Ira A. Fulton Schools of Engineering
Arizona State University

CSE 511: Data Processing at Scale

Assignment 4: SQL Query Processing

Purpose

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

Objectives

Learners will be able to:

- Link a database in a Python file and then write queries in the Python file to perform certain operations.
- Write code in Python files to observe how round-robin and range partitions function in practice.
- Query the range and round-robin partitions to fetch the required rows.

Technology Requirements

- Python 3.5
- PostgreSQL 9.5
- Psycopg2 2.7.4

Assignment Description

Input Data:

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

Review the psycopg.org website resource for more information on Psycopg. Psycopg is the PostgreSQL database adapter for Python.

Directions

Please review the **videos and additional deals regarding Assignment 3** before beginning. These are located in your course *Welcome and Start Here* section.

- Assignment 4: Query Processing Introduction Video
- Assignment 4: Query Processing Submission and Feedback Video
- Metadata Table in Assignments 3 and 4

Note: Project details in the Overview Document may have been updated since the recording of the videos, so some directions or items may not match perfectly. Please follow the Overview Document's directions to complete your work correctly.

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:

- Follow the function signature as provided in the Interface.py.
- Use the same database name, table name, user name, and password as provided in the assignment to keep it consistent.
- 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.
- Any print function you add to your Interface.py will be suppressed in the grader feedback.

Submission Directions for Assignment Deliverables

Only submit the **Interface.py file**. Do not change the file name. Do not put it into a folder or upload a zip file. **Use Python 3.5.x version.**

You will use the following files within your assignment (attached in Coursera's Assignment Overivew page). 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".
2. test_data.txt: Some test data
3. Interface.py: Implement the interface in Interface.py
4. Assignment1.py: The correct answer of Assignment 1 with slight modifications

When you are ready to submit:

1. Go to “**Programming Assignment: Assignment 4: Query Processing**”.
2. Click on the "**My submissions**" tab (located at the top of the page under the deadline date).
3. Click on “**Create submission**.”
4. Upload one file for the assignment and click “**Submit**.”
    a. If there is an error in your assignment, you may make corrections and resubmit.

Evaluation

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.**

The test cases are executed in a simultaneous manner. If you pass any 2 of the 5 test cases, you would receive 40% of the total marks.

Common Errors:

1. ‘Module’ object has no attribute 'RangeQuery’, ’PointQuery’ (Wrong assignment submission).
2. Error in RangeQuery: (rangeratingspart1,1,355,2.0) not found in your result for range query [1.5,3.5]!
3. No checks or constraints found for range query.
4. Error in PointQuery: (rangeratingspart1,1,355,2.0) not found in your result for point query 2!
5. Error in PointQuery: (roundrobinratingspart1,1,185,4.5) not found in your result for point query 4.5
6. Undefined global variables throw an error in Python.
7. Modifications in the function signature will lead to errors and fail test cases.
8. Error in RangeQuery: (partitionname,userid,movieid,rating) should not exist for range query [1.5,3.5], PointQuery: (partitionname,userid,movieid,rating) should not exist for point query 0.5!

Learner Checklist

Prior to submitting, read through the Learner Checklist to ensure you are ready to submit your best work.

- ☐ Did you title your file correctly and convert it into a single **Interface.py file**?
- ☐ Did you answer all of the questions to the best of your ability?
- ☐ Did you make sure your answers directly address the prompt(s) in an organized manner that is easy to follow?
- ☐ Did you proofread your work?