Ira A. Fulton Schools of Engineering
Arizona State University

CSE 511: Data Processing at Scale

Data Fragmentation Assignment

Purpose

The required task is to simulate data partitioning approaches on top of an open-source relational database management system (i.e., PostgreSQL).

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.

Technology Requirements

- Python 3.10
- PostgreSQL 16
- Psycopg2 2.9.9

Assignment Description

You must generate a set of Python functions that load the input data into a relational table, partition the table using different horizontal fragmentation approaches, and insert new tuples into the right fragment.

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:

The input data is a Movie Rating dataset collected from the MovieLens website (http://movielens.org). The raw data is available in the file ratings.dat.

The ratings.dat file contains 10 million ratings and 100,000 tag applications applied to 10,000 movies by 72,000 users. Each line of this file represents one rating of one movie by one user, and has the following format:

userid::movieid::rating::timestamp

Ratings are made on a 5-star scale, with half-star increments. Timestamps represent seconds since midnight Coordinated Universal Time (UTC) of January 1, 1970. A sample of the file contents is given below:

1::122::5::838985046

1::185::5::838983525

1::231::5::838983392

Directions

Assignment Files

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

1. Fragmentation.py: Implement the interface in **Fragmentation.py**
2. fragmentationTester.py: Test your **Fragmentation.py** using this tester.
3. testHelper.py: Put this one together with fragmentationTester.py
4. test_data.dat: Some test data

Assignment Directions

Follow the steps to fulfill this assignment:

1. Install PostgreSQL 16 if you have not already installed it.

2. Download ratings.dat file from the MovieLens website: http://files.grouplens.org/datasets/movielens/ml-10m.zip You can use partial data from ratings.dat for testing. You do not need to test the entire dataset.

3. Implement a Python function **loadRatings()** that takes a file system path that contains the ratings.dat file as input. **loadRatings()** then load the ratings.dat content into a table (saved in PostgreSQL) named 'ratings' that has the following schema:

   userid(int) – movieid(int) – rating(float)

4. Implement a Python function **rangePartition()** that takes as input: (1) the 'ratings' table stored in PostgreSQL and (2) an integer value N; that represents the number of partitions. **rangePartition()** then generates N horizontal fragments of the 'ratings' table and stores them in PostgreSQL. The algorithm should partition the 'ratings' table based on N uniform ranges of the rating attribute.

5. Implement a Python function **roundRobinPartition()** that takes as input: (1) the 'ratings' table stored in PostgreSQL and (2) an integer value N; that represents the number of partitions. The function then generates N horizontal fragments of the 'ratings' table and stores them in PostgreSQL. The algorithm should partition the 'ratings' table using the round-robin partitioning approach (explained in class).

6. Implement a Python function **roundrobininsert()** that takes as input: (1) 'ratings' table stored in PostgreSQL, (2) userid, (3) movieid, (4) rating. **roundrobininsert()** then inserts a new tuple to the 'ratings' table and the right fragment based on the round-robin approach.

7. Implement a Python function **rangeinsert()** that takes as input: (1) 'ratings' table stored in Postgresql (2) userid, (3) movieid, (4) rating. **rangeinsert()** then inserts a new tuple to the 'ratings' table and the correct fragment (of the partitioned ratings table) based upon the Rating value.

8. Implement function **deletepartitions()** for your testing convenience. It will not be graded.

Frequently Asked Questions:

- Partition numbers start from 0, if there are 3 partitions then **range_part0**, **range_part1**, **range_part2** are partition table names for range partitions, and **rrobin_part0**,**rrobin_part1**,**rrobin_part2** are partition table names for round robin partitions.
- Do **not** change partition table names prefix given in fragmentationTester.py
- Do not hard code input filename
- Do not hardcode the database name

- Table schema should be equivalent to what has been described in point 3
- **Use Python 3.10 version**

Partitioning Questions:

The number of partitions here refers to the number of tables to be created. For rating values in [0, 0.5, 1, 1.5, 2, 2.5, 3, 3.5, 4, 4.5, 5].

- Case N = 1
  - One table containing all the values.
- Case N = 2, Two tables
  - Partition 0 has values [0,2.5]
  - Partition 1 has values (2.5,5]
- Case N = 3, Three tables
  - Partition 0 has values [0, 1.67]
  - Partition 1 has values (1.67, 3.34]
  - Partition 2 has values (3.34, 5]

Uniform ranges mean a region is divided uniformly. I hope the example gives a clear picture.

Assignment Tips:

1. Do not use global variables in your implementation. A metadata table in the database is allowed.
2. You are not allowed to modify the data file on disk.
3. Two insert functions can be called many times at any time. They are designed for maintaining the tables in the database when insertions happen.
4. Any print function you add to your Fragmentation.py will be suppressed in the grader feedback.

Submission Directions for Assignment Deliverables

You are given an unlimited attempts to submit your best work. The number of attempts is given to anticipate any submission errors you may have in regards to properly submitting your best work within the deadline (e.g., accidentally submitting the wrong paper). It is not meant for you to receive multiple rounds of feedback and then one (1) final submission. Only your most recent submission will be assessed.

You must submit your Data Fragmentation 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 Data Fragmentation Assignment includes one (1) deliverable:

- **Python File**: Submit only the **Fragmentation.py**, 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: Data Fragmentation Assignment**".
2. Click the "**Load Submission…in new window**" button.
3. Once in Gradescope, select the assignment titled "**Submission: Data Fragmentation Assignment**" and a pop-up window will appear.
4. In the pop-up,
    a. Submit a single python file "**Fragmentation.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 a total of five (5) test cases. We will test your "loadRatings()", "rangePartition()", "rangeinsert()", "roundRobinPartition()", and "roundrobininsert()" sequentially, and **if one of the functions fails, you will see the corresponding .sql error logs that indicate where the error occurred.**

The test cases are executed simultaneously. If you pass any two (2) of the five (5) test cases, you will receive 40% of the total marks.

Common Errors:

1. Error in roundRobinPartition(): Range partitioning not done properly.
2. Error in roundrobininsert(): Round robin insert failed! - Couldn't find (100, 1, 3) tuple in rrobin_part0 table.
3. Error in loadRatings(): relation 'ratings' does not exist.
4. Error in rangePartition(): Range partitioning not done properly.
5. Error in rangeinsert(): Range insert failed! Couldnt find (100, 2, 3) tuple in the range_part2 table.
6. Exceptions in your Fragmentation.py with the following error messages: expected an indented block, No module named ‘pandas’ (Python Errors)
7. Errors in SQL Statements.
8. Incomplete assignments, missing code blocks, and poorly written queries.

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 **Fragmentation.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?