Ira A. Fulton Schools of Engineering
Arizona State University

CSE 511: Data Processing at Scale

Query Processing Assignment

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 3.10
- PostgreSQL 16
- Psycopg2 2.9.9

Assignment Description

Important: You must complete the Data Fragmentation Assignment before working on this assignment. The deliverables for this assignment are dependent on the solutions from your previous submission.

Optional: Review the **Additional Resource: Metadata Table** page for clarifying information but it is not required to complete the assignment. This is located in your course *Welcome and Start Here* module.

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.

Input Data:

Same as in Data Fragmentation Assignment (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

Assignment Files

You will use the following files within your assignment (attached in the Project Overviews and Resources page):

1. Interface.py: Implement the interface in **Interface.py**
2. Fragmentation.py: The correct answer of the Data Fragmentation Assignment with slight modifications
3. interfaceTester.py: Test your **Interface.py** using this tester. Run it using "python interfaceTester.py"
4. testHelper.py: Put this one together with tester.py
5. test_data.dat: Some test data

Assignment Directions

Follow the steps to fulfill 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
    - range_part0,1,377,0.5
    - rrobin_part1,1,377,0.5

In these examples, PartitionName represents the full name of the partition (such as range_part1 or rrobin_part4) 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.
- 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. range_part1 or rrobin_part4, etc.) in which this tuple resides.
- Example:
    - PartitionName, userid, movieid, rating
    - range_part3,23,459,3.5
    - rrobin_part4,31,221,0

Note: Please use ',' (COMMA) as delimiter between PartitionName, userid, 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
- PartitionNames are in lowercase.

How to Use interfaceTester.py:

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

Do not change interfaceTester.py and Fragmentation.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

You must submit your Query Processing Assignment deliverable through Gradescope. Carefully review submission directions outlined in this overview document in order to correctly earn credit for your work. Learners may not email or use other means to submit any assignment or project for review, including feedback, and grading.

The Query Processing Assignment includes one (1) deliverable:

- **Python File**: Submit only the **Interface.py file**, do not change the file name, and do not put it into a folder or upload a zip file.

Submitting to Gradescope

Your submission will be reviewed by the course team and then, after the due date has passed, your score will be populated from Gradescope into your Canvas grade.

1. Go to the Canvas Assignment, "**Submission: Query Processing Assignment**".
2. Click the "**Load Submission…in new window**" button.
3. Once in Gradescope, select the assignment titled "**Submission: Query Processing Assignment**" and a pop-up window will appear.
4. In the pop-up,
    a. Submit a single Python file "**Interface.py**".
    b. Click "**Upload**" to submit your work for grading.
5. You will know you have completed the assignment when feedback appears for each test case with a score.
6. If needed: to resubmit the assignment in Gradescope:
    a. Click the **"Resubmit"** button on the bottom right corner of the page and repeat the process from Step 3.

Evaluation

There are ten (10) test cases. 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 four (4) of the ten (10) 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: (range_part1,1,355,2.0) 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: (range_part1,1,355,2.0) not found in your result for point query 2!
5. Error in PointQuery: (rrobin_part1,1,185,4.5) 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?