Assignment 4: Query Processing

CSE 511: Data Processing at Scale - Spring 2023

Available: 03/21/2023 **Due Date**: 03/29/2023 11:59pm **Points:** 100

Introduction

The last assignment shows how to use Fragmentation in different methods within PostgreSQL. The task in this assignment is to build a simplified query processor that accesses data from the partitioned rating tables there.

Problem Statement

We have already provided the code that creates the `subreddits` table and partitioned tables as well for you already. The same environment will be maintained in the grading environment as well where you only need to perform implementations on the queries and nothing else. For this particular assignment, we have divided the problem statement into two high-level steps. Further information about the provided functions has also been made available in the `assignment4.py` file.

Step 1: Range Query (50 pts)

Below are the steps you need to follow to fulfil this assignment:
Implement a Python function `RangeQuery()`

1. It takes the following inputs:
    a. partition table name
    b. UTCMinValue
    c. UTCMaxValue
    d. Save table name
    e. connection
2. Please note that the `RangeQuery()` needs to use the parent partition tables (refer to assignment3 document for parent partition table).
3. `RangeQuery()` then returns all tuples for which the `created_utc` value is larger than `UTCMinValue` and less than or equal to `UTCMaxValue`. i.e. (UTCMinValue, UTCMaxValue].
4. The returned tuples **MUST** be stored in ascending order based on created_utc in a new **table**, the name of the table will be supplied by the variable, `save_table_name`

Step 2: Point Query (50 pts)

Implement a Python function `PointQuery()`

1. It takes the following inputs:
    a. partition table name,
    b. UTCValue.
    c. Save table name
    d. connection
2. Please note that `PointQuery()` needs to use the parent partition tables (refer to assignment3 document for parent partition table).
3. `PointQuery()` then returns all tuples for which the rating value is equal to UTCValue.
4. The returned tuples **MUST** be stored in a new **table**, the name of the table will be supplied by the variable, `save_table_name`

How to work on the assignment and test your code

- You can work choose to work on your local setup, or to use the docker image.
- PostgreSQL 14 is already present (but not running) in the docker image. Find the configurations below
    - username: postgres
    - password: postgres
- Meta-data table in the database is allowed.
- You are not allowed to modify the data file on disk.
- Test your code using the tester.py file. Run it using `python3 tester.py`

Grading

- The assignment is **auto-graded**.
- You will use the following files within your assignment. These files are used to test your code
    - `tester.py`: Test your code using this tester. Run it using "`python3 tester.py`".
    - `test_helper.py`: Provides tester.py with invaluable functions!
    - `assignment4.py`: File to Implement the interface. **DO NOT** change any other file.

- Make sure that your code runs on the docker container provided! That environment will allow for a partial reproduction of the grading environment. ***In case your script fails to run in the grading environment, you will be graded 0.***

Submission Requirements & Guidelines

- This is an **individual** assignment
- Maintain your code on the **provided private GitHub repository for assignment-4** in the [SPRING-2023] [CSE511] Data Processing at Scale Organization). Make sure you don't use any other repository.

- **What to submit on canvas?**
    - One python file (`assignment4.py`)
    - You **MUST** name your .python file as `assignment4.py`

Note:

- You **MUST** maintain the code on the private Github repository and also make necessary submissions on the canvas.
- Failure to miss any one of them and you will be awarded 0 grade points.
- We will use the submissions from canvas to grade your submissions and check your Github for valid code submissions commits.

Submission Policies

- Late submissions will *absolutely not* be graded (unless you have verifiable proof of emergency). It is much better to submit partial work on time and get partial credit than to submit late for no credit.
- Every student needs to *work independently* on this exercise. We encourage high-level discussions among students to help each other understand the concepts and principles. However, a code-level discussion is prohibited, and plagiarism will directly lead to the failure of this course. We will use anti-plagiarism tools to detect violations of this policy.