Ira A. Fulton Schools of Engineering
Arizona State University
Engineering | School of Computing, Informatics, and Decision Systems Engineering

CSE 511: Data Processing at Scale

Query Processing

Introduction:

The required task is to build a simplified query processor that accesses data from the partitioned ratings table.

Input Data

Same as in Assignment 1 (i.e. ratings.dat file).

For more information on Psycopg, click here. Psycopg is the PostgreSQL database adapter for Python.

Requirements:

Below are the steps you need to follow to complete this assignment:

RangeQuery() -

- Implement a Python function RangeQuery that takes as input: (1) Ratings table stored in PostgreSQL, (2) RatingMinValue, (3) RatingMaxValue, and (4) openconnection
    - Please note that the RangeQuery would not use ratings table but it would use the range and round robin partitions of the ratings table.
- RangeQuery() then returns all tuples for which the rating value is larger than or equal to RatingMinValue and less than or equal to RatingMaxValue.
- The returned tuples should be stored in a text file, named RangeQueryOut.txt (in the same directory where Interface.py is present). Each line of the file should represent a tuple that has the following format:
    - Example:

      PartitionName, UserID, MovieID, Rating
      RangeRatingsPart0,1,377,0.5
      RoundRobinRatingsPart1,1,377,0.5

In these examples, PartitionName represents the full name of the partition (such as RangeRatingsPart1 or RoundRobinRatingsPart4) in which the output tuple resides.

Note: Please use ',' (COMMA, no space character) as delimiter between PartitionName, UserID, MovieID, and Rating.

PointQuery() -

- Implement a Python function PointQuery that takes as input: (1) Ratings table stored in PostgreSQL, (2) RatingValue, and (3) openconnection
    - Please note that the PointQuery would not use ratings table but it would use the range and round robin partitions of the ratings table.
- PointQuery() then returns all tuples for which the rating value is equal to RatingValue.

RatingValue

- The returned tuples should be stored in a text file, named PointQueryOut.txt (in the same directory where Interface.py is present) such that each line represents a tuple that has the following format such that PartitionName represents the full name of the partition (i.e. RangeRatingsPart1 or RoundRobinRatingsPart4, etc.) in which this tuple resides.
    - Example:

      PartitionName, UserID, MovieID, Rating
      RangeRatingsPart3,23,459,3.5
      RoundRobinRatingsPart4,31,221,0

      Note: Please use ',' (COMMA) as delimiter between PartitionName, UserID, MovieID and Rating.

MovieID and Rating

- Please use the function signature exactly as mentioned in Interface.py.

Naming Conventions:

The naming convention is to be strictly followed:

- Database name: dds_assignment
- Name of rating table: ratings
- Postgres user name: postgres
- Postgres password: 1234
- Name of the Range Query output file: RangeQueryOut.txt

- Name of the Point Query output file: PointQueryOut.txt

How to Use Tester.py:

Implement your functions in Interface.py and once done, use the tester again to generate the output.

DO NOT CHANGE tester.py and Assignment1.py. Changing any of these would cause problems and the system will stop working, and may lead to deduction of marks.

PLEASE KEEP IN MIND, this tester is just for your help. For grading purposes, an additional set of test cases would be used. It will try to break your code. It is imperative that you provide the functionalities accordingly, so that it handles all possible scenarios.

Instructions for Assignment:

Please follow these instructions closely or else points will be deducted.

1. Please follow the function signature as provided in the Interface.py.
2. Please use the same database name, table name, user name, and password as provided in the assignment to keep it consistent.
3. Please make sure to run the provided tester and make sure there is no indentation error. In case of any compilation error, 0 marks will be given.
4. Any print function you add to your Interface.py will be suppressed in the grader feedback.

Submission:

Only submit the Interface.py file. Do not change the file name. Do not put it into a folder or upload a zip file.

We provide a virtual machine that has the testing environment and an installed PostgreSQL.

OS username: user

OS password: user

Postgres username: postgres

Postgres password: 1234

You can download it and use any VM software such as VirtualBox to import it:
https://drive.google.com/file/d/1EBlmGZmqDQ8XGTuiHPoqP7XE2ZykAXkX/view?usp=sharing

You will use the following files within your assignment. These files are used to test your Interface.py

1. tester.py: Test your interface.py using this tester. Run it using "python tester.py".
   tester.py
2. test_data.txt: Some test data
   test_data.txt
3. Interface.py: Implement the interface in Interface.py
   Interface.py
4. Assignment1.py: The correct answer of Assignment 1 with slight modifications
   Assignment1.py

Feedback:

There are ten test cases for a total of 20 points, so each test case is worth 2 points. We will run five test cases for "RangeQuery()" and five test cases for "PointQuery()". **If one of the functions fails, you will see the corresponding error logs that indicate *where* the error occurred.**

Test cases are not executed in parallel. You have to unlock all previous test cases to execute the current test case. For example, to execute test case 2 your code must be able to successfully execute test case 1. The test case number in the feedback will provide you an indication of which part of the submission caused the error.