CS471000: Introduction to Database Systems

Assignment 2

Report

109062102 侯皓予、110026507 唐健桓、109062303 劉元愷

2022/04/13

(一)Implementation

```
190  190          private void initKeywords() {
191       -              keywords = Arrays.asList("select", "from", "where", "and", "insert",
     191  +              keywords = Arrays.asList("explain","select", "from", "where", "and", "insert",
192  192                                       "into", "values", "delete", "drop", "update", "set", "create", "table",
193  193                                       "int", "double", "varchar", "view", "as", "index", "on",
194  194                                       "long", "order", "by", "asc", "desc", "sum", "count", "avg",
```

首先，我們在 Lexer 中增加了 explain 這個 keyword，讓他可以被辨識。

```
56        -      public QueryData(Set<String> projFields, Set<String> tables, Predicate pred,
     59   +      public QueryData(boolean explain,Set<String> projFields, Set<String> tables, Predicate pred,
57   60                           Set<String> groupFields, Set<AggregationFn> aggFn, List<String> sortFields, List<Integer>
 sortDirs) {
     61   +              this.explain=explain;
58   62                 this.projFields = projFields;
59   63                 this.tables = tables;
60   64                 this.pred = pred;
...  ...   @@ -128,9 +132,14 @@ public class QueryData {
128  132          public Set<AggregationFn> aggregationFn() {
129  133                 return aggFn;
130  134          }
     135  +      public boolean isexplain() {
     136  +              return explain;
     137  +      }
131  138
132  139          public String toString() {
133       -              StringBuilder result = new StringBuilder();;
     140  +              StringBuilder result = new StringBuilder();
     141  +              if (explain)
     142  +                      result.append("explain ");
```

然後修改了 QueryData，讓他也包含 explain 在內，同時也實作了 toString 裡面有關 explain 的部分，使其可以在需要時印出結果。

```
233  +              Set<String> explainFields = null;
234  +              boolean explain = false;
235  +              if (lex.matchKeyword("explain")) {
236  +                      explain=true;
237  +                      lex.eatKeyword("explain");
238  +              }
```

接著，會在 Parser 中讓他透過 lexer 辨識 sql 指令中是否有 explain，並且把辨識的結果存入變數，再傳給其後生成的 queryData 物件。

```
     135  +      public boolean isexplain() {
     136  +              return explain;
     137  +      }
```

並且寫一個 method 回傳 explain 值

```
262       -              return new QueryData(projs.asStringSet(), tables, pred,
     271  +              return new QueryData(explain,projs.asStringSet(), tables, pred,
263  272                                  groupFields, projs.aggregationFns(), sortFields, sortDirs);
264  273          }
```

最後在 return 的部分告訴 QueryData()傳過去的 explain 的參數值

```
70  +                // Step 7: Add a explain plan if specified
71  +                if (data.isexplain())
72  +                        p = new ExplainPlan(p);
```

在 BasicQueryPlanner 中，我們會藉由之前實作的函數去檢查是否有 explain，如果是的話會在 tree 上層增加 explainPlan。

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

然後，ExplainPlan 的 schema 只有一個 field，也就是要利用 varchar 去 output 出的所有結果。

```
28      @Override
29      public Scan open() {
30          return new ExplainScan(p, p.open(), schema);
31      }
```

在 ExplainPlan 的 open()函數中，我們直接回傳新的 ExplainScan 物件，並傳下層的 plan、它的 scan 以及 ExplainPlan 的 schema 進去。

```
public ExplainScan(Plan p, Scan s, Schema schema) {
        this.schema = schema;
        result = p.toString();

        s.beforeFirst();
        while (s.next())
                numRecs++;
        s.close();
        this.result = result + "Actual #recs: " + numRecs;
}
```

explainScan 中，我們呼叫下層 plan 的 toString()，以遞迴的方式產生我們要的字串，並且同時利用下層的 scan 去計算實際的 records 數量。

```
50      @Override
51      public Constant getVal(String fldName) {
52          if(fldName.equals("query-plan"))
53              return new VarcharConstant(result);
54          else
55              throw new RuntimeException("is not query-plan");
56      }
```

在 ExpainScan 中的 getVal()函數中，我們先確認 field 是不是“query-plan”，如果是的話就回傳上面建好的 result，如果不是的話就 throw runtime exception。

```
115     @Override
116     public String toString() {
117         String c = p.toString();
118         String[] cs = c.split("\n");
119         StringBuilder sb = new StringBuilder();
120         sb.append("->");
121         sb.append("ProjectPlan (#blks=" + blocksAccessed() + ", #recs="
122                 + recordsOutput() + ")\n");
123         for (String child : cs)
124             sb.append("\t").append(child).append("\n");
125         return sb.toString();
126     }
```

在 ProjectPlan 的 toString()函數中，我們先呼叫下層 plan 的 toString()並以'\n'將它們分開，然後我們要寫的字串放進一個 StringBuilder 物件中，再將剛剛拿到的字串加上縮排，再 append 到 StringBuilder 中並回傳。

```
380     @Override
381     public String toString() {
382         String c = p.toString();
383         String[] cs = c.split("\n");
384         StringBuilder sb = new StringBuilder();
385         sb.append("->");
386         sb.append("SelectPlan pred:(" + pred.toString() + ") (#blks=" + blocksAccessed() + ", #recs="
387                 + recordsOutput() + ")\n");
388         for (String child : cs)
389             sb.append("\t").append(child).append("\n");
390         return sb.toString();
391     }
```

在 SelectPlan 中的 toString() 和 ProjectPlan 只差在我們加了 pred.toString() 到字串中，以提供更多資訊。

```
142     @Override
143     public String toString() {
144         String c1 = p1.toString();
145         String c2 = p2.toString();
146         String[] cs = (c2 + c1).split("\n");
147         StringBuilder sb = new StringBuilder();
148         sb.append("->");
149         sb.append("ProductPlan (#blks=" + blocksAccessed() + ", #recs="
150                 + recordsOutput() + ")\n");
151         for (String child : cs)
152             sb.append("\t").append(child).append("\n");
153         return sb.toString();
154     }
```

ProductPlan 因為有兩個子 plan，所以要呼叫子 plan 的 toString()各一次，並串在一起 split()，其他和 ProjectPlan 一樣。

```
100     @Override
101     public String toString() {
102         StringBuilder sb = new StringBuilder();
103         sb.append("->");
104         sb.append("TablePlan on (" + ti.tableName() + ") (#blks=" + blocksAccessed() + ", #recs="
105                 + recordsOutput() + ")\n");
106         return sb.toString();
107     }
```

TablePlan 因為是葉節點，因此不用呼叫下層的 toString()，直接回傳字串就可以了，另外我們也在字串中間插入了 table 的名稱。

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

GroupByPlan 中我們新增了 group 根據的 fields，其他都一致。

```java
@Override
public String toString() {
    String c = p.toString();
    String[] cs = c.split("\n");
    StringBuilder sb = new StringBuilder();
    sb.append("->");
    sb.append("SortPlan sort by:(");
    for (int i = 0; i < sortFlds.size(); i++) {
        sb.append(sortFlds.get(i) + " ");
        sb.append(sortDirs.get(i) ==  DIR_ASC ? "ASC" : "DESC");
        if (i < sortFlds.size()-1)
            sb.append(", ");
    }
    sb.append(") (#blks=" + blocksAccessed() + ", #recs=" + recordsOutput() + ")\n");
    for (String child : cs)
        sb.append("\t").append(child).append("\n");
    return sb.toString();
}
```

SortPlan 中我們新增了 sort 根據的 fields 以及排序的方向，格式是(field1 direction1, field2 direction2, …)，其他也都一致。

(二)Tests

```
SQL> EXPLAIN SELECT c_id, c_first FROM customer WHERE c_id<2;

query-plan
------------------------------------------------------------------------------
->ProjectPlan (#blks=30001, #recs=0)
        ->SelectPlan pred:(c_id<2.0) (#blks=30001, #recs=0)
                ->TablePlan on (customer) (#blks=30001, #recs=60000)
Actual #recs: 20

SQL> EXPLAIN SELECT d_zip, w_name FROM warehouse, district WHERE w_id=d_w_id;

query-plan
------------------------------------------------------------------------------
->ProjectPlan (#blks=43, #recs=40)
        ->SelectPlan pred:(w_id=d_w_id) (#blks=43, #recs=40)
                ->ProductPlan (#blks=43, #recs=40)
                        ->TablePlan on (warehouse) (#blks=2, #recs=2)
                        ->TablePlan on (district) (#blks=3, #recs=20)
Actual #recs: 40

SQL> EXPLAIN SELECT c_id, c_since FROM customer ORDER BY c_since;

query-plan
------------------------------------------------------------------------------
->SortPlan sort by:(c_since ASC) (#blks=236, #recs=60000)
        ->ProjectPlan (#blks=30001, #recs=60000)
                ->SelectPlan pred:() (#blks=30001, #recs=60000)
                        ->TablePlan on (customer) (#blks=30001, #recs=60000)
Actual #recs: 60000

SQL> EXPLAIN SELECT c_id, max(c_since) FROM customer GROUP BY c_id;

query-plan
------------------------------------------------------------------------------
->ProjectPlan (#blks=30000, #recs=931)
        ->GroupByPlan group by :(c_id) (#blks=30000, #recs=931)
                ->SortPlan sort by:(c_id ASC) (#blks=30000, #recs=60000)
                        ->SelectPlan pred:() (#blks=30001, #recs=60000)
                                ->TablePlan on (customer) (#blks=30001, #recs=60000)
Actual #recs: 3000
```