p, Set<String> fldNames) {
    this.p = p;
    for (String fldname : fldNames)
        schema.addField(fldname, Type.VARCHAR(500));
    hist = projectHistogram(p.histogram(), fldNames);
}
```

2. toString()：當 Explain plan 的 toString 被呼叫到時，我們新增 Explain 需要印出的內容，在這邊為 “query-plan”及分隔線。

```
@Override
public String toString() {
    StringBuilder sb = new StringBuilder();
    sb.append(schema.fields().first() + "\n");
    sb.append("-------------------------------------------------------------------\n");
    return sb.toString();
}
```

3. open()：在 Explain plan 被 open 的時候同時創建 ExplainScan，並且呼叫 p.toString()，這個 function 會回傳 Explain plan 執行的結果，將這個結果傳進 ExplainScan 內並進行後續操作。

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

B. org.vanilladb.core.query.algebra.ExplainScan.java

我們一樣參考 ProjectScan 的方式新增一個 ExplainScan class，並對以下 function 進行修改。

1. Parameter :

在 ExplainScan 裡面除了原本紀錄的變數之外，另外紀錄 result、numRecords、flag 變數。result 負責紀錄 Explain 執行完畢後需要輸出的結果，numRecords 則負責統計 record 的數量，flag 則是進行是否已經印過 Explain 結果的判斷，以確保我們只會印出一次 Explain 的結果。

```
public class ExplainScan implements Scan {
    private Scan s;
    private Collection<String> fieldList;
    private String result = "";
    private int numRecords = 0;
    private boolean flag = false;
```

2. beforeFirst / next :

   我們在 beforeFirst 裡面將 flag 設成 true，並在 next 時判斷如果 flag 是 true 時才能回傳 true，並把 flag 設為 false，以確保只會印出一次 Explain 的結果。

```java
@Override
public void beforeFirst() {
    flag = true;
    s.beforeFirst();
}

@Override
public boolean next() {
    if(flag==true) {
        flag = false;
        return true;
    }
    else return false;
}
```

3. Constructor：

   在這邊的 resultstring 其實就是紀錄執行 Explain 之後得到的結果，我們取得 Explain 執行的結果之後，計算其中有多少 record，最後將加上 record 的統計資料與 Explain 執行結果合併，得到最終結果。

```java
public ExplainScan(Scan s, Collection<String> fieldList, String resultstring) {
    this.s = s;
    this.fieldList = fieldList;

    this.result = "\n" + resultstring;
    s.beforeFirst();
    while(s.next())
        numRecords++;
    s.close();
    this.result += "\nActual #recs: " + numRecords;
}
```

4. getVal：

   最後，在 client 端呼叫到 getVal 時，將結果包成 Constant 並回傳。

```java
@Override
public Constant getVal(String fldName) {
    if (fldName.equals("query-plan"))
        return new VarcharConstant(result);
    else
        throw new RuntimeException("field " + fldName + " not found.");
}
```

◈ Part3 : TablePlan, SelectPlan, ProjectPlan, ProductPlan

我們參考 SortPlan 和 MergeJoinPlan 中 toString()的寫法，根據 Planner 所建的 plan tree 呼叫對應的 plan，並記錄本次作業要求的 SQL 指令執行過程。

A. org.vanilladb.core.query.algebra.TablePlan.java

```
@Override
public String toString() {
    StringBuilder sb = new StringBuilder();
    sb.append("->");
    sb.append("TablePlan on (" + ti.tableName() +
              ") (#blks=" + blocksAccessed() + ", #recs=" + recordsOutput() + ")\n");
    return sb.toString();
}
```

因為 TablePlan 已經是 plan tree 的 leaf node，因此可以直接建立一個新的 StringBuilder，將 TablePlan explain 資訊記錄下來並 return。其中需要 block access 數量以及 record 數量可以直接呼叫定義好的 function blockAccessed()和 recordsOutput()取得。

B. org.vanilladb.core.query.algebra.SelectPlan.java

```
@Override
public String toString() {
    String c = p.toString();
    String[] cs = c.split("\n");
    StringBuilder sb = new StringBuilder();
    sb.append("->");
    sb.append("SelectPlan pred:(" + pred.toString() +
              ") (#blks=" + blocksAccessed() + ", #recs=" + recordsOutput() + ")\n");
    for (String child : cs)
        sb.append("\t").append(child).append("\n");
    return sb.toString();
}
```

與 SortPlan 架構相同，因為最終結果需要符合 hierarchical 結構，因此先將原本紀錄的 explain 資訊根據'\n'作切割，存在 string array cs[]中，並利用 StringBuilder 紀錄這一層的 SelectPlan explain 資訊，再將 cs[]的資料依序 append 在 SelectPlan explain 後面，其中還需要先 append '\t'達到階層式的結果。

C. org.vanilladb.core.query.algebra.ProjectPlan.java

```
@Override
public String toString() {
    String c = p.toString();
    String[] cs = c.split("\n");
    StringBuilder sb = new StringBuilder();
    sb.append("->");
    sb.append("ProjectPlan (#blks=" + blocksAccessed() + ", #recs=" + recordsOutput() + ")\n");
    for (String child : cs)
        sb.append("\t").append(child).append("\n");
    return sb.toString();
}
```

與上方 SelectPlan 的作法相同。

D. org.vanilladb.core.query.algebra.ProductPlan.java

```java
@Override
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();
}
```

ProductPlan 負責合併兩個 table(right child 和 left child)，因此同樣將資料根據'\n'切割後，先將這層 ProductPlan explain 結果存入 StringBuilder 中，需要再將兩個 child node 的 explain 結果都存入並 return。

◈ Part4 : Planner

A. org.vanilladb.core.query.planner.BasicQueryPlanner.java

```java
// Step7 : Add a explain plan if specified
if(data.isExplain()==true) {
    p = new ExplainPlan(p, data.explainFields());
}
return p;
```

在 createPlan()的最後新增，若有 EXPLAIN 則將先前已經按照 Keywords 建立好的 plan p 和 QueryData data 中的 explainFields()傳給 ExplainPlan()進行運算，最終將得到的 plan 接回 p 並 return。

◈ Part5 : Test Result

A. single table with WHERE

```
SQL> EXPLAIN SELECT o_id FROM orders WHERE o_id<100

query-plan
------------------------------------------------------------------------------
->ProjectPlan (#blks=296, #recs=953)
        ->SelectPlan pred:(o_id<100.0) (#blks=296, #recs=953)
                ->TablePlan on (orders) (#blks=296, #recs=30000)

Actual #recs: 990
```

B. multiple tables with WHERE

```
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 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
```

C. ORDER BY

```
SQL> EXPLAIN SELECT s_quantity FROM stock ORDER BY s_quantity

query-plan
-----------------------------------------------------------------------------------------------------
->SortPlan (#blks=393, #recs=100000)
        ->ProjectPlan (#blks=25001, #recs=100000)
                ->SelectPlan pred:() (#blks=25001, #recs=100000)
                        ->TablePlan on (stock) (#blks=25001, #recs=100000)

Actual #recs: 100000
```

D. GROUP BY with aggregation function

```
SQL> EXPLAIN SELECT MAX(i_price) FROM item GROUP BY i_price

query-plan
-----------------------------------------------------------------------------------------------------
->ProjectPlan (#blks=6250, #recs=107)
        ->GroupByPlan: (#blks=6250, #recs=107)
                ->SortPlan (#blks=6250, #recs=100000)
                        ->SelectPlan pred:() (#blks=6251, #recs=100000)
                                ->TablePlan on (item) (#blks=6251, #recs=100000)

Actual #recs: 99
```

◈ JUnit Test passed