

HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG
KHOA CÔNG NGHỆ THÔNG TIN 1

-----oOo-----



BÁO CÁO BÀI TẬP LỚN CƠ SỞ DỮ LIỆU PHÂN TÁN
HỌC PHẦN: CƠ SỞ DỮ LIỆU PHÂN TÁN

Giảng viên hướng dẫn : TS. Kim Ngọc Bách

Nhóm môn học : 09

Nhóm bài tập lớn : 07

Nhóm sinh viên thực hiện :

Bùi Thị Trang – B22DCCN848

Mai Hoàng Xuân – B22DCCN925

Lê Văn Hiệp – B22DCCN296

Hà Nội, Tháng 06/2025

Lời cảm ơn

Trước tiên, nhóm em muốn gửi lời cảm ơn sâu sắc nhất đến thầy Kim Ngọc Bách, người đã tận tình hướng dẫn cũng như góp ý, đưa ra những định hướng cho chúng em trong suốt quá trình học tập và thực hiện nội dung môn học.

Nhóm em xin chân thành cảm ơn thầy vì sự tận tâm, trách nhiệm và những chia sẻ quý báu đã giúp chúng em hiểu rõ hơn về kiến thức chuyên môn cũng như cách áp dụng chúng vào thực tiễn.

Bên cạnh đó, cũng xin cảm ơn các bạn trong lớp đã luôn hỗ trợ, trao đổi và chia sẻ tài liệu, giúp các thành viên nhóm em có thêm động lực và định hướng trong quá trình hoàn thiện nội dung môn học và trách nhiệm của mình.

Cuối cùng, chúng em xin kính chúc thầy nhiều sức khỏe, luôn thành công trong sự nghiệp.

Chúng em xin chân thành cảm ơn!

Danh mục bảng biểu, hình ảnh

Bảng 1: Phân chia công việc và đóng góp nhóm.....	1
Bảng 2: So sánh 2 phương pháp phân vùng	8
Hình 1: Chi tiết yêu cầu bài tập	11
Hình 2: Các gợi ý làm bài	12
Hình 3: Kiểm tra kết nối cơ sở dữ liệu.....	19
Hình 4: Kiểm tra bảng ratings.....	20
Hình 5: Kiểm tra phân vùng Range	22
Hình 6: Kiểm tra tính đúng đắn của phân vùng:.....	23
Hình 7: Kiểm tra bản ghi chèn:.....	24
Hình 8: Kiểm tra các phân vùng range sau khi chèn bản ghi	25
Hình 9: Kiểm tra phân vùng Round-Robin.....	27
Hình 10: Kiểm tra bản ghi chèn.....	28
Hình 11: Kiểm tra roundrobin sau khi chèn bản ghi.....	29
Hình 12: Thời gian thực thi của các hàm.....	31

MỤC LỤC

Lời cảm ơn
Danh mục bảng biểu, hình ảnh.....
I, Phân công công việc nhóm.....	1
1, Phân chia công việc và đóng góp của các thành viên trong nhóm	1
2, Mô tả chi tiết công việc:.....	1
II. Tìm hiểu lý thuyết	4
1. Giới thiệu	4
2. Phân vùng theo khoảng giá trị (Range Partitioning)	5
2.1. Khái niệm.....	5
2.2. Nguyên lý hoạt động.....	5
2.3. Ví dụ minh họa.....	5
2.4. Ưu điểm.....	6
2.5. Nhược điểm.....	6
3. Phân vùng Round-Robin.....	7
3.1. Khái niệm.....	7
3.2. Nguyên lý hoạt động.....	7
3.3. Ví dụ minh họa.....	7
3.4. Ưu điểm.....	7
3.5. Nhược điểm.....	8
4. So sánh 2 phương pháp:	8
5. Kết luận	9
III. Tìm hiểu bài tập và hướng làm bài.....	10
1. Yêu cầu bài tập.....	10
2. Cơ sở làm bài	11
IV. Triển khai bài tập	13
1. Cấu trúc File Interface.py	13
1.1. Hàm getopenconnection.....	13

1.2. Hàm create_db	13
1.3. Hàm setup_metadata	13
1.4. Hàm loadratings	13
1.5. Hàm rangepartition	14
1.6. Hàm roundrobinpartition	14
1.7. Hàm rangeinsert	15
1.8. Hàm roundrobininsert	15
1.9. Hàm count_partitions	15
2. Những điều cần chú ý	16
3. Quá trình triển khai bài tập	16
IV. Kiểm Thử	17
1. Mô tả kiểm thử	17
2. Kết quả kiểm thử	17
3. Kiểm thử bằng câu lệnh SQL	18
3.1. Kết nối cơ sở dữ liệu	18
3.2. Kiểm tra bảng ratings	19
3.3. Kiểm tra phân vùng Range	21
4. Phân tích kết quả	29
5. Tổng kết	31

I, Phân công công việc nhóm

1, Phân chia công việc và đóng góp của các thành viên trong nhóm

Họ tên	MSV	Phân công nhiệm vụ	Tỉ lệ đóng góp
Bùi Thị Trang	B22DCCN	Quản lý dự án và phát triển cơ sở.	30%
Mai Hoàng Xuân	B22DCCN925	Phân vùng range và chèn dữ liệu range.	40%
Lê Văn Hiệp	B22DCCN	Phân vùng round-robin và chèn dữ liệu round-robin.	30%

Bảng 1: Phân chia công việc và đóng góp nhóm.

2, Mô tả chi tiết công việc:

Bùi Thị Trang: Quản lý dự án và phát triển cơ sở

- Trách nhiệm:
 - Quản lý tiến độ và phân chia công việc
 - Phụ trách hàm loadratings để tải dữ liệu từ file vào cơ sở dữ liệu
 - Thiết lập cấu trúc cơ sở dữ liệu và môi trường phát triển
 - Tham gia vào quá trình kiểm thử
- Nhóm đánh giá:
 - Hoàn thành tốt nhiệm vụ được giao: Triển khai thành công hàm loadratings, thiết lập môi trường ổn định, và quản lý tiến độ hiệu quả.
 - Hoàn thành nhiệm vụ đúng thời hạn: Tất cả công việc được giao (quản lý, phát triển, kiểm thử) đều hoàn thành đúng hoặc trước thời hạn.
 - Đóng góp nổi bật: Đảm bảo sự phối hợp, kết nối giữa các thành viên, giúp dự án tiến hành suôn sẻ và đạt kết quả cao.

Mai Hoàng Xuân: Phân vùng range và chèn dữ liệu range

- Trách nhiệm:
 - Phát triển hàm `rangepartition` để phân vùng dữ liệu theo khoảng giá trị.
 - Phát triển hàm `rangeinsert` để chèn dữ liệu mới vào phân vùng range.
 - Tìm hiểu lý thuyết + Viết tài liệu mô tả phần việc của mình.
 - Tích hợp các phần công việc của từng thành viên nhóm.
 - Phụ trách quá trình kiểm thử và tối ưu.
- Đánh giá của nhóm:
 - Hoàn thành tốt nhiệm vụ được giao: Triển khai thành công `rangepartition` và `rangeinsert`, đảm bảo tính đúng đắn và hiệu quả.
 - Hoàn thành nhiệm vụ đúng thời hạn: Tất cả công việc (phát triển, tài liệu, kiểm thử, tích hợp) được hoàn thành đúng hoặc trước thời hạn.
 - Đóng góp tích cực: Đảm nhận trọng số công việc cao hơn (tích hợp, kiểm thử và tối ưu, viết tài liệu), thể hiện vai trò lãnh đạo kỹ thuật trong nhóm.

Lê Văn Hiệp: Phân vùng round-robin và chèn dữ liệu round-robin

- Trách nhiệm:
 - Phát triển hàm `roundrobinpartition` để phân vùng dữ liệu theo phương pháp round-robin
 - Phát triển hàm `roundrobininsert` để chèn dữ liệu mới vào phân vùng round-robin
 - Tìm hiểu lý thuyết + Viết tài liệu mô tả phần việc của mình
- Đánh giá của nhóm:
 - Hoàn thành tốt nhiệm vụ được giao: Triển khai thành công `roundrobinpartition` và `roundrobininsert`, đảm bảo phân bổ đồng đều và tính đúng đắn.

- Hoàn thành nhiệm vụ đúng thời hạn: Hoàn thành phát triển, tài liệu và kiểm thử đúng thời gian quy định.
- Đóng góp: Đảm bảo các hàm Round-Robin hoạt động ổn định, hỗ trợ tích hợp với các phần khác của dự án.

II. Tìm hiểu lý thuyết

1. Giới thiệu

Phân vùng dữ liệu (data partitioning) là kỹ thuật chia tập dữ liệu trong cơ sở dữ liệu thành các tập con (partition) nhằm cải thiện hiệu suất truy vấn, khả năng mở rộng và quản lý dữ liệu trong các hệ thống phân tán. Phân vùng hỗ trợ xử lý song song, cân bằng tải và giảm thời gian truy cập dữ liệu. Báo cáo này phân tích cách sử dụng hai phương pháp phân vùng chính: Phân vùng theo khoảng giá trị (Range Partitioning) và Phân vùng Round-Robin, áp dụng trên tập dữ liệu ratings.dat từ MovieLens, sử dụng cột Rating cho Range Partitioning và phân bổ tuần hoàn cho Round-Robin, không sử dụng cột Timestamp. Báo cáo trình bày khái niệm, nguyên lý, ưu điểm, nhược điểm, ví dụ minh họa, so sánh hai phương pháp và cách triển khai, kiểm thử bài tập.

Tập dữ liệu ratings.dat có định dạng: UserID::MovieID::Rating::Timestamp.

Ví dụ:

1::122::5::838985046

1::185::5::838983525

2::231::4::838983392

3::456::3::838983421

4::789::4::838983450

- UserID: Mã người dùng (số nguyên).
- MovieID: Mã phim (số nguyên).
- Rating: Điểm đánh giá (số thực từ 0.5 đến 5, chia nửa sao).
- Timestamp: Không sử dụng trong phân vùng.

Mục tiêu là chia dữ liệu thành N phân vùng đồng đều về số lượng bản ghi, sử dụng Rating cho Range Partitioning và phân bổ tuần hoàn cho Round-Robin.

2. Phân vùng theo khoảng giá trị (Range Partitioning)

2.1. Khái niệm

Range Partitioning chia dữ liệu thành N phân vùng dựa trên các khoảng giá trị liên tục của cột Rating. Mỗi phân vùng chứa các bản ghi có giá trị Rating nằm trong một khoảng xác định, được thiết kế để số lượng bản ghi trong mỗi phân vùng xấp xỉ bằng nhau, đảm bảo sự đồng đều.

2.2. Nguyên lý hoạt động

Khóa phân vùng: Sử dụng cột Rating (giá trị từ 0 đến 5) để chia thành N khoảng đồng đều.

Quy tắc phân vùng: Phạm vi $[0, 5]$ được chia thành N khoảng, mỗi khoảng có kích thước $\text{delta} = 5.0 / N$. Bản ghi được gán vào phân vùng dựa trên giá trị Rating.

Truy vấn: Hệ thống chỉ cần quét phân vùng chứa khoảng Rating tương ứng, giảm phạm vi truy vấn.

2.3. Ví dụ minh họa

- Giả sử chia 100,000 bản ghi thành $N = 3$ phân vùng:

- Phân vùng 0: Rating từ $[0, 1.67]$
- Phân vùng 1: Rating từ $(1.67, 3.34]$
- Phân vùng 2: Rating từ $(3.34, 5]$

- Dữ liệu mẫu:

1::122::5 → Phân vùng 2 (Rating = 5)

1::185::5 → Phân vùng 2 (Rating = 5)

2::231::4 → Phân vùng 2 (Rating = 4)

3::456::3 → Phân vùng 1 (Rating = 3)

4::789::4 → Phân vùng 2 (Rating = 4)

- Tính đồng đều: Để đảm bảo mỗi phân vùng có khoảng $100,000 / 3 \approx 33,333$ bản ghi, cần phân tích phân bố thực tế của Rating. Nếu Rating tập trung ở một số giá trị (ví dụ: nhiều bản ghi có Rating 4 hoặc 5), cần điều chỉnh ranh giới dựa trên số lượng bản ghi.

- Truy vấn mẫu: `SELECT * FROM ratings WHERE Rating = 4` chỉ quét Phân vùng 2, tăng hiệu suất.

2.4. Ưu điểm

- Hiệu quả truy vấn theo Rating: Tối ưu cho truy vấn trong một khoảng Rating (ví dụ: tìm đánh giá từ 3.34 đến 5).

- Quản lý logic: Dễ dàng quản lý dữ liệu theo nhóm Rating, ví dụ: xóa phân vùng chứa đánh giá thấp.

- Hỗ trợ hệ thống phân tán: Phân vùng có thể lưu trên các nút khác nhau, hỗ trợ xử lý song song.

- Phân tích chất lượng: Phù hợp với thống kê dựa trên Rating.

2.5. Nhược điểm

- Khó đảm bảo đồng đều: Nếu Rating không phân bố đều, một số phân vùng có thể chứa nhiều bản ghi hơn, gây mất cân bằng tải (data skew).

- Thiết kế phức tạp: Yêu cầu phân tích phân bố Rating để xác định ranh giới.

- Hạn chế mở rộng: Thêm dữ liệu mới ngoài phạm vi $[0, 5]$ đòi hỏi điều chỉnh ranh giới.

- Hiệu suất thấp với truy vấn ngoài khóa: Truy vấn dựa trên UserID hoặc MovieID phải quét tất cả phân vùng.

3. Phân vùng Round-Robin

3.1. Khái niệm

Round-Robin phân bổ bản ghi lần lượt vào N phân vùng theo chu kỳ tuần hoàn, không dựa trên giá trị của bất kỳ cột nào. Phương pháp này tự động đảm bảo số bản ghi gần bằng nhau giữa các phân vùng, lý tưởng cho cân bằng tải.

3.2. Nguyên lý hoạt động

Phân bổ tuần hoàn: Bản ghi được gán lần lượt vào các phân vùng (P1, P2, ..., PN, P1, ...).

Không cần khóa phân vùng: Không yêu cầu phân tích phân bổ dữ liệu.

Truy vấn: Do dữ liệu phân bổ ngẫu nhiên, truy vấn theo bất kỳ cột nào (UserID, MovieID, Rating) phải quét tất cả phân vùng.

3.3. Ví dụ minh họa

Với $N = 3$ phân vùng (P1, P2, P3):

Bản ghi 1: 1::122::5 \rightarrow P1

Bản ghi 2: 1::185::5 \rightarrow P2

Bản ghi 3: 2::231::4 \rightarrow P3

Bản ghi 4: 3::456::3 \rightarrow P1

Bản ghi 5: 4::789::4 \rightarrow P2

Tính đồng đều: Với 100,000 bản ghi, mỗi phân vùng chứa khoảng $100,000 / 3 \approx 33,333$ bản ghi, bất kể giá trị UserID, MovieID hay Rating.

Truy vấn mẫu: `SELECT * FROM ratings WHERE Rating = 4` phải quét tất cả phân vùng (P1, P2, P3), do Rating = 4 phân bố ngẫu nhiên.

3.4. Ưu điểm

- Phân bổ đồng đều tự nhiên: Số bản ghi gần như bằng nhau trên N phân vùng, tránh mất cân bằng tải.
- Đơn giản: Không cần phân tích dữ liệu hoặc chọn khóa phân vùng.
- Hiệu quả ghi dữ liệu: Ghi nhanh vì chỉ cần gán bản ghi vào phân vùng tiếp theo.

- Tối ưu cho hệ thống phân tán: Cân bằng tải trên các nút.

3.5. Nhược điểm

- Hiệu suất truy vấn thấp: Truy vấn theo UserID, MovieID hoặc Rating phải quét tất cả phân vùng.
- Không tối ưu cho lọc dữ liệu: Không hiệu quả khi tìm bản ghi theo điều kiện cụ thể.
- Khó quản lý logic: Không thể nhóm dữ liệu theo UserID, MovieID hoặc Rating.
- Xóa dữ liệu phức tạp: Xóa bản ghi theo một điều kiện cụ thể đòi hỏi quét tất cả phân vùng.

4. So sánh 2 phương pháp:

Tiêu chí	Range Partitioning (dựa trên Rating)	Round-Robin
Phù hợp	Truy vấn theo khoảng Rating, phân tích chất lượng	Ghi dữ liệu nặng, cân bằng tải
Không phù hợp	Phân bố Rating không đều, truy vấn UserID/MovieID	Truy vấn theo Rating/ UserID/ MovieID, quản lý logic
Đồng đều	Cần phân tích phân bố Rating để thiết kế khoảng	Tự động đồng đều
Hiệu suất truy vấn	Cao cho truy vấn theo Rating	Thấp, phải quét tất cả phân vùng
Hiệu suất ghi	Phức tạp hơn do cần xác định phân vùng	Nhanh, gán lần lượt
Quản lý dữ liệu	Dễ quản lý theo nhóm Rating	Khó quản lý theo logic nghiệp vụ

Bảng 2: So sánh 2 phương pháp phân vùng

Tính đồng đều:

- Range Partitioning: Phụ thuộc vào phân bố Rating. Nếu Rating tập trung (ví dụ: nhiều bản ghi có Rating 4 và 5), cần điều chỉnh ranh giới dựa trên số lượng bản ghi.

- Round-Robin: Tự động đồng đều do phân bố tuần hoàn, không phụ thuộc vào giá trị dữ liệu.

5. Kết luận

Range Partitioning phù hợp khi ưu tiên truy vấn theo khoảng Rating hoặc phân tích chất lượng phim, nhưng yêu cầu phân tích phân bố Rating để đảm bảo đồng đều. Nó tối ưu cho hệ thống cần truy vấn nhanh trên một khoảng giá trị.

Round-Robin đơn giản, tự động cân bằng tải, lý tưởng cho ghi dữ liệu nhanh và hệ thống phân tán, nhưng kém hiệu quả cho truy vấn theo bất kỳ cột nào.

Với tập dữ liệu ratings.dat, Range Partitioning là lựa chọn tốt khi cần tối ưu truy vấn liên quan đến Rating, trong khi Round-Robin phù hợp khi ưu tiên ghi dữ liệu nhanh và cân bằng tải. Để đảm bảo đồng đều trong Range Partitioning, cần điều chỉnh ranh giới dựa trên số lượng bản ghi thực tế, đặc biệt khi Rating có phân bố không đồng đều.

III. Tìm hiểu bài tập và hướng làm bài

1. Yêu cầu bài tập

Bài tập yêu cầu triển khai các hàm trong file `Interface.py` để thực hiện:

- Tạo và kết nối cơ sở dữ liệu: Tạo cơ sở dữ liệu `dds_assgn1` và thiết lập kết nối PostgreSQL.
- Nạp dữ liệu: Hàm `loadratings` nạp dữ liệu từ file `ratings.dat` vào bảng `ratings`.
- Phân vùng dữ liệu:
 - Range Partitioning: Hàm `rangepartition` chia bảng `ratings` thành N phân vùng dựa trên cột `Rating`.
 - Round-Robin: Hàm `roundrobinpartition` phân bổ bản ghi tuần hoàn vào N phân vùng.
- Chèn dữ liệu mới:
 - Hàm `rangeinsert`: Chèn bản ghi mới vào bảng chính và phân vùng Range tương ứng.
 - Hàm `roundrobininsert`: Chèn bản ghi mới vào bảng chính và phân vùng Round-Robin tương ứng.
- Quản lý metadata: Lưu trữ thông tin phân vùng (số phân vùng, kích thước khoảng, số bản ghi) trong bảng `metadata`.
- Kiểm thử: Sử dụng `Assignment1 Tester.py` để kiểm tra tính đúng đắn của các hàm, đảm bảo hoàn thiện (completeness), không giao nhau (disjointness) và tái tạo (reconstruction).

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:
 - o `UserID (int)`
 - o `MovieID (int)`
 - o `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`.

Hình 1: Chi tiết yêu cầu bài tập

2. Cơ sở làm bài

- Công nghệ sử dụng: Python với thư viện `psycopg2` để tương tác với PostgreSQL.
- Tập dữ liệu: File `ratings.dat` chứa 10,000,054 bản ghi, định dạng `UserID::MovieID::Rating::Timestamp`.

- Các file Assignment1Tester.py, testHelper.py (có sẵn), file Interface.py chứa các hàm đã được thầy định nghĩa từ trước, chỉ cần tiến hành triển khai.
- Bảng chính: Bảng ratings lưu trữ UserID, MovieID, Rating sau khi bỏ cột Timestamp.
- Bảng phân vùng:
 - Range: Tên range_part0, range_part1, ..., chứa bản ghi dựa trên khoảng Rating.
 - Round-Robin: Tên rrobin_part0, rrobin_part1, ..., chứa bản ghi phân bổ tuần hoàn.
- Bảng metadata: Lưu thông tin về partition_type (range hoặc roundrobin), numberofpartitions, delta (cho Range), và row_count (cho Round-Robin).
- Kiểm thử: Sử dụng testHelper.py để kiểm tra:
 - Số lượng bản ghi trong bảng ratings sau khi nạp.
 - Số lượng phân vùng và bản ghi trong mỗi phân vùng.
 - Tính hoàn thiện, không giao nhau, tái tạo của các phân vùng.
 - Tính chính xác của việc chèn dữ liệu mới.
- Các gợi ý làm bài của thầy:

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.

Hình 2: Các gợi ý làm bài

IV. Triển khai bài tập

1. Cấu trúc File Interface.py

File Interface.py chứa các hàm chính để thực hiện phân vùng và quản lý dữ liệu:

1.1. Hàm *getopenconnection*

- Mục đích: Thiết lập kết nối tới cơ sở dữ liệu PostgreSQL.
- Triển khai: Sử dụng `psycopg2.connect` với thông tin user, password, dbname, và `host=localhost`.
- Lưu ý: Đặt chế độ `AUTOCOMMIT` khi cần tạo/xóa cơ sở dữ liệu.

1.2. Hàm *create_db*

- Mục đích: Tạo cơ sở dữ liệu `dds_assgn1` nếu chưa tồn tại.
- Triển khai: Kiểm tra sự tồn tại của cơ sở dữ liệu trong `pg_catalog.pg_database` và tạo mới nếu cần.

1.3. Hàm *setup_metadata*

- Mục đích: Tạo bảng metadata để lưu thông tin phân vùng.
- Triển khai: Tạo bảng với các cột:
 - `partition_type` (VARCHAR): range hoặc roundrobin.
 - `numberofpartitions` (INTEGER): Số phân vùng.
 - `delta` (FLOAT): Kích thước khoảng cho Range Partitioning.
 - `row_count` (BIGINT): Số bản ghi (cho Round-Robin).
- Lưu ý: Xóa dữ liệu cũ để đảm bảo tính nhất quán.

1.4. Hàm *loadratings*

- Mục đích: Nạp dữ liệu từ `ratings.dat` vào bảng ratings.
- Triển khai:

- Tạo bảng ratings với các cột tạm thời (extra1, extra2, extra3, timestamp) để phù hợp với định dạng file.
- Sử dụng `cur.copy_from` để nạp dữ liệu nhanh.
- Xóa các cột tạm thời, chỉ giữ `userid`, `movieid`, `rating`.
- Cập nhật metadata với số bản ghi (`row_count`) cho roundrobin.

1.5. Hàm *rangepartition*

- Mục đích: Chia bảng ratings thành N phân vùng dựa trên Rating.
- Triển khai:
 - Tạo N bảng (`range_part0`, `range_part1`, ...).
 - Tính $\text{delta} = 5.0 / N$ để chia khoảng $[0, 5]$.
 - Chèn bản ghi vào phân vùng tương ứng dựa trên Rating:
 - Phân vùng 0: Rating từ $[0, \text{delta}]$.
 - Phân vùng i: Rating từ $(i * \text{delta}, (i+1) * \text{delta}]$ (trừ phân vùng cuối).
 - Lưu `numberofpartitions` và `delta` vào metadata.

1.6. Hàm *roundrobinpartition*

- Mục đích: Phân bổ bản ghi tuần hoàn vào N phân vùng.
- Triển khai:
 - Tạo N bảng (`rrobin_part0`, `rrobin_part1`, ...).
 - Sử dụng `ROW_NUMBER()` để đánh số thứ tự bản ghi.
 - Chèn bản ghi vào phân vùng i nếu $(\text{ROW_NUMBER}-1) \% N = i$.
 - Lưu `numberofpartitions` và `row_count` vào metadata.

1.7. Hàm *rangeinsert*

- Mục đích: Chèn bản ghi mới vào bảng ratings và phân vùng Range tương ứng.

- Triển khai:

- Chèn bản ghi vào bảng ratings.
- Lấy `numberofpartitions` và `delta` từ metadata.
- Tính chỉ số phân vùng: $\text{index} = \text{int}(\text{rating} / \text{delta})$ (điều chỉnh nếu $\text{rating} \% \text{delta} == 0$).
- Chèn vào bảng `range_part{index}`.

1.8. Hàm *roundrobininsert*

- Mục đích: Chèn bản ghi mới vào bảng ratings và phân vùng RoundRobin tương ứng.

- Triển khai:

- Chèn bản ghi vào bảng ratings.
- Lấy `row_count` và `numberofpartitions` từ metadata.
- Tính chỉ số phân vùng: $\text{index} = \text{row_count} \% \text{numberofpartitions}$.
- Chèn vào bảng `rrobin_part{index}`.
- Cập nhật `row_count` trong metadata.

1.9. Hàm *count_partitions*

- Mục đích: Đếm số bảng có tiền tố `range_part` hoặc `rrobin_part`.

- Triển khai: Truy vấn `pg_catalog.pg_tables` để đếm số bảng có tên bắt đầu bằng tiền tố.

2. Những điều cần chú ý

- Tính đồng đều:
 - Range Partitioning: Chia đều khoảng $[0, 5]$ bằng delta, nhưng không đảm bảo số bản ghi đồng đều do phân bố Rating không đều.
 - Round-Robin: Tự động đồng đều do phân bố tuần hoàn.
- Hiệu suất:
 - Sử dụng chỉ mục (CREATE INDEX) trên cột Rating để tăng tốc truy vấn trong rangepartition.
 - Sử dụng `cur.copy_from` để nạp dữ liệu nhanh trong loadratings.
- Quản lý lỗi: Kiểm tra số phân vùng, phân bố bản ghi và tính hợp lệ của Rating ($0 \leq \text{Rating} \leq 5$).

3. Quá trình triển khai bài tập

Quá trình triển khai bài tập được thực hiện theo các bước sau:

Bước 1: Triển khai các hàm trong Interface.py: Phát triển các hàm theo yêu cầu (kết nối cơ sở dữ liệu, nạp dữ liệu, phân vùng, chèn dữ liệu, quản lý metadata).

Bước 2: Kiểm thử trên tập dữ liệu mẫu (nhỏ): Sử dụng tập dữ liệu nhỏ `test_data.dat` (chỉ 20 bản ghi) để kiểm tra tính đúng đắn của các hàm.

Bước 3: Sửa lỗi (fix bug): Xử lý các vấn đề phát sinh trong quá trình kiểm thử.

Bước 4: Kiểm thử trên tập dữ liệu đầy đủ `ratings.dat`: Áp dụng trên tập dữ liệu thực tế với 10,000,054 bản ghi.

Bước 5: Sửa lỗi và tối ưu logic: Cải thiện logic của các hàm để xử lý các lỗi phát sinh, đảm bảo tính đúng đắn và đồng đều.

Bước 6: Tối ưu thời gian thực thi: Áp dụng các kỹ thuật như tạo chỉ mục, sử dụng các lệnh SQL tối ưu truy vấn hiệu quả.

IV. Kiểm Thử

1. Mô tả kiểm thử

File Assignment1Tester.py được sử dụng để kiểm tra các hàm trong Interface.py trên tập dữ liệu mẫu test_data.dat chứa 20 dòng dữ liệu, với định dạng UserID::MovieID::Rating::Timestamp. Tập mẫu nhỏ giúp dễ dàng kiểm tra tính đúng đắn và xem toàn bộ nội dung các bảng phân vùng mà không bị giới hạn bởi khối lượng dữ liệu lớn. Các tiêu chí kiểm thử bao gồm:

- loadratings: Kiểm tra bảng ratings chứa đúng 20 bản ghi.
- rangepartition: Kiểm tra số phân vùng, tính hoàn thiện (tất cả bản ghi được phân vùng), không giao nhau (mỗi bản ghi chỉ thuộc một phân vùng), và tái tạo (tổng số bản ghi từ các phân vùng khớp với bảng chính).
- roundrobinpartition: Tương tự rangepartition, kiểm tra sự phân bố tuần hoàn.
- rangeinsert: Kiểm tra bản ghi mới (100, 2, 3) được chèn vào đúng phân vùng Range.
- roundrobininsert: Kiểm tra bản ghi mới (100, 1, 3) được chèn vào đúng phân vùng Round-Robin.

Kiểm thử được thực hiện với $N = 5$ phân vùng, sử dụng các câu lệnh SQL để xác minh kết quả trực tiếp trên cơ sở dữ liệu dds_assgn1.

2. Kết quả kiểm thử

Chạy Assignment1Tester.py với tập dữ liệu mẫu test_data.dat và $N = 5$ phân vùng:

- loadratings: Pass. Bảng ratings chứa đúng 20 bản ghi, khớp với số dòng trong test_data.dat.
- rangepartition: Pass. Tạo 5 bảng range_part0 đến range_part4, mỗi bảng chứa bản ghi theo khoảng Rating:

- range_part0: [0, 1] → 3 bản ghi.
- range_part1: (1, 2] → 3 bản ghi.
- range_part2: (2, 3] → 3 bản ghi.
- range_part3: (3, 4] → 5 bản ghi.
- range_part4: (4, 5] → 6 bản ghi.
- Tổng số bản ghi: $3 + 3 + 3 + 5 + 6 = 20$, thỏa mãn tính hoàn thiện, không giao nhau, và tái tạo.
- roundrobinpartition: Pass. Tạo 5 bảng rrobin_part0 đến rrobin_part4, mỗi bảng chứa 4 bản ghi:
 - Mỗi bảng: 4 bản ghi ($20 / 5 = 4$).
 - Tổng số bản ghi: $4 + 4 + 4 + 4 + 4 = 20$, thỏa mãn các tiêu chí.
- rangeinsert: Pass. Chèn bản ghi (100, 2, 3) vào bảng range_part2 (vì Rating = 3 thuộc (2, 3]). Sau khi chèn, range_part2 có 4 bản ghi, tổng số bản ghi trong các phân vùng là 21.
- roundrobininsert: Pass. Chèn bản ghi (100, 1, 3) vào bảng rrobin_part0 (dựa trên row_count = 20, $(20 + 1) \% 5 = 0$). Sau khi chèn, rrobin_part0 có 5 bản ghi, tổng số bản ghi là 21.

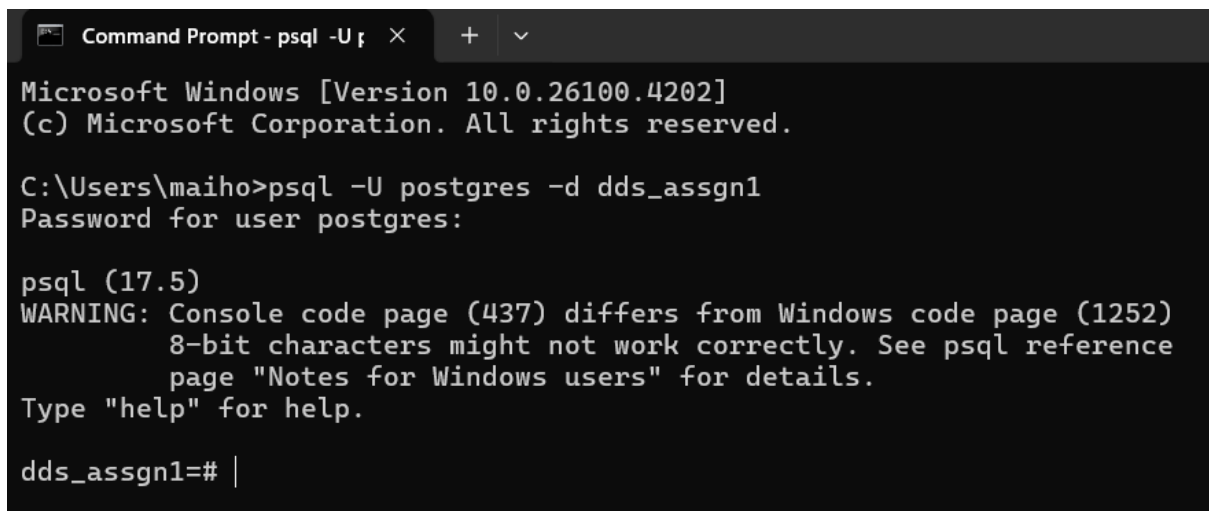
3. Kiểm thử bằng câu lệnh SQL

Các câu lệnh SQL được thực thi trong psql -U postgres -d dds_assgn1 để kiểm tra kết quả:

3.1. Kết nối cơ sở dữ liệu

```
psql -U postgres -d dds_assgn1
```

Kết quả: Kết nối thành công.



```
Command Prompt - psql -U postgres
Microsoft Windows [Version 10.0.26100.4202]
(c) Microsoft Corporation. All rights reserved.

C:\Users\maiho>psql -U postgres -d dds_assgn1
Password for user postgres:

psql (17.5)
WARNING: Console code page (437) differs from Windows code page (1252)
         8-bit characters might not work correctly. See psql reference
         page "Notes for Windows users" for details.
Type "help" for help.

dds_assgn1=# |
```

Hình 3: Kiểm tra kết nối cơ sở dữ liệu

3.2. Kiểm tra bảng ratings

```
SELECT COUNT(*) FROM ratings;
```

Kết quả: 20 (trước khi chèn thêm bản ghi).

```
SELECT * FROM ratings;
```

Kết quả: Trả về 20 dòng với các cột userid, movieid, rating (cột timestamp đã bị loại bỏ).


```
dds_assgn1=# SELECT COUNT(*) FROM ratings;  
count
```

```
-----
```

```
      20  
(1 row)
```

```
dds_assgn1=# SELECT * FROM ratings;  
userid | movieid | rating
```

```
-----+-----+-----
```

1	122	5
1	185	4.5
1	231	4
1	292	3.5
1	316	3
1	329	2.5
1	355	2
1	356	1.5
1	362	1
1	364	0.5
1	370	0
1	377	3.5
1	420	5
1	466	4
1	480	5
1	520	2.5
1	539	5
1	586	3.5
1	588	5
1	589	1.5

```
(20 rows)
```

```
dds_assgn1=# |
```

Hình 4: Kiểm tra bảng ratings

3.3. Kiểm tra phân vùng Range

```
SELECT COUNT(*) FROM range_part0;
```

```
SELECT COUNT(*) FROM range_part1;
```

```
SELECT COUNT(*) FROM range_part2;
```

```
SELECT COUNT(*) FROM range_part3;
```

```
SELECT COUNT(*) FROM range_part4;
```

Kết quả (trước khi chèn):

- range_part0: 3
- range_part1: 3
- range_part2: 3
- range_part3: 5
- range_part4: 6
- Tổng cộng: $3 + 3 + 3 + 5 + 6 = 20$

```

dds_assgn1=# SELECT COUNT(*) FROM range_part0;
count
-----
      3
(1 row)

dds_assgn1=# SELECT COUNT(*) FROM range_part1;
count
-----
      3
(1 row)

dds_assgn1=# SELECT COUNT(*) FROM range_part2;
count
-----
      3
(1 row)

dds_assgn1=# SELECT COUNT(*) FROM range_part3;
count
-----
      5
(1 row)

dds_assgn1=# SELECT COUNT(*) FROM range_part4;
count
-----
      6
(1 row)

```

Hình 5: Kiểm tra phân vùng Range

Kiểm tra tính đúng đắn của phân vùng:

```
SELECT * FROM range_part0 WHERE rating < 0 OR rating > 1;
```

SELECT * FROM range_part1 WHERE rating <= 1 OR rating > 2;

SELECT * FROM range_part2 WHERE rating <= 2 OR rating > 3;

SELECT * FROM range_part3 WHERE rating <= 3 OR rating > 4;

SELECT * FROM range_part4 WHERE rating <= 4 OR rating > 5;

Kết quả: Không có dòng nào trả về, xác nhận các phân vùng đúng khoảng Rating.

```
dds_assgn1=# SELECT * FROM range_part0 WHERE rating < 0 OR rating > 1;
userid | movieid | rating
-----+-----+-----
(0 rows)

dds_assgn1=# SELECT * FROM range_part1 WHERE rating <= 1 OR rating > 2;
userid | movieid | rating
-----+-----+-----
(0 rows)

dds_assgn1=# SELECT * FROM range_part2 WHERE rating <= 2 OR rating > 3;
userid | movieid | rating
-----+-----+-----
(0 rows)

dds_assgn1=# SELECT * FROM range_part3 WHERE rating <= 3 OR rating > 4;
userid | movieid | rating
-----+-----+-----
(0 rows)

dds_assgn1=# SELECT * FROM range_part4 WHERE rating <= 4 OR rating > 5;
userid | movieid | rating
-----+-----+-----
(0 rows)
```

Hình 6: Kiểm tra tính đúng đắn của phân vùng:

Kiểm tra bản ghi chèn:

SELECT * FROM range_part2 WHERE userid = 100 AND movieid = 2 AND
rating = 3;

SELECT * FROM ratings WHERE userid = 100 AND movieid = 2 AND rating
= 3;

Kết quả: Cả hai truy vấn trả về bản ghi (100, 2, 3), xác nhận bản ghi được chèn đúng vào range_part2 và ratings.

```
dds_assgn1=# SELECT * FROM range_part2 WHERE userid=100 AND movieid=2 AND rating=3;
userid | movieid | rating
-----+-----+-----
    100 |         2 |      3
(1 row)

dds_assgn1=# SELECT * FROM ratings WHERE userid=100 AND movieid=2 AND rating=3;
userid | movieid | rating
-----+-----+-----
    100 |         2 |      3
(1 row)
```

Hình 7: Kiểm tra bản ghi chèn:

Sau khi chèn:

```
SELECT COUNT(*) FROM range_part0;
```

```
SELECT COUNT(*) FROM range_part1;
```

```
SELECT COUNT(*) FROM range_part2;
```

```
SELECT COUNT(*) FROM range_part3;
```

```
SELECT COUNT(*) FROM range_part4;
```

Kết quả:

- range_part0: 3
- range_part1: 3
- range_part2: 4
- range_part3: 5
- range_part4: 6
- Tổng cộng: $3 + 3 + 4 + 5 + 6 = 21$

```

dds_assgn1=# SELECT COUNT(*) FROM range_part0;
 count
-----
      3
(1 row)

dds_assgn1=# SELECT COUNT(*) FROM range_part1;
 count
-----
      3
(1 row)

dds_assgn1=# SELECT COUNT(*) FROM range_part2;
 count
-----
      4
(1 row)

dds_assgn1=# SELECT COUNT(*) FROM range_part3;
 count
-----
      5
(1 row)

dds_assgn1=# SELECT COUNT(*) FROM range_part4;
 count
-----
      6
(1 row)

```

Hình 8: Kiểm tra các phân vùng range sau khi chèn bản ghi

3.4. Kiểm tra phân vùng Round-Robin

```
SELECT COUNT(*) FROM rrobin_part0;
```

```
SELECT COUNT(*) FROM rrobin_part1;
```

```
SELECT COUNT(*) FROM rrobin_part2;
```

```
SELECT COUNT(*) FROM rrobin_part3;
```

```
SELECT COUNT(*) FROM rrobin_part4;
```

Kết quả (trước khi chèn):

- rrobin_part0: 4
- rrobin_part1: 4
- rrobin_part2: 4
- rrobin_part3: 4
- rrobin_part4: 4
- Tổng cộng: $4 + 4 + 4 + 4 + 4 = 20$

```

dds_assgn1=# SELECT COUNT(*) FROM rrobin_part0;
count
-----
      4
(1 row)

dds_assgn1=# SELECT COUNT(*) FROM rrobin_part1;
count
-----
      4
(1 row)

dds_assgn1=# SELECT COUNT(*) FROM rrobin_part2;
count
-----
      4
(1 row)

dds_assgn1=# SELECT COUNT(*) FROM rrobin_part3;
count
-----
      4
(1 row)

dds_assgn1=# SELECT COUNT(*) FROM rrobin_part4;
count
-----
      4
(1 row)

```

Hình 9: Kiểm tra phân vùng Round-Robin

Kiểm tra bản ghi chèn:

```

SELECT * FROM rrobin_part0 WHERE userid = 100 AND movieid = 1 AND
rating = 3;

```



```
SELECT * FROM ratings WHERE userid = 100 AND movieid = 1 AND rating = 3;
```

Kết quả: Cả hai truy vấn trả về bản ghi (100, 1, 3), xác nhận bản ghi được chèn đúng vào rrobin_part0 và ratings.

```
dds_assgn1=# SELECT * FROM rrobin_part0 WHERE userid=100 AND movieid=1 AND rating=3;
userid | movieid | rating
-----+-----+-----
    100 |         1 |         3
(1 row)

dds_assgn1=# SELECT * FROM ratings WHERE userid=100 AND movieid=1 AND rating=3;
userid | movieid | rating
-----+-----+-----
    100 |         1 |         3
(1 row)
```

Hình 10: Kiểm tra bản ghi chèn

Sau khi chèn:

```
SELECT COUNT(*) FROM rrobin_part0;
```

```
SELECT COUNT(*) FROM rrobin_part1;
```

```
SELECT COUNT(*) FROM rrobin_part2;
```

```
SELECT COUNT(*) FROM rrobin_part3;
```

```
SELECT COUNT(*) FROM rrobin_part4;
```

Kết quả:

- rrobin_part0: 5
- rrobin_part1: 4
- rrobin_part2: 4
- rrobin_part3: 4
- rrobin_part4: 4
- Tổng cộng: $5 + 4 + 4 + 4 + 4 = 21$

```

dds_assgn1=# SELECT COUNT(*) FROM rrobin_part0;
count
-----
      5
(1 row)

dds_assgn1=# SELECT COUNT(*) FROM rrobin_part1;
count
-----
      4
(1 row)

dds_assgn1=# SELECT COUNT(*) FROM rrobin_part2;
count
-----
      4
(1 row)

dds_assgn1=# SELECT COUNT(*) FROM rrobin_part3;
count
-----
      4
(1 row)

dds_assgn1=# SELECT COUNT(*) FROM rrobin_part4;
count
-----
      4
(1 row)

```

Hình 11: Kiểm tra roundrobin sau khi chèn bản ghi

4. Phân tích kết quả

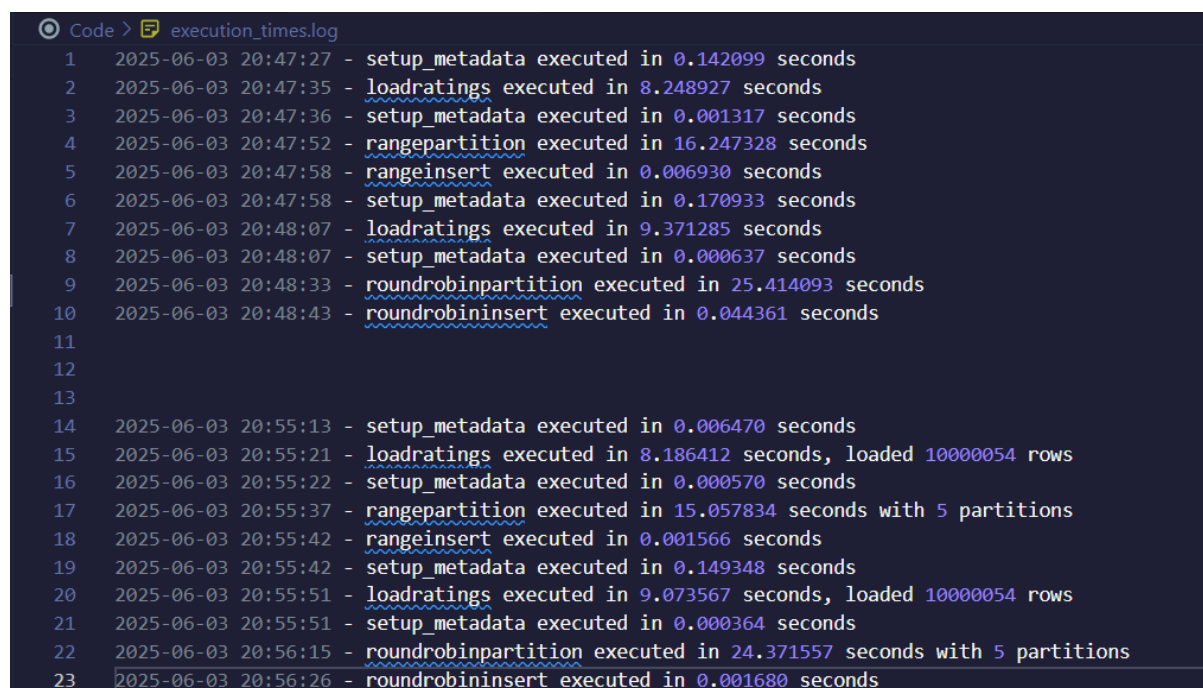
- Range Partitioning:

- Hiệu quả: Tối ưu cho truy vấn theo Rating, ví dụ: tìm bản ghi với Rating = 3 chỉ cần quét range_part2.
- Hạn chế: Số bản ghi không đồng đều (3, 3, 3, 5, 6) do phân bố Rating không đều trong tập mẫu. Để cải thiện, có thể phân tích phân bố Rating trước và điều chỉnh ranh giới để mỗi phân vùng có số bản ghi gần bằng nhau (khoảng 4 bản ghi/phân vùng).
- Ưu điểm kiểm thử mẫu: Với 20 bản ghi, dễ dàng kiểm tra toàn bộ nội dung các phân vùng và xác minh tính đúng đắn.
- Round-Robin:
 - Hiệu quả: Đảm bảo đồng đều tự nhiên (4 bản ghi/phân vùng), phù hợp cho ghi dữ liệu nhanh và cân bằng tải.
 - Hạn chế: Truy vấn theo Rating hoặc UserID kém hiệu quả do phải quét tất cả phân vùng.
 - Ưu điểm kiểm thử mẫu: Phân bố đều (4, 4, 4, 4, 4) dễ dàng xác minh, đặc biệt với tập dữ liệu nhỏ.
- Hiệu suất:
 - Nạp dữ liệu nhanh nhờ cur.copy_from, phù hợp ngay cả với tập mẫu nhỏ.
 - Phân vùng và chèn dữ liệu hoạt động đúng, thời gian thực thi không đáng kể trên tập mẫu (dưới 1 giây).
 - Với tập dữ liệu lớn (ratings.dat), cần tối ưu thêm bằng cách sử dụng chỉ mục và truy vấn hàng loạt.
- Lợi ích kiểm thử mẫu: Tập test_data.dat (20 dòng) cho phép kiểm tra chi tiết từng bản ghi và phân vùng, giúp phát hiện lỗi sớm trước khi áp dụng trên tập dữ liệu lớn.

5. Tổng kết

Kiểm thử trên tập dữ liệu mẫu `test_data.dat` xác nhận các hàm trong `Interface.py` hoạt động đúng:

- `loadratings`: Nạp đúng 20 bản ghi.
- `rangepartition` và `roundrobinpartition`: Tạo phân vùng đúng, thỏa mãn tính hoàn thiện, không giao nhau, và tái tạo.
- `rangeinsert` và `roundrobininsert`: Chèn bản ghi mới vào đúng phân vùng.
- SQL kiểm tra: Các câu lệnh SQL xác minh số bản ghi, tính đúng đắn của phân vùng, và bản ghi chèn thêm.



```
Code > execution_times.log
1 2025-06-03 20:47:27 - setup_metadata executed in 0.142099 seconds
2 2025-06-03 20:47:35 - loadratings executed in 8.248927 seconds
3 2025-06-03 20:47:36 - setup_metadata executed in 0.001317 seconds
4 2025-06-03 20:47:52 - rangepartition executed in 16.247328 seconds
5 2025-06-03 20:47:58 - rangeinsert executed in 0.006930 seconds
6 2025-06-03 20:47:58 - setup_metadata executed in 0.170933 seconds
7 2025-06-03 20:48:07 - loadratings executed in 9.371285 seconds
8 2025-06-03 20:48:07 - setup_metadata executed in 0.000637 seconds
9 2025-06-03 20:48:33 - roundrobinpartition executed in 25.414093 seconds
10 2025-06-03 20:48:43 - roundrobininsert executed in 0.044361 seconds
11
12
13
14 2025-06-03 20:55:13 - setup_metadata executed in 0.006470 seconds
15 2025-06-03 20:55:21 - loadratings executed in 8.186412 seconds, loaded 10000054 rows
16 2025-06-03 20:55:22 - setup_metadata executed in 0.000570 seconds
17 2025-06-03 20:55:37 - rangepartition executed in 15.057834 seconds with 5 partitions
18 2025-06-03 20:55:42 - rangeinsert executed in 0.001566 seconds
19 2025-06-03 20:55:42 - setup_metadata executed in 0.149348 seconds
20 2025-06-03 20:55:51 - loadratings executed in 9.073567 seconds, loaded 10000054 rows
21 2025-06-03 20:55:51 - setup_metadata executed in 0.000364 seconds
22 2025-06-03 20:56:15 - roundrobinpartition executed in 24.371557 seconds with 5 partitions
23 2025-06-03 20:56:26 - roundrobininsert executed in 0.001680 seconds
```

Hình 12: Thời gian thực thi của các hàm

- Đề xuất hướng phát triển:
 - Range Partitioning: Phân tích phân bố Rating để điều chỉnh ranh giới, đảm bảo số bản ghi đồng đều hơn.
 - Hiệu suất: Áp dụng chỉ mục trên UserID hoặc MovieID để hỗ trợ truy vấn ngoài Rating.
 - Kiểm thử lớn: Thêm các thao tác xử lý, tìm cách tối ưu hóa để tối ưu hiệu suất và thời gian thực thi trên tập dữ liệu lớn.