Bài tập lớn

Nhiệm vụ yêu cầu là mô phỏng các phương pháp phân mảnh dữ liệu trên một hệ quản trị cơ sở dữ liệu quan hệ mã nguồn mở (ví dụ: PostgreSQL hoặc MySQL). Mỗi nhóm sinh viên phải tạo một tập các hàm Python để tải dữ liệu đầu vào vào một bảng quan hệ, phân mảnh bảng này bằng các phương pháp phân mảnh ngang khác nhau, và chèn các bộ dữ liệu mới vào đúng phân mảnh.

Dữ liệu đầu vào
Dữ liệu đầu vào là một tập dữ liệu đánh giá phim được thu thập từ trang web MovieLens (http://movielens.org). Dữ liệu thô có trong tệp `ratings.dat`.

Tệp `ratings.dat` chứa 10 triệu đánh giá và 100.000 thẻ được áp dụng cho 10.000 bộ phim bởi 72.000 người dùng. Mỗi dòng trong tệp đại diện cho một đánh giá của một người dùng với một bộ phim, và có định dạng như sau:
`UserID::MovieID::Rating::Timestamp`

Các đánh giá được thực hiện trên thang điểm 5 sao, có thể chia nửa sao. Dấu thời gian (Timestamp) là số giây kể từ nửa đêm UTC ngày 1 tháng 1 năm 1970. Ví dụ nội dung tệp:

```
1::122::5::838985046
1::185::5::838983525
1::231::5::838983392
```

Nhiệm vụ yêu cầu
Dưới đây là các bước bạn cần thực hiện để hoàn thành bài tập:

1. Tải về máy ảo có môi trường giống với máy chấm điểm. Bạn có thể dùng máy của mình, nhưng không đảm bảo mã của bạn sẽ chạy đúng trên máy chấm điểm. Nếu bạn dùng máy ảo được cung cấp thì bỏ qua Bước 2.
   Cấu hình máy ảo: Python 3.12.x, OS: Ubuntu hoặc Windows 10.
2. Cài đặt PostgreSQL hoặc MySQL.
3. Tải tệp `rating.dat` từ trang MovieLens:
   http://files.grouplens.org/datasets/movielens/ml-10m.zip
   Bạn có thể sử dụng dữ liệu một phần để kiểm tra. Một tệp dữ liệu kiểm tra được cung cấp trong repository này.
4. Cài đặt hàm Python `LoadRatings()` nhận vào một đường dẫn tuyệt đối đến tệp `rating.dat`.
   `LoadRatings()` sẽ tải nội dung tệp vào một bảng trong PostgreSQL có tên **Ratings** với schema sau:
   - `UserID` (int)
   - `MovieID` (int)
   - `Rating` (float)
5. Cài đặt hàm Python `Range_Partition()` nhận vào: (1) bảng Ratings trong PostgreSQL (hoặc MySQL) và (2) một số nguyên N là số phân mảnh cần tạo.
   `Range_Partition()` sẽ tạo N phân mảnh ngang của bảng Ratings và lưu vào PostgreSQL.
   Thuật toán sẽ phân chia dựa trên N khoảng giá trị đồng đều của thuộc tính Rating.

6. Cài đặt hàm Python `RoundRobin_Partition()` nhận vào: (1) bảng Ratings trong PostgreSQL (hoặc MySQL) và (2) một số nguyên N là số phân mảnh cần tạo.
   Hàm sẽ tạo N phân mảnh ngang của bảng Ratings và lưu chúng trong PostgreSQL (hoặc MySQL), sử dụng phương pháp phân mảnh kiểu **vòng tròn (round robin)** (đã được giải thích trong lớp).
7. Cài đặt hàm Python `RoundRobin_Insert()` nhận vào: (1) bảng Ratings trong PostgreSQL, (2) `UserID`, (3) `ItemID`, (4) `Rating`.
   `RoundRobin_Insert()` sẽ chèn một bộ mới vào bảng Ratings và vào đúng phân mảnh theo cách round robin.
8. Cài đặt hàm Python `Range_Insert()` nhận vào: (1) bảng Ratings trong PostgreSQL (hoặc MySQL), (2) `UserID`, (3) `ItemID`, (4) `Rating`.
   `Range_Insert()` sẽ chèn một bộ mới vào bảng Ratings và vào đúng phân mảnh dựa trên giá trị của Rating.

Câu hỏi thường gặp:

- Số phân mảnh bắt đầu từ 0, nếu có 3 phân mảnh thì tên các bảng sẽ là `range_part0`, `range_part1`, `range_part2` cho phân mảnh theo khoảng, và tương tự cho phân mảnh vòng tròn.
- Không được thay đổi tiền tố tên bảng phân mảnh đã được cung cấp trong `assignment_tester.py`.
- Không được mã hóa cứng tên tệp đầu vào.
- Không được đóng kết nối bên trong các hàm đã triển khai.
- Không được mã hóa cứng tên cơ sở dữ liệu.
- Lược đồ bảng phải giống với mô tả ở bước 4.

Về phân mảnh: Số phân mảnh nghĩa là số bảng sẽ được tạo ra.

Đối với giá trị Rating trong khoảng [0, 0.5, 1, 1.5, 2, 2.5, 3, 3.5, 4, 4.5, 5]:

- Trường hợp N = 1
  Một bảng chứa tất cả các giá trị.
- Trường hợp N = 2
  Hai bảng:
  `Partition 0` chứa các giá trị [0, 2.5]
  `Partition 1` chứa các giá trị (2.5, 5]
- Trường hợp N = 3
  Ba bảng:
  `Partition 0` chứa [0, 1.67]
  `Partition 1` chứa (1.67, 3.34]
  `Partition 2` chứa (3.34, 5]

"Uniform ranges" nghĩa là chia vùng giá trị một cách đồng đều. Hy vọng ví dụ đã làm rõ điều này.

Gợi ý:

1. Không dùng biến toàn cục trong quá trình triển khai. Cho phép sử dụng bảng meta-data.
2. Không được sửa đổi tệp dữ liệu.
3. Việc vượt qua tất cả các testcase kiểm thử không đảm bảo kết quả đúng hoàn toàn. Nó chỉ có nghĩa là mã của bạn không có lỗi biên dịch. Để kiểm tra đầy đủ, bạn cần kiểm tra nội dung các bảng trong cơ sở dữ liệu.
4. Hai hàm chèn có thể được gọi nhiều lần bất kỳ lúc nào. Chúng được thiết kế để duy trì các bảng trong cơ sở dữ liệu khi có thao tác chèn.

Nộp bài:

1. Sinh viên tạo thành nhóm 3 người.
2. Mỗi nhóm tạo 1 Github repository và nộp bài thông qua repository này. Lưu ý để chế độ public cho repository này.
3. Bài làm bao gồm phần code và báo cáo.
4. Yêu cầu về báo cáo:
    - Trình bày và giải thích cách giải quyết vấn đề
    - Phân chia công việc giữa các thành viên trong nhóm
5. Hạn nộp bài: thứ Bảy 10/6/2025