PROJECT 2: VISUAL EXPLORATION OF QUERY EXECUTION PLANS

CE4031/CZ4031 DATABASE SYSTEM PRINCIPLES

TOTAL MARKS: 100

Due Date: Nov 16, 2018; 5 PM

A DBMS query optimizer executes a query execution plan (QEP) to process a query. This plan is typically displayed as a tree-structure by your DBMS software. However, such structure can be large especially for complex SQL, typically has a very low degree of interactivity in most DBMS, and does not clearly show how different parts of the QEP are related to the different components of the corresponding SQL query. Furthermore, it does not visually show various interesting features of a plan (e.g., which part is the most time consuming one?). Hence, the goal of this project is to build a tool that enables **interactive visual exploration of QEPs** for SQL queries. Specifically, given a query Q and its executed plan P, your goal is to (a) automatically correlate components of Q and P using various color codes and (b) enable visual and interactive exploration of P (what exploration features you should support is open-ended…so you need to be creative ☺). To this end, you should do the followings:

- Design and implement an algorithm that takes as input a SQL query and its execution plan, and realizes the aforementioned goals. You must ensure generality of the solution (i.e., it should handle a wide variety of queries and query plans across different database instances). You should not hard code components of a specific SQL query or QEP in your code. Your algorithm should automatically extract various features from any SQL query and correlate them with the corresponding QEP. Similarly, your exploration algorithm should be able to handle any QEP across different database instances.
- A user-friendly, graphical user interface (GUI) to enable the aforementioned goals.

You can use **Java or Python** as the host language for your project. The DBMS allowed in this project is **PostgreSQL** (freely downloadable)**.** You should use the following two database instances to demonstrate your project:

- The relational database instance and SQL queries in **Project 1**
- A relational database instance and benchmark SQL queries using the **TPC-H benchmark** dataset (see *Appendix A*).

Your submission should include the followings:

- Hard copy of the detailed description of the algorithm (with examples) used to address the problem.
- Hard copy of the source code.
- Submit a softcopy of the program through NTULearn. Note that we will execute the software to check its correctness on a **different** database to check for the generality of the solution. You may attach an installation guide to ensure that your software can be run successfully.

Note: All hard copies need to be submitted to the Software Project Lab by 5 PM latest on Nov 16, 2018. Late submission will be penalized.

We may randomly select one or more groups to demonstrate their project subsequently.

Appendix A

Steps to create TPC-H database instance in PostgresSQL

1) Go to http://www.tpc.org/tpc_documents_current_versions/current_specifications.asp and download TPC-H_Tools_v2.17.3.zip. Note that the version may differ as the tool may have been updated by the developer.
2) Unzip the package. You will find a folder "dbgen" in it.
3) To generate an instance of the TPC-H database:
    - Open up tpch.vcproj using visual studio software.
    - Build the tpch project. When the build is successful, a command prompt will appear with "TPC-H Population Generator <Version 2.17.3>" and several *.tbl files will be generated. You should expect the following .tbl files: customer.tbl, lineitem.tbl, nation.tbl, orders.tbl, part.tbl, partsupp.tbl, region.tbl, supplier.tbl
    - Save these .tbl files as .csv files
    - These .csv files contain an extra "|" character at the end of each line. These "|" characters are incompatible with the format that PostgreSQL is expecting. Write a small piece of code to remove the last "|" character in each line. Now you are ready to load the .csv files into PostgreSQL
    - Open up PostgreSQL. Add a new database "TPC-H".
    - Create new tables for "customer", "lineitem", "nation", "orders", "part", "partsupp", "region" and "supplier"
    - Import the relevant .csv into each table. Note that pgAdmin4 for PostgreSQL (windows version) allows you to perform import easily. You can select to view the first 100 rows to check if the import has been done correctly.