Team5 Assignment2 Report2

107062313 黃寶萱　107062117 李采蓉　107062317 陳怡汝

◈ Part1 : Implement vanillabench property

A. vanillabench.properties

1. 我們的 project 多新增了 TOTAL_UPDATE_COUNT

   我們參照 As2ReadItemParamGen.java 中讀取 TOTAL_READ_COUNT 的方式建立 TOTAL_UPDATE_COUNT，故在 vanillabench.properties 中新增 TOTAL_UPDATE_COUNT 方便其他檔案讀取。我們認為我們將 READ 和 UPDATE 分開紀錄的做法可以更好的維護、修改及使用 TOTAL_UPDATE_COUNT。

B. org.vanilladb.bench.benchmarks.as2.As2BenchConstants.java

   除了 READ_WRITE_TX_RATE 的 defaultValue 有異，其餘皆相同。

C. org.vanilladb.bench.benchmarks.as2.As2BenchTransactionType.java

   兩個 project 皆新增 UPDATE_ITEM_PRICE(true)，並無不同。

◈ Part2 : Implement RTE

A. org.vanilladb.bench.benchmarks.as2.rte.As2BenchmarkRte.java

1. 助教 project 中設定 flag 時有強制轉型成 int

   我們認為在判斷變數大小時，應確保型態一致，以降低出錯機率。所以以助教的方式完善地確認變數型態才是更好的寫法。

2. executor 寫法不同

   助教 project 是建立一個 TxParamGenerator 傳入不同 ParamGen 的 function 作為 As2BenchmarkTxExecutor 的參數，而我們 project 參照原本 As2BenchmarkRte()中 executor 的寫法直接在不同 case 時一次寫好 executor 並 return，過程結果並無太大不同。

B. org.vanilladb.bench.benchmarks.as2.rte. As2UpdateItemPriceParamGen.java

1. TOTAL_UPDATE_COUNT / WRITE_COUNT 的讀取

   我們參照 As2ReadItemParamGen.java 中讀取 TOTAL_READ_COUNT 的方式建立 TOTAL_UPDATE_COUNT，故在此檔案中藉由 BenchProperties.getLoader().getPropertyAsInteger()讀取變數。我們認為以我們的方式可以更好維護 TOTAL_UPDATE_COUNT。

2. 助教 project 有新增 UpdateItemPriceTxnParam.java

   助教在 UpdateItemPriceTxnParam.java 建立一個 parameter class，裡面存放 itemId 和 raise，並在 As2UpdateItemPriceParamGen.java 中以一組變數的形式存入 paramList，而我們 project 是直接將兩個變數依序存入 paramList。助教的方式能確保 itemId 和 raise 數值的正確以及是否為同一組 parameter，因此助教的寫法較為完善且正確。

◈ **Part3 : Implement JDBC**

A. org.vanilladb.bench.benchmarks.as2.rte.jdbc.As2BenchJdbcExecutor.java

   兩個 project 皆新增 case UPDATE_ITEM_PRICE，並無不同。

B. org.vanilladb.bench.server.param.as2.UpdateItemPriceProcParamHelper.java

   1. getUpdatedItemPrice 寫法不同

      助教的寫法是在 return 時做判斷有無超過 MAX_PRICE，而我們 project 的寫法是先判斷再 return，過程結果並無太大不同。另外助教在 return 時有強制轉型成 double，為降低錯誤，應以助教的方式完善地確認變數型態才是更好的寫法。

   2. 取得 parameter 的方式不同

      為方便變數讀取，助教有新增 UpdateItemPriceTxnParam.java，故在處理變數時是以一個 class 的兩個變數 itemId 和 raise 分別讀取，而在我們 project 是連續讀取兩個變數作為 itemId 和 raise，我們認為助教的寫法更好，才能確保變數的正確性。

◈ **Part4 : Implement Stored Procedure**

A. org.vanilladb.bench.benchmarks.as2.rte.jdbc.UpdateItemPriceTxnJdbcJob.java

   1. pars parameters：我們藉由 class UpdateItemPriceProcParamHelper()中的 function 來存取 update count、item id 與 item price 資料，而助教的作法是直接將資料存在 local itemIds[]與 raises[]中，並沒有透過 paramHelper()，代表不會動到 paramHelper()中的資料，我們認為助教的做法可以較好的將 JDBC client 端與 SP server 端的資料處理分開，確保資料的正確性。

2. SELECT、UPDATE：兩個 project 皆是利用 statement.executeQuery() 和 statement.executeUpdate()完成下 SELECT 和 UPDATE 指令的動作，因此並沒有相異處。

B. org.vanilladb.bench.server.procedure.as2.As2BenchStoredProcFactory.java
兩個 project 皆新增 case UPDATE_ITEM_PRICE，並無不同。

C. org.vanilladb.bench.server.procedure.as2.UpdateItemPriceTxnProc.java
1. 兩個 project 皆利用 getParamHelper()得到 update price 時需要用的參數。
2. SELECT part：在建立 planner 以及 scan 的部分，助教使用 VanillaDb.newPlanner().createQueryPlan()建立 planner，並呼叫 function open()得到對應的 scan 來完成 SELECT query，而我們這部分直接使用 class StoredProcedureHelper 中的 function executeQuery()，但實際上兩者的做法是一樣的，只是呼叫 function 的方式不同而已。
3. UPDATE part：這部分如同上述第 2 點，我們直接使用 class StoredProcedureHelper 中的 function executeUpdate()完成 UPDATE 的指令，而助教使用 VanillaDb.newPlanner().executeUpdate()。
4. 根據執行 UPDATE 指令回傳的結果，我們多加判斷當 result>0 (result 代表更新的 record 個數)時會利用 paramHelper getUpdateItemPrice()得到更新後的價格，並呼叫 setUpdateItemPrice()確定 array itemPrice[ ] 中的價格有被更改到，這部分我們認為可寫可不寫，因為每次當 client-side 要讀取價格資料時，實際上會直接從 storage 讀取，再將資料利用 setItemPrice()寫到 array itemPrice[ ]中，因此 client-side 拿到的資料一定會是正確的。

◈ **Part5 : StatisticMgr**

1. 統計 transection 的方式不同

助教在 outputDetailReport 中，順便紀錄了每一個 transaction 所需要的 response time 以及屬於哪一個 slot，在輸出新的 report 的時候只需查詢每個 time slot 記下的資料。

我們的作法則是在輸出新的 report 時，重新遍歷每個 transaction 並且計算每個 time slot 的統計結果，比起助教的方式更多了一次遍歷每個 transection 的步驟，因此我們認為助教的方式更有效率。

2. 統計數據紀錄資料結構不同

在記錄每一筆統計數據時，助教使用 TreeMap 的資料結構，裡面包了一個 Long 代表 timeslot 及 ArrayList 代表每個 time slot 中的所有 transaction，這樣的紀錄方式好處是可以對某一個 time slot 進行 random access，因此在需要 random access 的情況下更有效率。我們紀錄數據的資料結構則是用單一 ArrayList 記住單一 time slot 內所有 transaction，在每個 time slot 開始時清空目前 ArrayList 再次紀錄新的數據，相較於助教的作法我們減少了創建多個 ArrayList 所需要的時間成本，並且因為每次只記一個 time slot 內的數據，我們可以使用更少的空間，但同時也損失了助教可以 random access 所有 time slot 內所有統計資料的優點，因此這部分的優劣根據數據的使用情境會有所不同。

3. 數值計算方式不同

我們取 25th, median, 75th 資料的方式是直接取該 time slot 排序後的第 25、50、75 筆資料，助教的方式則是將數據分成偶數及奇數進行運算，因此在數值計算上助教的方式較為正確，我們的作法在數量為偶數的情況下存在一些誤差。而對於上述資料及 min、max 資料的取得，我們對資料進行排序，並且直接索引排序後的資料，時間複雜度為 O(1)，因此我們的作法有更高的效能。