Introduction to Database System - Assignment 2 Report

Team 22 - 108020017 黃珮綺, 108020021 賴柏翰

Phase 2 Report

1. Implementation Differences

- Shared classes
    - vanillabench.properties
    - As2BenchTransactionType
- Client-side classes
    - UpdatePriceParamGen
    - UpdatePriceJdbcJob
    - As2BenchJdbcExecutor
    - As2BenchmarkRte
- Server-side classes
    - UpdatePriceProcParamHelper
    - UpdatePriceProc
    - As2BenchStoredProcFactory
- Statistic Analysis
    - StatisticMgr

2. Implementation Details

UpdatePriceParamGen

In our implementation, `paramList` is an `ArrayList` containing the number of items to be updated, followed by a list of pairs consisting of an index and a price raise. In the solution, `paramList` is initialized as a `LinkedList`, and it consists of the number of items to be updated, followed by a list of `UpdateItemPriceTxnParam` objects. The class `UpdateItemPriceTxnParam` is defined in another file, and each instance stores the ID and the raise of the item.

Since we only add objects to the end of `paramList`, the time complexity of `add` in `ArrayList` and `Linkedlist` are both O(1). Also, `paramList` will be turned into an `Array` before it is returned, so there is no need of considering the time complexity of `get` method. Hence, both `ArrayList` and `Linkedlist` are compatible.

Practically, our implementation is easier in generating and extracting `paramList`. However, the time spent on extracting `paramList` with `UpdateItemPriceTxnParam` objects might be slightly faster than that without. This is because the length of `paramList` with plain numbers will be twice as many as the length of `paramList` whose values are packed in `UpdateItemPriceTxnParam` objects. Furthermore, it will be more complex if more values are needed to be generated in former one. In this case, the implementation of the solution is better.

UpdatePriceProcParamHelper

The implementations in our repository and in the solution are similar. The only difference is the way of extracting the parameter list. The comparison of both implementation is described in the above section.

UpdatePriceJdbcJob

The main difference between our implementation and the solution is the method of parsing parameter lists. In the solution, it directly extracts the parameter list in the same file. Yet, we leave this task to be handled by the provided class `UpdatePriceProcParamHelper` whose implementation is detailed in the previous report.

As the way of parsing parameter lists does not change whether it is handled directly in the same file nor in `UpdatePriceProcParamHelper`, the performance may not differ in theory. We also underwent an experiment in practice. Based on the result in our environment, there is no significant difference between these two methods.

UpdatePriceProc

For `SELECT` query, in the solution, a `Plan` object is created with the given query, and then a `Scan` object will open the optimized plan tree to execute the query. In our implementation, this is simply done by calling `StoredProcedureHelper.executeQuery()`, which also creates a `Plan` and returns a `Scan` following the plan.

For `UPDATE` query, the solution directly uses the `executeUpdate` function in `VanillaDb.newPlanner` to update the price, while, in our implementation, we call the same function via `StoredProcedureHelper.executeUpdate()`.

These two implementations would be of the same performance, although using a `StoredProcedureHelper` can simplify the code.

StatisticMgr

The implementation in the solution handles the case that the timeslot value is `NULL`, which is not considered in our implementation. Additionally, we uses the "round down" principle to calculate the 25th, 50th, and 75th percentile, while in the solution the principle is "round to nearest" that ensures the precision.

Reference:

Source Code