DB Assignment 3

Team members

廖庭萱 107062108
廖凰汝 107060017
楊善雅 107062211

Implementation

1. Praser.java
   我們在 queryCommand() 中，新增一個 Boolean isExplain 來判斷現在的 command 中有沒有 explain 這個關鍵字。預設值是 false，但假如我們有吃到 explain 關鍵字的話，則 isExplain 為 true。並且，在 return new QueryData 的地方，我們也會將 isExplain 回傳。

2. Lexer.java
   在這個檔案中，我們在 initKeywords() 裡新增一個關鍵字 “explain”。

3. QueryData.java
   因為從 Parser/queryCommand 傳回的 isExplain 會變成 QueryData 的 private variable，因此我們寫了一個 isExplain() function 來讓其他 class 可以得知現在指令中是否有 explain 關鍵字。在 toString function 中，如果 isExplain 為真，則在 result 裡增加 “explain”。

4. BasicQueryPlanner.java
   在 BasicQueryPlanner 中，我們透過 createPlan 來創建新的 Plan。在 Plantree 中由下而上，我們依序建立以下的 Plan：
   a. TablePlan：負責讀取一個 table 的資料。
   b. ProductPlan：當有多個 table 被讀取時，ProductPlab 會兩兩將所選 table 做 full join。
   c. SelectPlan：負責 SQL 指令中，有關 WHERE 語句的選擇。
   d. GroupByPlan：負責 SQL 指令中的 GROUP BY 語法，包含 COUNT、SUM、AVG、MIX、MAX 等各種 aggregation functions 的處理。
   e. ProjectPlan：負責 SQL 指令中，有關 SELECT 語句的篩選，可以選出使用者想要的 fields。
   f. SortPlan：負責 SQL 指令中的 ORDER BY 語法，處理有關資料排序的部分。會建立一個暫存的 table，裡面有預先排序好的資料，這樣一來每次取 next() 時就不用重新再比一次大小。
   g. ExplainPlan：這次作業實作的指令，當 SQL 的指令中有 EXPLAIN 出現時，則把 plan tree 的結構建立成一個 query-plan field。

5. ExplainPlan.java
   ExplainPlan 在所有的 plan 中是包在最終層的 Plan。在這裡我們傳入他上一層的 plan, 並 new 一個 schema, 加入 query-plan 這個 filed 存放我們的 explain string。另外, 為了要紀錄 plan tree, 這裏實作了 toString() 的 function。在這個 function 中我們呼叫上一層 plan 的 toString(), 就會得到所有下層的 explain string。接下來再透過上一層 plan 的 scan, 跑完所有 scan 紀錄 actual record 的數量。把所有下層的 explain string 以及 actual record append 起來就會是最終我們要的 explain string。open 一個 ExplainScan 的時候, 就會把這個 explain string 傳進去。

6. Plan
   為了回傳每一步預估的 accessed block, record 數量, 我們在各個 Plan 中都加入了toString() function, 有做新增的檔案包含:
   - ProductPlan.java
   - ProjectPlan.java
   - SelectPlan.java
   - TablePlan.java

   其中 TablePlan 為最初始創建的 plan, 紀錄預估的 block, record 數量, 在把 plan 傳到下一步時, 會再把預估數量 attend 在紀錄前端, 最後匯集成呼叫 Explain command 時會印出的資訊。
   其中的寫法都是利用 Plan class 中定義的 blocksAccessed() 和 recordsOutput() function 來取得預估的數量, 而在 TablePlan 中需要額外紀錄 table 的名稱, 所以另外呼叫 tableName(), 並且在串接前面的紀錄時會加上tab , 因此最後印出的資料就會以 intend 來保持呼叫時的順序。

7. ExplainScan.java
   在ExplainScan 裡我們傳進上一層的 scan, filed list 以及 explain string。
   a. beforeFirst(): 是要把指標指到 record 的前一筆, 在這裡我們就直接呼叫上一層 scan 的 beforeFirst 一路 recursive 下去就好。
   b. next(): 因為 explain string 只存在第一筆 record 的 query-plan 裡, 所以 next() 要判斷這次呼叫是不是第一次呼叫。只有第一次呼叫會回傳 True（把 first flag 也設成 false), 其餘則 回傳False。
   c. getVal(): 輸入一個 filed name， getVal 會回傳對應的資料。在 getVal 裡我們先判斷這個 Schema 有沒有這個 field, 如果有, 我們就直接回傳 explain string。

Experiment

- A query accessing single table with WHERE

```
SQL> EXPLAIN SELECT i_price, i_id FROM item WHERE i_price<50

query-plan
-----------------------------------------------------------------------
->ProjectPlan (#blks=6251, #recs=47908)
        ->SelectPlan (#blks=6251, #recs=47908)
                ->TablePlan on (item) (#blks=6251, #recs=100000)
Actual #recs: 49500
```

- A query accessing multiple tables with WHERE

```
SQL> EXPLAIN SELECT c_id, w_street_1 FROM warehouse, customer WHERE c_w_id=w_id

query-plan
-----------------------------------------------------------------------------
->ProjectPlan (#blks=15003, #recs=6971)
        ->SelectPlan (#blks=15003, #recs=6971)
                ->ProductPlan (#blks=15003, #recs=30000)
                        ->TablePlan on (warehouse) (#blks=2, #recs=1)
                        ->TablePlan on (customer) (#blks=15001, #recs=30000)
Actual #recs: 30000
```

- A query with ORDER BY

```
SQL> EXPLAIN SELECT w_name, w_tax FROM warehouse, district WHERE w_id=d_w_id ORDER BY w_tax

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

- A query with GROUP BY and at least one aggregation function

```
SQL> EXPLAIN SELECT COUNT(i_id) FROM item WHERE i_price<10 GROUP BY i_price

query-plan
-----------------------------------------------------------------------------------------------
->ProjectPlan (#blks=495, #recs=9)
        ->GroupByPlan: (#blks=495, #recs=9)
                ->SortPlan (#blks=495, #recs=7918)
                        ->SelectPlan (#blks=6251, #recs=7918)
                                ->TablePlan on (item) (#blks=6251, #recs=100000)
Actual #recs: 9
```

Worth Mentioning

- 我們發現 plan tree 出來的 record 數量跟 actual record 的數量真的會有一些誤差，而且每次重啟 server 時，預估的 records 就會跟上次預估得不一樣，即便我們下的指令是相同的。由次也可看出 Plan 真的是負責預估，並不代表實際的 record 數量。

- 我們在嘗試 *EXPLAIN SELECT c_id, h_amount FROM customer, history WHERE c_id=h_c_id AND c_id<10 ORDER BY h_date* 這個指令時，跑很久，猜想原因應該是 history 乘 customer 的這個 table 太大了，先做 product 再 select 時就會很久。改進的方法是如果我們先 select 再做 product，可能會快很多。