101062124 戴宏穎

Cloud DB Assignment 2

Part 1. 如何實作 Explain 的功能

根據我 trace code 以後的結果，當接到一個 SQL Query 的時候，會依照以下流程：

1. 把 SQL Query 傳給 Planner 的 createQueryPlan
2. 在 createQueryPlan 裡面會把 SQL Command 再傳給 Parser 和 Lexer 來解析 Query Command 並拿到 QueryData 。
3. 把 QueryData 傳進去 qPlanner (在這裡是 BasicQueryPlanner) 的 createPlan
4. 在 createPlan 裡頭就開始把 Parse 好的 Command 根據類別分別去產生出對應的 Plan 一個接一個得把它們串起來。當我實際上在跑指令時，在 Plan 也會去呼叫對應的 Scan 來完成任務。

為了實作 Explain 的需求，根據上面的流程，我對每一階段都做了一些修改：

1. 這邊沒有做更動，還是一樣單純的把 SQL Query 傳進 createQueryPlan 裡面。
2. 在這個階段，Parser, Lexer, QueryData 都有做修改：
    1. QueryData：
        A. 為了能讓我在之後判斷說是否為 explain 的指令，所以我增加了一個布林值 isExplainQuery 也加上對應的 set, get 的 API ，方便我來設定或取得狀態
    2. Lexer：
        A. 因為原本的 Lexer 並不認識 “explain” 這個關鍵字，要先加上這個關鍵字。
        B. 為了能讓我的 Parser 能拿到當下吃到的字，所以我新增了一個 API - getCurrentToken 來拿取。
    3. Parser：
        A. 因為必須透過 queryCommand 來驅動，所以我透過剛才新增的 getCurrentToken 來判斷說需不需要把 explain 這個關鍵字吃掉，如果沒有吃掉的話，會被 eatKeyword(“select”) 吃到，然後就壞掉了。
        B. 最後要產生 QueryData 的時候，把是否為 explain 指令的狀態也存進去，方便之後使用。
3. 進到 BasicQueryPlanner 的 createPlan ，這邊就開始要產生整個 Plan 和 Scan ，也是我們要把 query-plan 存下來的地方。除了更動的地方有點多外，還有加上幾個 class 與 API 來幫助我取得資料：
    1. createPlan in BasicQueryPlanner：
        A. 我先用一個 HashMap 來紀錄沿途遇到的 query-plan ，我是用 level 對應一個 String List 代表說在 level 的階層中有幾個 Plan 的 String。
        B. 在每個 Plan 被 new 出來的同時，我就透過 Plan 裡面的 API 把估計的 block 和 record 的數量拿出來，並轉成 String 存進 HashMap 中。
        C. 最後透過上個階段存下來的 isExplainQuery 來判斷，如果是 Explain 指令的話，那我就產生 query-plan 並在 Plan 加上 ExplainPlan 來把資訊送到 Client 端；如果只是單純的 Explain 指令，就不加入 query-plan 的資訊，讓他保持原本的行為。
    2. map2str in BasicQueryPlanner：
        A. 這個 helper function 適用來幫助我把 HashMap 裡面的 query-plan 轉成一條長字串，方便我放入 ExplainPlan 和 ExplainScan 中。
    3. GroupByPlan：
        A. 因為在 GroupByPlan 中有機會呼叫到 SortPlan ，我在 GroupByPlan 中增加了可以拿到 SortPlan 的 API 讓我可以把 SortPlan 的 query-plan 送回 BasicQueryPlanner 的 HashMap 中。
    4. ExplainPlan & ExplainScan：

A. 因為 Client 端拿到輸出的方式是透過 Scan 裡面的 getVal ，為了能不更動到 table 就能處理好 query-plan 的輸出。
B. 在這邊我增加了 ExplainPlan 和 ExplainScan，讓 getVal 實際上是把 query-plan 轉成一個 Constant String 傳回 Client 端。
C. ExplainPlan 實際上是一個空殼，他並沒有做任何行為，只是單純的能讓我們把 query-plan 存進來，之後轉交給 ExplainScan。
D. ExplainScan 則是把 query-plan 吃進來，當 Client 呼叫 next 的時候就把 select 的結果跑完，拿到實際的 record 數量，接著等待 Client 呼叫 getVal 時把 query-plan 加上 next 時算出的 record 數量合併，轉成常數字串送回 Client 端，完成任務。

4. ConsoleSQLInterpreter：
因為我們是寫死 query-plan 成一個字串傳回，原本的輸出方式並不符合我的需求，需要加上判斷來處理輸出結果。如果是 explain 指令，我就直接把 query-plan 印出，不需要印出 record 的結果；如果是 select 指令的話，就印出原本的 record 結果。

Part 2. 一些 Explain 的 query 和 output
1. Single table query：
   1. CREATE TABLE meow (id INT, name VARCHAR(10))
      1. 0 records processed
   2. INSERT INTO meow (id,name) VALUES (1, 'Pusheen1')
      1. 1 records processed
   3. SELECT name FROM meow WHERE id > 0
      1. name
      2. ----------
      3. Pusheen1
   4. INSERT INTO meow (id,name) VALUES (2,'Pusheen2')
      1. 1 records processed
   5. SELECT name FROM meow WHERE id > 0
      1. name
      2. ----------
      3. Pusheen1
      4. Pusheen2
   6. EXPLAIN SELECT name FROM meow WHERE id > 0
      1. query-plan
      2. --
      3. ->ProjectPlan (#blks=0, #recs=0.0)
      4. ->SelectPlan pred: (id>0.0) (#blks=0, #recs=0.0)
      5. ->TablePlan on (meow) (#blks=0, #recs=0.0)
      6.
      7. Actual #recs: 2
2. Multiple tables query：
   1. SELECT d_id FROM district, warehouse WHERE d_w_id = w_id AND d_id < 5
      1. d_id
      2. ------
      3. 1
      4. 2
      5. 3
      6. 4

2. EXPLAIN SELECT d_id FROM district, warehouse
   WHERE d_w_id = w_id AND d_id < 5
   1. query-plan
   2. --
   3. ->ProjectPlan (#blks=4, #recs=4.0)
   4.   ->SelectPlan pred: (d_w_id=w_id and d_id<5.0) (#blks=4, #recs=4.0)
   5.   ->ProductPlan (#blks=4, #recs=10.0)
   6.   ->TablePlan on (warehouse) (#blks=2, #recs=1.0)
   7.   ->TablePlan on (district) (#blks=2, #recs=10.0)
   8.
   9. Actual #recs: 4

3. Query with GROUP BY or ORDER BY：
   1. SELECT d_id FROM district, warehouse
      WHERE d_w_id = w_id AND d_id > 5 ORDER BY d_id DESC
      1.   d_id
      2. ------
      3.   10
      4.   9
      5.   8
      6.   7
      7.   6
   2. EXPLAIN SELECT d_id FROM district, warehouse
      WHERE d_w_id = w_id AND d_id > 5 ORDER BY d_id DESC
      1. query-plan
      2. --
      3. ->SortPlan (#blks=1, #recs=5.0)
      4.   ->ProjectPlan (#blks=4, #recs=5.0)
      5.   ->SelectPlan pred: (d_w_id=w_id and d_id>5.0) (#blks=4,
         #recs=5.0)
      6.   ->ProductPlan (#blks=4, #recs=10.0)
      7.   ->TablePlan on (warehouse) (#blks=2, #recs=1.0)
      8.   ->TablePlan on (district) (#blks=2, #recs=10.0)
      9.
      10.Actual #recs: 5

4. Query with at least one aggregation function：
   1. SELECT COUNT(d_id) FROM district, warehouse WHERE d_w_id = w_id GROUP BY w_id
      1. countofd_id
      2. -----------
      3.   10
   2. EXPLAIN SELECT COUNT(d_id) FROM district, warehouse WHERE d_w_id = w_id GROUP BY w_id
      1. query-plan
      2. --
      3. ->ProjectPlan (#blks=2, #recs=1.0)
      4.   ->GroupByPlan (#blks=2, #recs=1.0)
      5.   ->SortPlan (#blks=2, #recs=10.0)
      6.   ->SelectPlan pred: (d_w_id=w_id) (#blks=4, #recs=10.0)
      7.   ->ProductPlan (#blks=4, #recs=10.0)
      8.   ->TablePlan on (warehouse) (#blks=2, #recs=1.0)
      9.   ->TablePlan on (district) (#blks=2, #recs=10.0)
      10.
      11.Actual #recs: 1
   3. SELECT SUM(d_id), COUNT(d_id), AVG(d_id) FROM district, warehouse WHERE d_w_id = w_id GROUP BY w_id
      1.   avgofd_id   countofd_id   sumofd_id
      2. -------------------------------------
      3.   5.500000   10   55.000000
   4. EXPLAIN SELECT SUM(d_id), COUNT(d_id), AVG(d_id) FROM district, warehouse WHERE d_w_id = w_id GROUP BY w_id
      1. query-plan
      2. --
      3. ->ProjectPlan (#blks=2, #recs=1.0)
      4.   ->GroupByPlan (#blks=2, #recs=1.0)
      5.   ->SortPlan (#blks=2, #recs=10.0)
      6.   ->SelectPlan pred: (d_w_id=w_id) (#blks=4, #recs=10.0)
      7.   ->ProductPlan (#blks=4, #recs=10.0)
      8.   ->TablePlan on (warehouse) (#blks=2, #recs=1.0)
      9.   ->TablePlan on (district) (#blks=2, #recs=10.0)
      10.
      11.Actual #recs: 1

Part3. 其他想講的事情

本次的作業其實我沒有懂得很徹底，在誤打誤撞間把功能完成，最不了解的就是我們拿到的預測值是怎麼來的？他有什麼樣實質的運用與實際的含義？

這次我做的時候有動到 GroupByPlan 等類別，甚至加上一些他們本來沒有的 API ，感覺上好像破壞了程式設計的結構，這個部分可能來參考之後助教放出的解答來分析才能知道有什麼好一些的解法。