

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



BÁO CÁO BÀI TẬP LỚN

ĐỀ TÀI:

**PHÂN MẢNH NGANG DỮ LIỆU SỬ DỤNG THUẬT TOÁN
ROUND_ROBIN VÀ RANGE**

Giảng viên hướng dẫn : Kim Ngọc Bách

Thành viên nhóm	Nguyễn Khắc Trường	B22DCCN884
	Nguyễn Thế Thịnh	B22DCCN834
	Tạ Trường Vũ	B22DCCN918

HÀ NỘI 2025

LỜI CẢM ƠN

Trước tiên, chúng em xin được bày tỏ lòng biết ơn sâu sắc đến Học viện Công nghệ Bưu chính Viễn thông và Khoa Công nghệ thông tin 1 đã tạo điều kiện thuận lợi để sinh viên được học tập và rèn luyện trong môn học Cơ sở dữ liệu phân tán. Đây là một môn học không chỉ trang bị cho chúng em nền tảng kiến thức quan trọng về quản lý và xử lý dữ liệu trong môi trường phân tán mà còn giúp chúng em tiếp cận các bài toán thực tế trong lĩnh vực công nghệ thông tin.

Chúng em cũng xin gửi lời cảm ơn chân thành nhất đến thầy Kim Ngọc Bách – người thầy đã trực tiếp giảng dạy và hướng dẫn chúng em trong suốt quá trình học tập và thực hiện dự án của môn học. Thông qua các buổi học lý thuyết trên lớp, kết hợp với phần thực hành và bài tập lớn, chúng em đã có cơ hội củng cố kiến thức và ứng dụng các phương pháp xử lý dữ liệu phân tán vào các tình huống cụ thể. Sự tận tình, tâm huyết và trách nhiệm của thầy không chỉ giúp chúng em hiểu sâu hơn về môn học mà còn truyền cảm hứng và động lực để chúng em hoàn thành dự án một cách hiệu quả.

Quá trình học và làm dự án đã mang lại cho chúng em những trải nghiệm quý báu, giúp chúng em rèn luyện kỹ năng làm việc nhóm, tư duy phân tích và giải quyết vấn đề, đồng thời nâng cao khả năng áp dụng lý thuyết vào thực tế. Đây chắc chắn là hành trang quan trọng cho công việc và sự nghiệp của chúng em sau này.

Một lần nữa, chúng em xin được bày tỏ lòng biết ơn sâu sắc và kính chúc thầy luôn mạnh khỏe, hạnh phúc và tiếp tục đồng hành, hỗ trợ nhiều thế hệ sinh viên trên con đường học tập và phát triển.

MỤC LỤC

MỤC LỤC	3
BẢNG PHÂN CÔNG CÔNG VIỆC	5
DANH MỤC CÁC HÌNH VẼ.....	6
I. GIỚI THIỆU VỀ BÀI TOÁN.....	8
1.1 Tổng quan về bài toán.....	8
1.2 Ý nghĩa của việc phân mảnh.....	8
1.3 Mô tả dữ liệu đầu vào.....	8
1.4 Yêu cầu của bài toán	9
1.4.1 Yêu cầu kỹ thuật.....	9
1.4.2 Yêu cầu xây dựng bài toán	9
II. CÀI ĐẶT MÔI TRƯỜNG.....	10
2.1 Lựa chọn môi trường.....	10
2.2 Lựa chọn thư viện	10
III. THIẾT KẾ.....	11
3.1 Hàm load_ratings().....	11
3.1.1 Hàm create_db()	13
3.1.1.1 Hàm getopenconnection().....	15
3.2 Hàm rangepartition().....	16
3.3 Hàm rangeinsert().....	19
3.3.1 Hàm count_partitions().....	21
3.4 Hàm roundrobinpartition()	22
3.5 Hàm roundrobininsert().....	26
IV. KIỂM THỬ	29
4.1 Kiểm thử trên bộ test.....	29

4.1.1	Miêu tả bộ dữ liệu test.....	29
4.1.2	Cấu hình bộ dữ liệu test.....	29
4.1.3	Kết quả trên bộ test.....	29
4.1.3.1	Kết quả hàm load_ratings().....	29
4.1.3.2	Kết quả hàm rangepartitions()	31
4.1.3.3	Kết quả hàm rangeinsert().....	34
4.1.3.4	Kết quả hàm roundrobinpartition()	37
4.1.3.5	Kết quả hàm roundrobininsert().....	41
4.2	Kiểm thử trên bộ dữ liệu thực tế	44
4.2.1	Miêu tả bộ dữ liệu thực tế.....	44
4.2.2	Cấu hình bộ dữ liệu test.....	44
4.2.3	Kết quả trên bộ dữ liệu thực tế.....	44
4.2.3.1	Kết quả hàm load_ratings	44
4.2.3.2	Kết quả hàm rangepartitions()	45
4.2.3.3	Kết quả hàm rangeinsert().....	52
4.2.3.4	Kết quả hàm roundrobinpartition()	54
4.2.3.5	Kết quả hàm roundrobininsert().....	60
V.	KẾT LUẬN	64

BẢNG PHÂN CÔNG CÔNG VIỆC

Tên thành viên	Công việc thực hiện
Nguyễn Khắc Trường	<ul style="list-style-type: none">▪ Hoàn thiện phần tải dữ liệu (Load_ratings)▪ Viết báo cáo
Nguyễn Thế Thịnh	<ul style="list-style-type: none">▪ Hoàn thiện thuật toán phân mảnh theo khoảng (Range)
Tạ Trường Vũ	<ul style="list-style-type: none">▪ Hoàn thiện thuật toán phân mảnh vòng tròn (Round_Robin)

DANH MỤC CÁC HÌNH VẼ

Hình 4.1. Log thể hiện function load_ratings đã pass.....	29
Hình 4.2. Kết quả bảng ratings trong cơ sở dữ liệu.	30
Hình 4.3. Log thể hiện function rangepartitions đã pass.	31
Hình 4.4. Kết quả bảng range_part0 trong cơ sở dữ liệu.....	31
Hình 4.5. Kết quả bảng range_part1 trong cơ sở dữ liệu.....	32
Hình 4.6. Kết quả bảng range_part2 trong cơ sở dữ liệu.....	33
Hình 4.7. Kết quả bảng range_part3 trong cơ sở dữ liệu.....	33
Hình 4.8. Kết quả bảng range_part4 trong cơ sở dữ liệu.....	34
Hình 4.9. Log thể hiện function rangepartitions đã pass.	35
Hình 4.10. Kết quả bảng range_part2 trong cơ sở dữ liệu.....	35
Hình 4.11. Kết quả bảng ratings trong cơ sở dữ liệu.	36
Hình 4.12. Log thể hiện function roundrobinpartition đã pass.	37
Hình 4.13. Kết quả bảng rrobin_part0 trong cơ sở dữ liệu.....	37
Hình 4.14. Kết quả bảng rrobin_part1 trong cơ sở dữ liệu.....	38
Hình 4.15. Kết quả bảng rrobin_part2 trong cơ sở dữ liệu.....	39
Hình 4.16. Kết quả bảng rrobin_part3 trong cơ sở dữ liệu.....	39
Hình 4.17. Kết quả bảng rrobin_part4 trong cơ sở dữ liệu.....	40
Hình 4.18. Kết quả bảng metadata hay meta_rr robin_part_info trong cơ sở dữ liệu. ..	40
Hình 4.19. Log thể hiện function roundrobinpartition đã pass.	41
Hình 4.20. Kết quả bảng metadata hay meta_rr robin_part_info trong cơ sở dữ liệu. ..	42
Hình 4.21. Kết quả bảng rrobin_part0 trong cơ sở dữ liệu.....	43
Hình 4.22. Kết quả bảng ratings trong cơ sở dữ liệu.	43
Hình 4.23. Log thể hiện function load_ratings đã pass.....	44
Hình 4.24. Kết quả bảng ratings trong cơ sở dữ liệu.	45
Hình 4.25. Log thể hiện function rangepartitions đã pass.	46

Hình 4.26. Kết quả bảng range_part0 trong cơ sở dữ liệu.....	47
Hình 4.27. Kết quả bảng range_part1 trong cơ sở dữ liệu.....	48
Hình 4.28. Kết quả bảng range_part2 trong cơ sở dữ liệu.....	49
Hình 4.29. Kết quả bảng range_part3 trong cơ sở dữ liệu.....	50
Hình 4.30. Kết quả bảng range_part4 trong cơ sở dữ liệu.....	51
Hình 4.31. Log thể hiện function rangepartitions đã pass.	52
Hình 4.32. Kết quả bảng range_part2 trong cơ sở dữ liệu.....	53
Hình 4.33. Kết quả bảng ratings trong cơ sở dữ liệu.	53
Hình 4.34. Log thể hiện function roundrobinpartition đã pass.	54
Hình 4.35. Kết quả bảng rrobin_part0 trong cơ sở dữ liệu.....	55
Hình 4.36. Kết quả bảng rrobin_part1 trong cơ sở dữ liệu.....	56
Hình 4.37. Kết quả bảng rrobin_part2 trong cơ sở dữ liệu.....	57
Hình 4.38. Kết quả bảng rrobin_part3 trong cơ sở dữ liệu.....	58
Hình 4.39. Kết quả bảng rrobin_part4 trong cơ sở dữ liệu.....	59
Hình 4.40. Kết quả bảng metadata hay meta_rrobin_part_info trong cơ sở dữ liệu. ..	60
Hình 4.41. Log thể hiện function roundrobinpartition đã pass.	60
Hình 4.42. Kết quả bảng metadata hay meta_rrobin_part_info trong cơ sở dữ liệu. ..	61
Hình 4.43. Kết quả bảng rrobin_part4 trong cơ sở dữ liệu.....	62
Hình 4.44. Kết quả bảng ratings trong cơ sở dữ liệu.	63

I. GIỚI THIỆU VỀ BÀI TOÁN

1.1 Tổng quan về bài toán

Bài toán yêu cầu mô phỏng các phương pháp phân mảnh ngang dữ liệu trong hệ quản trị cơ sở dữ liệu quan hệ mã nguồn mở (PostgreSQL/MySQL) bao gồm phương pháp phân mảnh theo khoảng (Range partitioning) và phân mảnh vòng tròn (Round-Robin partitioning)

1.2 Ý nghĩa của việc phân mảnh

Phân mảnh dữ liệu mang lại các lợi ích sau:

- **Tăng hiệu suất truy vấn:** Dữ liệu được chia thành các bảng nhỏ hơn, giúp giảm thời gian truy vấn bằng cách chỉ truy cập vào các phân mảnh liên quan thay vì toàn bộ bảng lớn.
- **Dễ mở rộng hệ thống:** Phân mảnh cho phép triển khai hệ thống phân tán, dễ dàng thêm các máy chủ để lưu trữ các phân mảnh khác nhau, từ đó tăng khả năng mở rộng.
- **Hỗ trợ quản lý dữ liệu lớn:** Các bảng phân mảnh nhỏ hơn dễ quản lý, sao lưu và khôi phục hơn so với một bảng chứa toàn bộ dữ liệu.

1.3 Mô tả dữ liệu đầu vào

Dữ liệu đầu vào được lấy từ tập dữ liệu ratings.dat của MovieLens:

- **Quy mô dữ liệu:** 10 triệu đánh giá từ 72.000 người dùng cho 10.000 bộ phim.
- **Định dạng mỗi dòng:** UserID::MovieID::Rating::Timestamp.
 - **UserID:** Mã định danh người dùng (kiểu int).
 - **MovieID:** Mã định danh bộ phim (kiểu int).
 - **Rating:** Điểm đánh giá từ 0 đến 5.
 - **Timestamp:** Thời gian đánh giá (không dùng trong bài toán này).

1.4 Yêu cầu của bài toán

1.4.1 Yêu cầu kỹ thuật

Bài toán có các yêu cầu về kỹ thuật như sau:

- Sử dụng Python 3.12.x trên Ubuntu hoặc Windows 10.
- Sử dụng PostgreSQL hoặc MySQL.
- Không đóng kết nối cơ sở dữ liệu bên trong các hàm.
- Không hardcode tên cơ sở dữ liệu và tên file input.
- Lưu tên bảng phân mảnh theo đúng format yêu cầu (range_partX, roundrobin_partX).

1.4.2 Yêu cầu xây dựng bài toán

Bài toán yêu cầu xây dựng các hàm sau:

- **LoadRatings():** Đọc dữ liệu từ file ratings.dat và lưu vào bảng Ratings trong cơ sở dữ liệu.
- **Range_Partition():** Phân mảnh bảng Ratings thành N phân mảnh theo khoảng giá trị của thuộc tính Rating.
- **RoundRobin_Partition():** Phân mảnh bảng Ratings thành N phân mảnh theo phương pháp vòng tròn.
- **Range_Insert():** Chèn một bản ghi mới vào bảng Ratings và vào phân mảnh tương ứng dựa trên giá trị Rating.
- **RoundRobin_Insert():** Chèn một bản ghi mới vào bảng Ratings và vào phân mảnh ứng dựa trên vị trí vòng tròn hiện tại của chỉ số phân mảnh.

II. CÀI ĐẶT MÔI TRƯỜNG

2.1 Lựa chọn môi trường

Với yêu cầu bài toán trên, nhóm quyết định lựa chọn các thành phần môi trường sau:

- Hệ điều hành: Window 10.
- Ngôn ngữ lập trình: Python phiên bản 3.12.4.
- Hệ quản trị cơ sở dữ liệu: PostgreSQL.

2.2 Lựa chọn thư viện

Để dễ dàng cho việc phân mảnh, nhóm quyết định lựa chọn các thư viện hỗ trợ sau:

- psycopg2: Thư viện kết nối cơ sở dữ liệu PostgreSQL.
- sql từ psycopg2: Hỗ trợ xây dựng câu lệnh SQL an toàn, tránh lỗi injection.
- StringIO từ module io: Dùng để tách và xử lý chuỗi đầu vào khi đọc file dữ liệu.
- time: Đo thời gian thực thi khi phân mảnh để đánh giá hiệu suất.

III. THIẾT KẾ

3.1 Hàm load_ratings()

Ý nghĩa của hàm:

Hàm này dùng để ** nạp dữ liệu từ một file chứa rating (đánh giá phim) vào một bảng PostgreSQL có tên là ratingtablename.

Tham số đầu vào:

- ratingtablename: Tên bảng trong database để lưu dữ liệu ratings.
- ratingsfilepath: Đường dẫn tuyệt đối đến file chứa dữ liệu đánh giá (có định dạng: userid::movieid::rating::timestamp).
- openconnection: Một đối tượng kết nối đã mở (connection object) tới cơ sở dữ liệu PostgreSQL.

Nội dung hàm:

```
create_db(openconnection.get_dsn_parameters()['dbname'])
```

- Ý nghĩa: Gọi hàm create_db (sẽ được đề cập ở phần 3.1.1) để đảm bảo database đã tồn tại (tên database lấy từ openconnection), nếu database chưa tồn tại thì hàm sẽ tạo database mới.

```
con = openconnection  
cur = con.cursor()
```

- Ý nghĩa: Gán openconnection từ tham số đầu vào vào 1 biến con mới giúp thao tác trên connection ngắn hơn, sau đó tạo con trỏ (cursor) để thực thi các truy vấn SQL trên kết nối đã mở.

```
cur.execute("DROP TABLE IF EXISTS " + ratingtablename)
```

- Ý nghĩa: Xóa bảng với tên là biến ratingtablename ở tham số đầu vào nếu đã tồn tại để tránh conflict khi gọi nhiều lần.

```
cur.execute(  
    "CREATE TABLE " + ratingtablename + " (userid INTEGER,  
    movieid INTEGER, rating FLOAT)"  
)
```

- Ý nghĩa: Tạo bảng với tên là biến `ratingstablename` ở tham số đầu vào với schema sau:

- `UserID (int)`
- `MovieID (int)`
- `Rating (float)`

```
from io import StringIO
processed_data = StringIO()
```

- Ý nghĩa:

- Tạo `StringIO` object để chứa dữ liệu đã được xử lý.
- `StringIO` cho phép ta tạo một "file ảo" trong memory.
- `Copy_from` có thể đọc trực tiếp từ `StringIO` như đọc file thật.

```
with open(ratingsfilepath, 'r') as input_file:
    for line in input_file:
        line = line.strip() # Xóa ký tự xuống dòng và
        khoảng trắng
        if line: # Bỏ qua dòng trống
            # Tách dòng theo delimiter '::'
            parts = line.split '::')
            if len(parts) >= 3: # Đảm bảo có đủ 3 trường
                userid = parts[0]
                movieid = parts[1]
                rating = parts[2]
                # Chỉ lấy 3 trường cần thiết, bỏ qua
                timestamp (parts[3])
                # Ghi vào StringIO với tab separator (mặc
                định của copy_from)

processed_data.write(f"{userid}\t{movieid}\t{rating}\n")
```

- Ý nghĩa: Chuyển format từ "`userid::movieid::rating::timestamp`"

thành "`userid\tmovie_id\trating`" (tab-separated cho `copy_from`).

- Cách thực hiện: Duyệt qua từng dòng trong file dữ liệu, thực hiện tách từng phần trong xâu ra thành 4 phần tử của mảng (ví dụ `1::2::3::4` thành `[1,2,3,4]`) sau đó chuyển các phần tử vừa tạo (không lấy phần tử thứ 4 tức là `timestamp`) thành 1 xâu và ghi vào "file ảo" tạo ở bước trên (ví dụ `[1,2,3,4]` thành "`1 2 3`").

```
processed_data.seek(0)
```

- Ý nghĩa: Reset con trỏ StringIO về đầu để thực hiện ghi dữ liệu từ phần đầu của “file ảo”.

```
cur.copy_from(processed_data, ratingtablename,
columns=('userid', 'movieid', 'rating'))
```

- Ý nghĩa: Ghi dữ liệu từ “file ảo” vào bảng mà ta đã tạo trên theo thứ tự các cột userid -> movieid -> rating (ví dụ xâu “1 2 3” sẽ được trên như sau: 1 vào userid, 2 vào movieid, 3 vào rating).

```
processed_data.close()
```

- Ý nghĩa: Đóng “file ảo” để giải phóng memory khi đã không còn dùng đến.

```
con.commit()
cur.close()
```

- Ý nghĩa: Lưu thay đổi vào CSDL và đóng con trỏ.

3.1.1 Hàm create_db()

Ý nghĩa của hàm:

Hàm này dùng để tạo cơ sở dữ liệu mới với tên được truyền vào (vd: 'dds_assgn1') nếu nó chưa tồn tại.

Tham số đầu vào:

- dbname: Tên cơ sở dữ liệu

Nội dung hàm:

```
con = getopenconnection(dbname='postgres') # dùng
'postgres' để tạo các DB khác
```

- Ý nghĩa: Gọi đến hàm getopenconnection (sẽ được đề cập đến ở phần 3.1.1.1) để khởi tạo kết nối đến database với tên là 'postgres', đây là database có sẵn của postgres. Do để tạo 1 database mới trong postgres ta cần phải truy cập đến 1 database bất kỳ đang có sẵn mới có thể thực hiện lệnh nên ta sẽ kết nối đến database có sẵn của postgres.

```
con.set_isolation_level(psycopg2.extensions.ISOLATION_LEVEL_AUTOCOMMIT)
```

- o Ý nghĩa: Đặt chế độ giao dịch (transaction isolation level) của kết nối PostgreSQL sang AUTOCOMMIT có nghĩa là mỗi câu lệnh SQL sẽ được thực hiện và tự động commit ngay lập tức, thay vì đợi lệnh commit() thủ công từ người lập trình. Lý do phải đổi chế độ giao dịch này là PostgreSQL không cho phép lệnh CREATE DATABASE nằm trong một transaction.

```
cur = con.cursor()
```

- o Ý nghĩa: Tạo con trỏ (cursor) để thực thi các truy vấn SQL trên kết nối đã mở.

```
cur.execute(
    sql.SQL("SELECT COUNT(*) FROM pg_catalog.pg_database
WHERE datname = %s"),
    [dbname]
)

# Lấy kết quả: nếu count = 0 → database chưa tồn tại
count = cur.fetchone()[0]
```

- o Ý nghĩa: Tạo truy vấn đếm tất cả các database có tên giống với biến dbname ở tham số đầu vào. Mục đích của truy vấn này là kiểm tra xem database với tên là biến dbname đã tồn tại hay chưa.

```
if count == 0:
    # Tạo database mới với tên đã cho nếu chưa tồn tại
    cur.execute(sql.SQL("CREATE DATABASE
{}").format(sql.Identifier(dbname)))
    print(f"Đã tạo cơ sở dữ liệu mới: {dbname}")
else:
    # Nếu database đã tồn tại, thông báo cho người dùng
    print(f"Cơ sở dữ liệu '{dbname}' đã tồn tại. Không cần tạo lại.")
```

- o Ý nghĩa: Nếu database không tồn tại hay số lượng database đếm được ở câu lệnh phía trên = 0 thì hàm sẽ tạo 1 database mới và thông báo đã tạo cơ sở dữ

liệu mới. Ngược lại nếu nó đã tồn tại thì thông báo cơ sở dữ liệu đã tồn tại và không tạo mới nữa.

3.1.1.1 Hàm `getopenconnection()`

Ý nghĩa của hàm:

Hàm này tạo và trả về một kết nối (connection object) đến cơ sở dữ liệu PostgreSQL thông qua thư viện `psycopg2`.

Tham số đầu vào:

- `user`: tên người dùng PostgreSQL (mặc định là 'postgres' – user mặc định khi cài đặt).
- `password`: mật khẩu của user PostgreSQL.
- `dbname`: tên database muốn kết nối tới.

Trả về:

Một đối tượng kết nối (connection object) đã được mở tới cơ sở dữ liệu PostgreSQL.

Nội dung hàm:

```
return psycopg2.connect(  
    "dbname='" + dbname + "' user='" + user + "'  
    host='localhost' password='" + password + "'" +  
)
```

- Ý nghĩa: Kết nối đến cơ sở dữ liệu PostgreSQL thông qua thư viện `psycopg2` với các tham số:
 - `dbname`: tên cơ sở dữ liệu mà bạn muốn kết nối.
 - `user`: tên người dùng để xác thực.
 - `host`: địa chỉ máy chủ (ở đây mặc định là 'localhost' tức là máy tính của chính bạn).
 - `password`: mật khẩu đăng nhập.

3.2 Hàm rangepartition()

Ý nghĩa của hàm:

Hàm này dùng để phân mảnh bảng ratings thành N phân mảnh ngang theo giá trị cột rating. Phương pháp Range Partition chia dữ liệu dựa trên khoảng giá trị - mỗi phân mảnh chứa các bản ghi có rating trong một khoảng nhất định (ví dụ: 0-1, 1-2, 2-3, 3-4, 4-5).

Tham số đầu vào:

- ratingtablename: Tên bảng gốc chứa dữ liệu đánh giá người dùng.
- numberofpartitions: Số phân mảnh cần tạo (ví dụ: 5).
- openconnection: Đối tượng kết nối tới PostgreSQL (đã mở, không đóng trong hàm).

Nội dung hàm:

```
con = openconnection # Kết nối cơ sở dữ liệu đã mở sẵn
cur = con.cursor() # Tạo đối tượng thực thi truy vấn SQL
```

- Ý nghĩa: Gán openconnection từ tham số đầu vào vào 1 biến con mới giúp thao tác trên connection ngắn hơn, sau đó tạo con trỏ (cursor) để thực thi các truy vấn SQL trên kết nối đã mở.

```
delta = 5.0 / numberofpartitions # Độ rộng của mỗi phân mảnh
RANGE_TABLE_PREFIX = 'range_part' # Tiền tố tên bảng phân mảnh
sql_batch = "" # Ghép toàn bộ câu lệnh SQL để gửi 1 lần duy nhất
```

- Ý nghĩa: Tính toán khoảng phân chia và khởi tạo biến, cụ thể:
 - **delta**: Độ rộng của mỗi phân mảnh. Vì rating từ 0.0 đến 5.0 nên chia cho số phân mảnh. Ví dụ: 5 phân mảnh \rightarrow $\text{delta} = 5.0/5 = 1.0$ (mỗi phân mảnh rộng 1.0 đơn vị)

- **RANGE_TABLE_PREFIX**: Tiền tố tên bảng phân mảnh (range_part0, range_part1, ...)
- **sql_batch**: Chuỗi ghép tất cả câu lệnh SQL để gửi 1 lần duy nhất (tối ưu hiệu suất)

```

for i in range(numberofpartitions):
    # 1. Tính giá trị nhỏ nhất, lớn nhất trong phân mảnh
    thứ i
    min_val = i * delta # Giá trị nhỏ nhất trong phân mảnh
    thứ i
    max_val = min_val + delta # Giá trị lớn nhất trong phân
    mảnh thứ i
    # 2. Tạo tên bảng phân mảnh
    table_name = f"{RANGE_TABLE_PREFIX}{i}" # Tên bảng phân
    mảnh, ví dụ: range_part0

    # 3. Tạo schema cho bảng phân mảnh
    sql_batch += f"""
        CREATE TABLE {table_name} (
            userid INTEGER,
            movieid INTEGER,
            rating FLOAT
        );
    """

    # 4. Tạo điều kiện lọc rating phù hợp
    if i == 0:
        # Phân mảnh đầu tiên lấy cả [min, max]
        condition = f"rating >= {min_val} AND rating <=
    {max_val}"
    else:
        # Các phân mảnh sau: (min, max] để tránh trùng
        rating biên
        condition = f"rating > {min_val} AND rating <=
    {max_val}"

    # 5. Thêm câu lệnh chèn vào phân mảnh i
    sql_batch += f"""
        INSERT INTO {table_name} (userid, movieid, rating)
        SELECT userid, movieid, rating
        FROM {ratingstablename}
    """

```

```
WHERE {condition};  
"""
```

- Ý nghĩa: Duyệt qua từng phân mảnh để phân mảnh và tạo bảng, cụ thể:
 1. Tính giá trị nhỏ nhất, lớn nhất trong phân mảnh thứ i :
 - **min_val**: Giá trị rating nhỏ nhất trong phân mảnh thứ i (giả sử $\text{delta} = 1$ hay số phân mảnh = 5)
Phân mảnh 0: $\text{min_val} = 0 * 1.0 = 0.0$
Phân mảnh 1: $\text{min_val} = 1 * 1.0 = 1.0$
 - **max_val**: Giá trị rating lớn nhất trong phân mảnh thứ i (giả sử $\text{delta} = 1$ hay số phân mảnh = 5)
Phân mảnh 0: $\text{max_val} = 0.0 + 1.0 = 1.0$
Phân mảnh 1: $\text{max_val} = 1.0 + 1.0 = 2.0$
 2. Tạo tên bảng phân mảnh: Kết hợp tiền tố `range_part` với số thứ tự vòng lặp hiện tại (ví dụ vòng lặp 0 thì tên bảng hiện tại là `range_part0`)
 3. Tạo bảng với tên ở bước 2 với schema sau:
 - UserID (int)
 - MovieID (int)
 - Rating (float)
 4. Tạo điều kiện lọc rating
 - Phân mảnh đầu tiên ($i=0$): Lấy cả $[\text{min}, \text{max}]$ - bao gồm cả 2 biên.
Ví dụ: $\text{rating} \geq 0.0 \text{ AND } \text{rating} \leq 1.0$.
 - Các phân mảnh sau: Lấy $(\text{min}, \text{max}]$ - loại trừ biên trái để tránh trùng lặp.
Ví dụ: $\text{rating} > 1.0 \text{ AND } \text{rating} \leq 2.0$.
 - Lý do: Tránh rating biên (như 1.0, 2.0) bị chèn vào 2 phân mảnh cùng lúc.
 5. Thêm câu lệnh chèn vào phân mảnh i :
 - Lưu vào bảng đã tạo ở bước 3 với các dữ liệu từ bảng gốc đã tạo ở hàm `load_ratings` với điều kiện lọc ở bước 4.

```
cur.execute(sql_batch)
cur.close() # Đóng cursor
con.commit()
```

- o Ý nghĩa: Thực thi câu sql đã tạo ở bước trên đồng thời lưu thay đổi vào cơ sở dữ liệu và đóng con trỏ lệnh cursor.

3.3 Hàm rangeinsert()

Ý nghĩa của hàm:

Hàm này dùng để chèn một dòng dữ liệu mới vào bảng chính ratings và vào đúng phân mảnh range tương ứng. Đảm bảo dữ liệu được đồng bộ giữa bảng gốc và các phân mảnh.

Tham số đầu vào:

- ratingtablename: Tên bảng gốc chứa dữ liệu đánh giá người dùng.
- userid: ID của người dùng (user) thực hiện đánh giá.
- itemid: ID của bộ phim được đánh giá (trong hệ thống này itemid chính là movieid).
- rating: Điểm số đánh giá (giá trị từ 0.0 đến 5.0).
- openconnection: Đối tượng kết nối tới PostgreSQL (đã mở, không đóng trong hàm).

Nội dung hàm:

```
con = openconnection # Kết nối cơ sở dữ liệu đã mở sẵn
cur = con.cursor() # Tạo đối tượng thực thi truy vấn SQL
```

- o Ý nghĩa: Gán openconnection từ tham số đầu vào vào 1 biến con mới giúp thao tác trên connection ngắn hơn, sau đó tạo con trỏ (cursor) để thực thi các truy vấn SQL trên kết nối đã mở.

```
insert_main_table = f"""
    INSERT INTO {ratingtablename} (userid,
movieid, rating)
    VALUES ({userid}, {itemid}, {rating});
```

```
"""
cur.execute(insert_main_table)
```

- Ý nghĩa: Chèn vào bảng chính với tên bảng là biến `ratingtablename` ở tham số đầu vào của hàm với các tham số `userid`, `itemid`, `rating` ở tham số đầu vào của hàm

```
RANGE_TABLE_PREFIX = 'range_part' # Tiền tố bảng phân mảnh
numberofpartitions = count_partitions(RANGE_TABLE_PREFIX,
openconnection) # Đếm số phân mảnh hiện có
delta = 5.0 / numberofpartitions # Độ rộng của mỗi khoảng phân mảnh
```

- Ý nghĩa: Tạo tiền tố bảng phân mảnh và tính toán độ rộng mỗi phân mảnh (`delta`) như hàm `rangepartition`. Đồng thời gọi tới hàm `count_partitions` (sẽ được đề cập tới ở mục 3.3.1) để tính số phân mảnh hiện có.

```
index = int(rating / delta) # Xác định phân mảnh dựa trên
giá trị rating

# Nếu rating là biên chia (ví dụ: 2.0) và không phải phân
mảnh đầu tiên,
# thì trừ đi 1 để nó thuộc phân mảnh bên trái (đảm bảo
không bị trùng)
if rating % delta == 0 and index != 0:
    index = index - 1
```

- Ý nghĩa: Xác định chỉ số phân mảnh:
 - `index = int(rating / delta)`: Tính phân mảnh dựa trên giá trị `rating`.
Ví dụ: `rating=2.5`, `delta=1.0` → `index = int(2.5/1.0) = 2` → `range_part2`.
 - Xử lý trường hợp biên: Nếu `rating` là bội số của `delta` (như 2.0, 3.0) và không phải phân mảnh đầu tiên thì trừ đi 1 để đưa về phân mảnh bên trái (tránh trùng lặp).
Ví dụ: `rating=2.0` → `index=2`, nhưng trừ 1 → `index=1` (thuộc `range_part1`).

```
table_name = RANGE_TABLE_PREFIX + str(index) # Tên bảng
phân mảnh cần chèn
```

```
# Chèn dòng vào bảng phân mảnh tương ứng
cur.execute("insert into " + table_name + "(userid,
movieid, rating) values (" + str(userid) + "," +
str(itemid) + "," + str(rating) + ");")
```

- Ý nghĩa: Chèn dữ liệu vào phân mảnh tương ứng:
 - Nối tiền tố tên bảng với index đã tính được ở bước trên để tạo ra tên bảng phân mảnh `table_name`.
 - Chèn dữ liệu `userid`, `itemid`, `rating` ở tham số đầu vào của hàm vào bảng với tên `table_name`.

```
con.commit()
cur.close()
```

- Ý nghĩa: Lưu thay đổi vào CSDL và đóng con trỏ.

3.3.1 Hàm `count_partitions()`

Ý nghĩa của hàm:

Hàm này dùng để đếm số lượng bảng phân mảnh có tên bắt đầu với prefix trong cơ sở dữ liệu. Hỗ trợ các hàm khác xác định số phân mảnh hiện tại.

Tham số đầu vào:

- `prefix`: Tiền tố tên bảng cần đếm (ví dụ: 'range_part', 'rrobin_part')
- `openconnection`: Đối tượng kết nối tới PostgreSQL (đã mở, không đóng trong hàm).

Trả về:

Số lượng bảng phù hợp với tiền tố (tức là số phân mảnh hiện tại).

Nội dung hàm:

```
con = openconnection # Kết nối cơ sở dữ liệu đã mở sẵn
cur = con.cursor() # Tạo đối tượng thực thi truy vấn SQL
```

- Ý nghĩa: Gán openconnection từ tham số đầu vào vào 1 biến con mới giúp thao tác trên connection ngắn hơn, sau đó tạo con trỏ (cursor) để thực thi các truy vấn SQL trên kết nối đã mở.

```
cur.execute("select count(*) from pg_stat_user_tables where  
relname like " + "'" + prefix + "%';")  
count = cur.fetchone()[0] # Lấy giá trị đếm ra từ kết quả  
truy vấn
```

- Ý nghĩa: Truy vấn đếm bảng: Đếm số lượng bảng có tên giống với biến prefix được cung cấp tham số đầu vào.

```
cur.close()  
  
return count # Trả về số bảng phân mảnh
```

- Ý nghĩa: Đóng con trỏ và trả về kết quả là số bảng phân mảnh.

3.4 Hàm roundrobinpartition()

Ý nghĩa của hàm:

Hàm này dùng để phân mảnh bảng ratings thành N phân mảnh ngang theo phương pháp Round-Robin.

Mỗi bản ghi từ bảng gốc được đánh số thứ tự (row number) theo thứ tự đọc và được phân phối tuần tự vào các bảng phân mảnh (ví dụ: record 0 vào part0, record 1 vào part1, record 2 vào part2, rồi quay lại record 3 vào part0, ...).

Phương pháp này đảm bảo các phân mảnh có kích thước gần bằng nhau, đồng đều. Đồng thời, hàm cũng tạo bảng metadata để lưu thông tin về trạng thái phân mảnh, phục vụ cho việc chèn dữ liệu tiếp theo.

Tham số đầu vào:

- ratingstablename: Tên bảng gốc chứa dữ liệu đánh giá người dùng.
- numberofpartitions: Số phân mảnh cần tạo.

- openconnection: Đối tượng kết nối tới PostgreSQL (đã mở, không đóng trong hàm).

Nội dung hàm:

```
con = openconnection # Kết nối cơ sở dữ liệu đã mở sẵn
cur = con.cursor() # Tạo đối tượng thực thi truy vấn SQL
```

- Ý nghĩa: Gán openconnection từ tham số đầu vào vào 1 biến con mới giúp thao tác trên connection ngắn gọn hơn, sau đó tạo con trỏ (cursor) để thực thi các truy vấn SQL trên kết nối đã mở.

```
RROBIN_TABLE_PREFIX = 'rrobin_part'
```

- Ý nghĩa: Tạo tiền tố tên bảng phân mảnh (ví dụ: rrobin_part0, rrobin_part1, ...).

```
create_partition_sql = ""
for i in range(numberofpartitions):
    # Lệnh xóa bảng phân mảnh cũ nếu nó đã tồn tại để đảm
    bảo hàm có thể chạy lại nhiều lần.
    create_partition_sql += f"DROP TABLE IF EXISTS
{RROBIN_TABLE_PREFIX}{i};\n"
    # Lệnh tạo bảng phân mảnh mới với cấu trúc (schema)
    được định nghĩa trước.
    create_partition_sql += f"CREATE TABLE
{RROBIN_TABLE_PREFIX}{i} (userid INT, movieid INT, rating
FLOAT);\n"

# Thực thi tất cả các câu lệnh tạo/xóa bảng trong một lần
gọi duy nhất.
cur.execute(create_partition_sql)
```

- Ý nghĩa: Xóa các bảng phân mảnh (nếu đã tồn tại) và tạo các bảng phân mảnh mới.
- Cách thực hiện: Lặp qua từng index của số phân mảnh, xóa bảng có tên với tiền tố là biến được đề cập ở hàm trên RROBIN_TABLE_PREFIX + với index hiện thời (ví dụ: Index hiện tại là 1 thì xóa bảng “rrobin_part1”) sau đó tạo lại 1 bảng mới với tên tương tự và có cấu trúc là:
 - UserID (int)

- MovieID (int)
- Rating (float)

```
cur.execute(f"""
    WITH numbered AS (
        SELECT userid, movieid, rating,
               ROW_NUMBER() OVER () - 1 AS row_num
        FROM {ratingstablename}
    )
    SELECT *
    INTO TEMP temp_numbered_rows
    FROM numbered;
""")
```

○ Ý nghĩa: Tạo bảng tạm với đánh số thứ tự dòng:

- ROW_NUMBER() OVER () - 1: Gán số thứ tự cho mỗi dòng, bắt đầu từ 0.
- TEMP temp_numbered_rows: Tạo bảng tạm chỉ tồn tại trong phiên hiện tại.
- Mục đích: Tránh phải tính toán lại ROW_NUMBER() nhiều lần cho từng phân mảnh.

```
insert_all_sql = ""
for i in range(numberofpartitions):
    # Logic round robin: Dòng nào có (row_num %
    # numberofpartitions) bằng với chỉ số phân mảnh 'i'
    # thì sẽ được chèn vào phân mảnh thứ 'i'.
    # PostgreSQL hỗ trợ hàm MOD(a, b) tương đương với toán
    # tử a % b.
    insert_all_sql += f"""
        INSERT INTO {RROBIN_TABLE_PREFIX}{i} (userid, movieid,
        rating)
        SELECT userid, movieid, rating FROM temp_numbered_rows
        WHERE MOD(row_num, {numberofpartitions}) = {i};
        """
cur.execute(insert_all_sql)
```

○ Ý nghĩa: Tạo câu lệnh chèn dữ liệu vào các phân mảnh:

- Logic Round-Robin: Dòng có $\text{row_num} \% \text{numberofpartitions} = i$ sẽ được chèn vào phân mảnh thứ i . Việc chia dư sẽ giúp các bản ghi dữ liệu sẽ chỉ

nằm trong giới hạn số lượng phân mảnh, giả sử số phân mảnh là 3 và cột ta đang xét là 7 thì nó sẽ nằm trong bảng thứ $7\%3=1$ và $1<3$.

- Ví dụ: Với 3 phân mảnh:

Dòng 0, 3, 6, 9... → phân mảnh 0.

Dòng 1, 4, 7, 10... → phân mảnh 1.

Dòng 2, 5, 8, 11... → phân mảnh 2.

```
cur.execute("DROP TABLE IF EXISTS meta_rrobin_part_info;")
cur.execute("""
    CREATE TABLE meta_rrobin_part_info (
        partition_number INT,
        no_of_partitions INT
    );
""")
```

- Ý nghĩa: Xóa bảng metadata (nếu đã tồn tại) và tạo lại bảng metadata để lưu trạng thái phân mảnh:

- `partition_number`: lưu chỉ số của phân mảnh gần nhất đã nhận dữ liệu insert.
- `no_of_partitions`: tổng số phân mảnh đang có.

```
cur.execute("SELECT MAX(row_num) + 1 FROM
temp_numbered_rows;") # Lấy tổng số dòng đã chèn vào bảng
tạm
total_rows = cur.fetchone()[0] or 0
last_partition = (total_rows - 1) % numberofpartitions if
total_rows > 0 else -1 # Chỉ số phân mảnh cuối cùng đã chèn
dữ liệu
```

- Ý nghĩa: Tính toán chỉ số của phân mảnh cuối cùng đã được chèn dữ liệu: Lấy tổng số dòng đã chèn vào bảng tạm chia dư cho số phân mảnh sẽ ra được bảng phân mảnh nào được chèn vào cuối cùng (ví dụ: Số lượng dữ liệu là 5 và số phân mảnh là 3 thì bảng phân mảnh cuối cùng được chèn là $[5-1]\%3=1$ tức là bảng 1 [số thứ tự bảng bắt đầu từ 0 nên phải -1])

```
cur.execute("DROP TABLE temp_numbered_rows;")
```

- Ý nghĩa: Xóa bảng tạm sau khi đã sử dụng xong để giải phóng tài nguyên trên server.

```
con.commit()
cur.close()
```

- o Ý nghĩa: Lưu thay đổi vào CSDL và đóng con trỏ.

3.5 Hàm roundrobininsert()

Ý nghĩa của hàm:

Hàm này dùng để chèn một dòng dữ liệu mới vào bảng chính ratings và vào đúng phân mảnh theo logic Round-Robin. Đảm bảo dữ liệu được đồng bộ giữa bảng gốc và các phân mảnh.

Tham số đầu vào:

- ratingtablename: Tên bảng gốc chứa dữ liệu đánh giá người dùng.
- userid: ID của người dùng (user) thực hiện đánh giá.
- itemid: ID của bộ phim được đánh giá (trong hệ thống này itemid chính là movieid).
- rating: Điểm số đánh giá (giá trị từ 0.0 đến 5.0).
- openconnection: Đối tượng kết nối tới PostgreSQL (đã mở, không đóng trong hàm).

Nội dung hàm:

```
con = openconnection # Kết nối cơ sở dữ liệu đã mở sẵn
cur = con.cursor() # Tạo đối tượng thực thi truy vấn SQL
```

- o Ý nghĩa: Gán openconnection từ tham số đầu vào vào 1 biến con mới giúp thao tác trên connection ngắn hơn, sau đó tạo con trỏ (cursor) để thực thi các truy vấn SQL trên kết nối đã mở.

```
RROBIN_TABLE_PREFIX = 'rrobin_part'
```

- o Ý nghĩa: Tạo tiền tố tên bảng phân mảnh (ví dụ: rrobin_part0, rrobin_part1, ...).

```
cur.execute("SELECT partition_number, no_of_partitions FROM
meta_rrobin_part_info;")
row = cur.fetchone()
```

```

if row is None:
    raise Exception("Bảng metadata cho Round-Robin chưa
được khởi tạo. Vui lòng chạy hàm roundrobinpartition
trước.")

partition_number = row[0] # Số phân mảnh đã insert gần
nhất
no_of_partitions = row[1] # Tổng số phân mảnh

```

- Ý nghĩa: Lấy thông tin trạng thái phân mảnh từ bảng metadata:
 - Kiểm tra xem bảng metadata có tồn tại không, nếu không thì thông báo rằng bảng chưa tồn tại và ta cần thực hiện hàm roundrobinpartition trước
 - Lấy số phân mảnh gần nhất đã insert và tổng số phân mảnh

```

insert_main_table = f"""
        INSERT INTO {ratingstablename} (userid,
movieid, rating)
        VALUES ({userid}, {itemid}, {rating});
        """
cur.execute(insert_main_table)

```

- Ý nghĩa: Chèn vào bảng chính với tên bảng là biến ratingstablename ở tham số đầu vào của hàm với các tham số userid, itemid, rating ở tham số đầu vào của hàm

```

next_partition = (partition_number + 1) % no_of_partitions
partition_table_name =
f"{RROBIN_TABLE_PREFIX}{next_partition}"

```

- Ý nghĩa: Tính toán chỉ số của phân mảnh tiếp theo theo nguyên tắc round robin và gán tên vào bảng ta cần chèn dữ liệu.
 - **Công thức:** $(\text{partition_number} + 1) \% \text{no_of_partitions}$.
 - **Ví dụ:** nếu lần cuối chèn vào phân mảnh 0 và có 5 phân mảnh, $(0 + 1) \% 5 = 1$. Lần chèn tiếp theo sẽ vào phân mảnh 1.

```

cur.execute(
    f"INSERT INTO {partition_table_name} (userid, movieid,
rating) VALUES (%s, %s, %s);",
    (userid, itemid, rating)
)

```

- Ý nghĩa: Chèn dữ liệu vào bảng phân mảnh tương ứng với tên bảng đã tính được ở bước trên.

```
cur.execute("UPDATE meta_rrobin_part_info SET  
partition_number = %s;", (next_partition,))
```

- Ý nghĩa: Cập nhật số thứ tự phân mảnh của bảng metadata.

```
con.commit()  
cur.close()
```

- Ý nghĩa: Lưu thay đổi vào CSDL và đóng con trỏ.

IV. KIỂM THỬ

4.1 Kiểm thử trên bộ test

4.1.1 Miêu tả bộ dữ liệu test

- **Quy mô dữ liệu:** 20 đánh giá.
- **Định dạng mỗi dòng:** UserID::MovieID::Rating::Timestamp.
 - **UserID:** Mã định danh người dùng (kiểu int).
 - **MovieID:** Mã định danh bộ phim (kiểu int).
 - **Rating:** Điểm đánh giá từ 0 đến 5.
 - **Timestamp:** Thời gian đánh giá.

4.1.2 Cấu hình bộ dữ liệu test

- Đối với thuật toán phân mảnh theo khoảng (RangePartition), số phân mảnh sẽ là 5 và index bảng bắt đầu trên dữ liệu là 0.
- Đối với thuật toán phân mảnh vòng tròn (RoundRobinPartition), số phân mảnh sẽ là 5 và index bảng bắt đầu trên dữ liệu là 0.

4.1.3 Kết quả trên bộ test

4.1.3.1 Kết quả hàm load_ratings()

Chạy file Assignment1 Test, kết quả log ra là function đã pass, cụ thể:

```
C:\Users\truong\AppData\Local\Programs\Python\Python312\python.exe "F:\code\python_learn\Range_Robin_fragmentation\Range-Robin_fragmentation_ddb\cd
Cơ sở dữ liệu 'dds_assgn1' đã tồn tại. Không cần tạo lại.
loadratings function pass!
```

Hình 4.1. Log thể hiện function load_ratings đã pass.

4.1.3.2 Kết quả hàm rangepartitions()

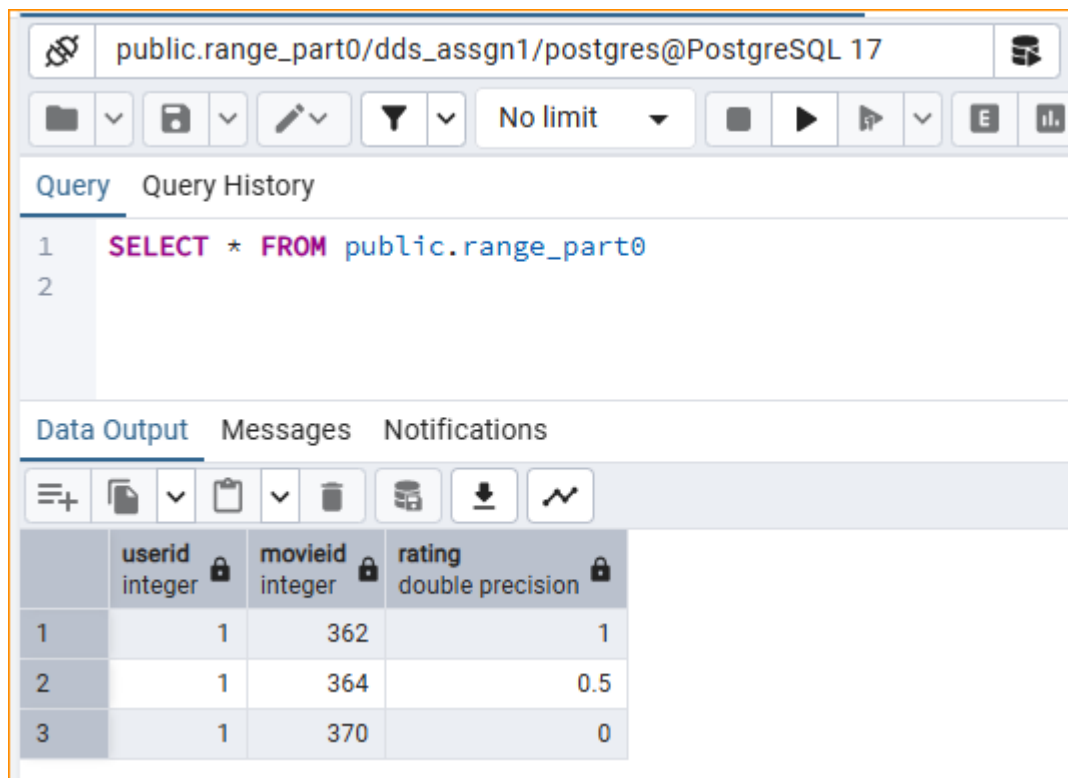
Chạy file Assignment1 Test, kết quả log ra là function đã pass, cụ thể:

```
C:\Users\truong\AppData\Local\Programs\Python\Python312\python.exe "F:\code\python_learn\Range_Robin_fragmentation\Ra
A database named "dds_assgn1" already exists
Cơ sở dữ liệu 'dds_assgn1' đã tồn tại. Không cần tạo lại.
loadratings function pass!

Chọn thuật toán phân mảnh để test (range / roundrobin): range

Testing RANGE partitioning...
rangepartition function pass!
Nhập Enter để tiếp tục sau khi đã kiểm tra rangepartition...
```

Hình 4.3. Log thể hiện function rangepartitions đã pass.



The screenshot shows a PostgreSQL query editor interface. The top bar displays the connection name 'public.range_part0/dds_assgn1/postgres@PostgreSQL 17'. Below the toolbar, the 'Query' tab is active, showing a SQL query: 'SELECT * FROM public.range_part0'. The 'Data Output' tab is also visible, showing the results of the query in a table format. The table has four columns: 'userid' (integer), 'movieid' (integer), and 'rating' (double precision). The results are as follows:

	userid integer	movieid integer	rating double precision
1	1	362	1
2	1	364	0.5
3	1	370	0

Hình 4.4. Kết quả bảng range_part0 trong cơ sở dữ liệu.

The screenshot shows a PostgreSQL query editor interface. The connection is 'public.range_part1/dds_assgn1/postgres@PostgreSQL 17'. The query is 'SELECT * FROM public.range_part1'. The results are displayed in a table with columns 'userid', 'movieid', and 'rating'.

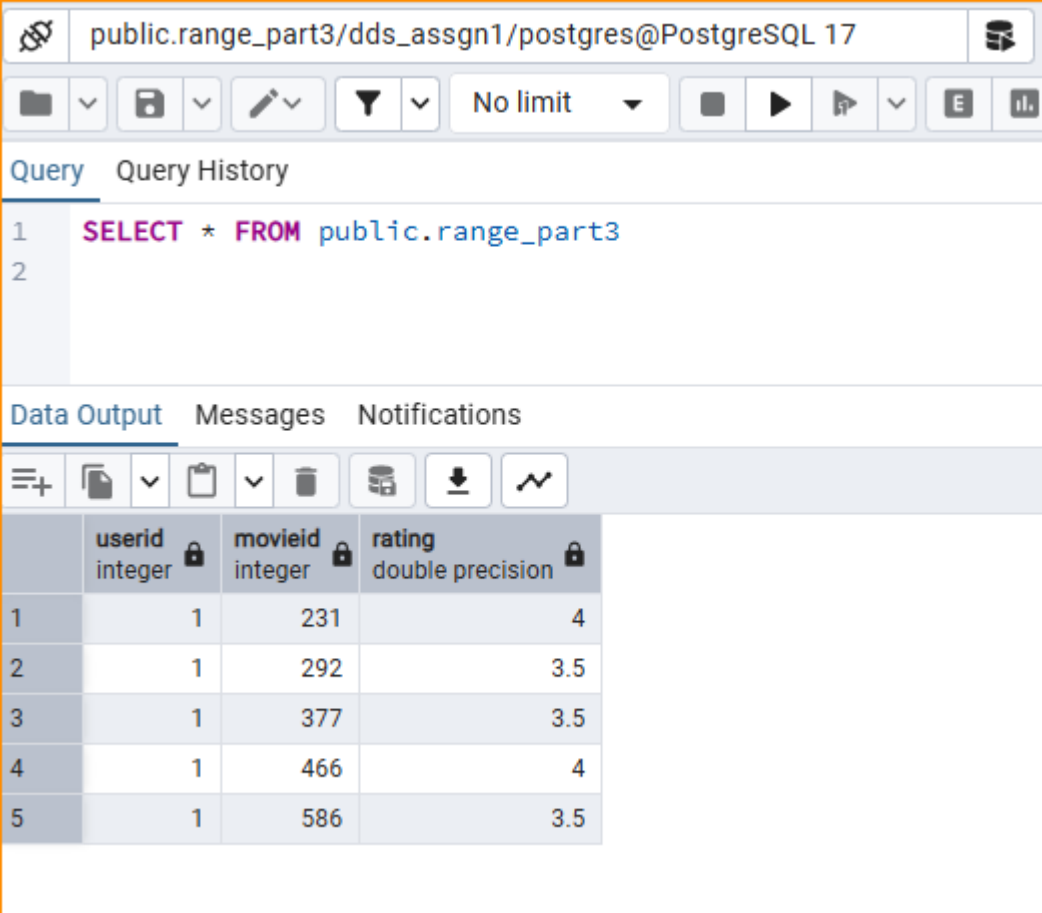
	userid integer	movieid integer	rating double precision
1	1	355	2
2	1	356	1.5
3	1	589	1.5

Hình 4.5. Kết quả bảng range_part1 trong cơ sở dữ liệu.

The screenshot shows a PostgreSQL query editor interface. The connection is 'public.range_part2/dds_assgn1/postgres@PostgreSQL 17'. The query is 'SELECT * FROM public.range_part2'. The results are displayed in a table with columns 'userid', 'movieid', and 'rating'.

	userid integer	movieid integer	rating double precision
1	1	316	3
2	1	329	2.5
3	1	520	2.5

Hình 4.6. Kết quả bảng range_part2 trong cơ sở dữ liệu.



The screenshot shows a PostgreSQL query editor interface. The top bar displays the connection string 'public.range_part3/dds_assgn1/postgres@PostgreSQL 17'. Below the bar is a toolbar with icons for file operations, query execution, and settings. The 'Query' tab is active, showing a SQL query: `SELECT * FROM public.range_part3`. The 'Data Output' tab is also visible, showing the results of the query in a table format. The table has four columns: 'userid' (integer), 'movieid' (integer), and 'rating' (double precision). The results are as follows:

	userid integer	movieid integer	rating double precision
1	1	231	4
2	1	292	3.5
3	1	377	3.5
4	1	466	4
5	1	586	3.5

Hình 4.7. Kết quả bảng range_part3 trong cơ sở dữ liệu.

The screenshot shows a PostgreSQL query editor interface. At the top, the connection string is 'public.range_part4/dds_assgn1/postgres@PostgreSQL 17'. Below the connection bar, there are tabs for 'Query' and 'Query History'. The 'Query' tab is active, showing a SQL query: `SELECT * FROM public.range_part4`. Below the query editor, there are tabs for 'Data Output', 'Messages', and 'Notifications'. The 'Data Output' tab is active, displaying a table with 6 rows and 3 columns: 'userid' (integer), 'movieid' (integer), and 'rating' (double precision). The table data is as follows:

	userid integer	movieid integer	rating double precision
1	1	122	5
2	1	185	4.5
3	1	420	5
4	1	480	5
5	1	539	5
6	1	588	5

Hình 4.8. Kết quả bảng range_part4 trong cơ sở dữ liệu.

4.1.3.3 Kết quả hàm rangeinsert()

Chạy file Assignment1 Test, dữ liệu được thêm vào sẽ là 1 bản ghi với thông tin:

- userid: 100
- movieid: 2
- rating: 3

Vậy sẽ chỉ có bảng ratings và range_part2 bị ảnh hưởng.

Kết quả log ra là function đã pass, cụ thể:

```

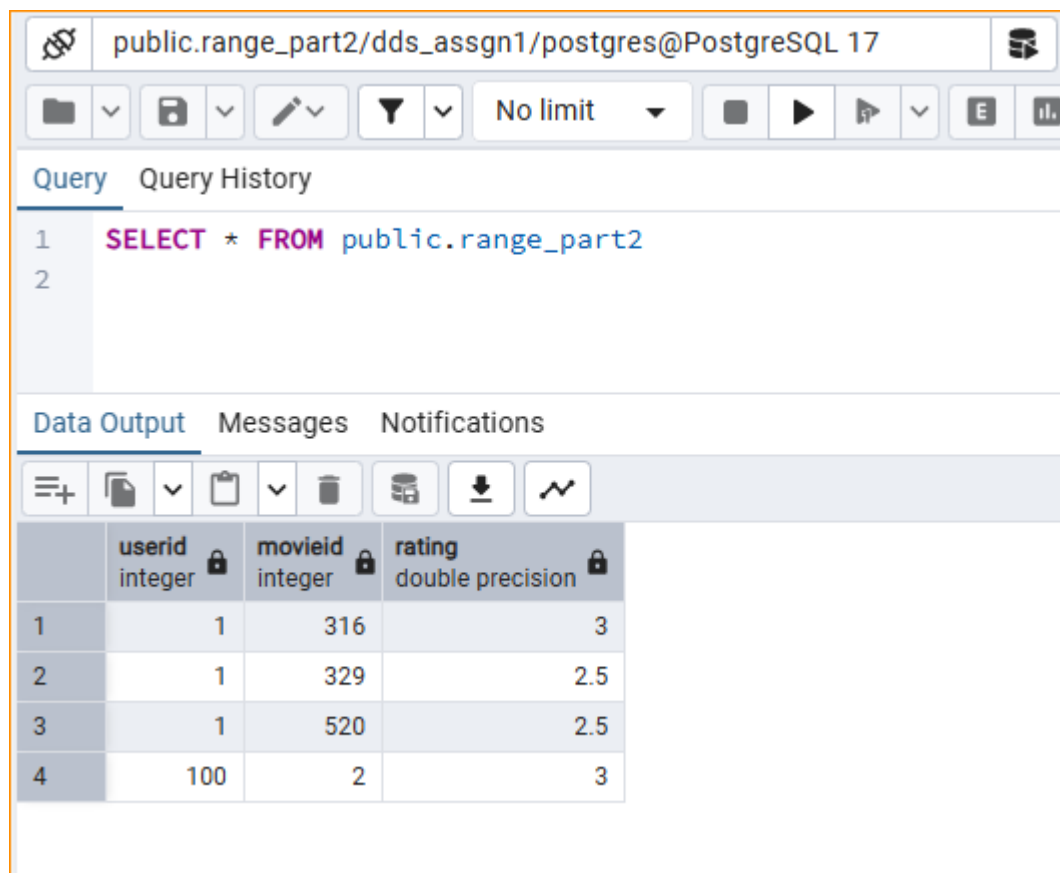
C:\Users\truong\AppData\Local\Programs\Python\Python312\python
A database named "dds_assgn1" already exists
Cơ sở dữ liệu 'dds_assgn1' đã tồn tại. Không cần tạo lại.
loadratings function pass!

Chọn thuật toán phân mảnh để test (range / roundrobin): range

Testing RANGE partitioning...
rangepartition function pass!
Nhập Enter để tiếp tục sau khi đã kiểm tra rangepartition...
rangeinsert function pass!

```

Hình 4.9. Log thể hiện function rangepartitions đã pass.



The screenshot shows a PostgreSQL client window with the connection string 'public.range_part2/dds_assgn1/postgres@PostgreSQL 17'. The query editor contains the SQL statement 'SELECT * FROM public.range_part2'. The 'Data Output' tab is active, displaying a table with 4 rows and 3 columns: 'userid' (integer), 'movieid' (integer), and 'rating' (double precision). The data is as follows:

	userid integer	movieid integer	rating double precision
1	1	316	3
2	1	329	2.5
3	1	520	2.5
4	100	2	3

Hình 4.10. Kết quả bảng range_part2 trong cơ sở dữ liệu.

4.1.3.4 Kết quả hàm roundrobinpartition()

Chạy file Assignment1Test, kết quả log ra là function đã pass, cụ thể:

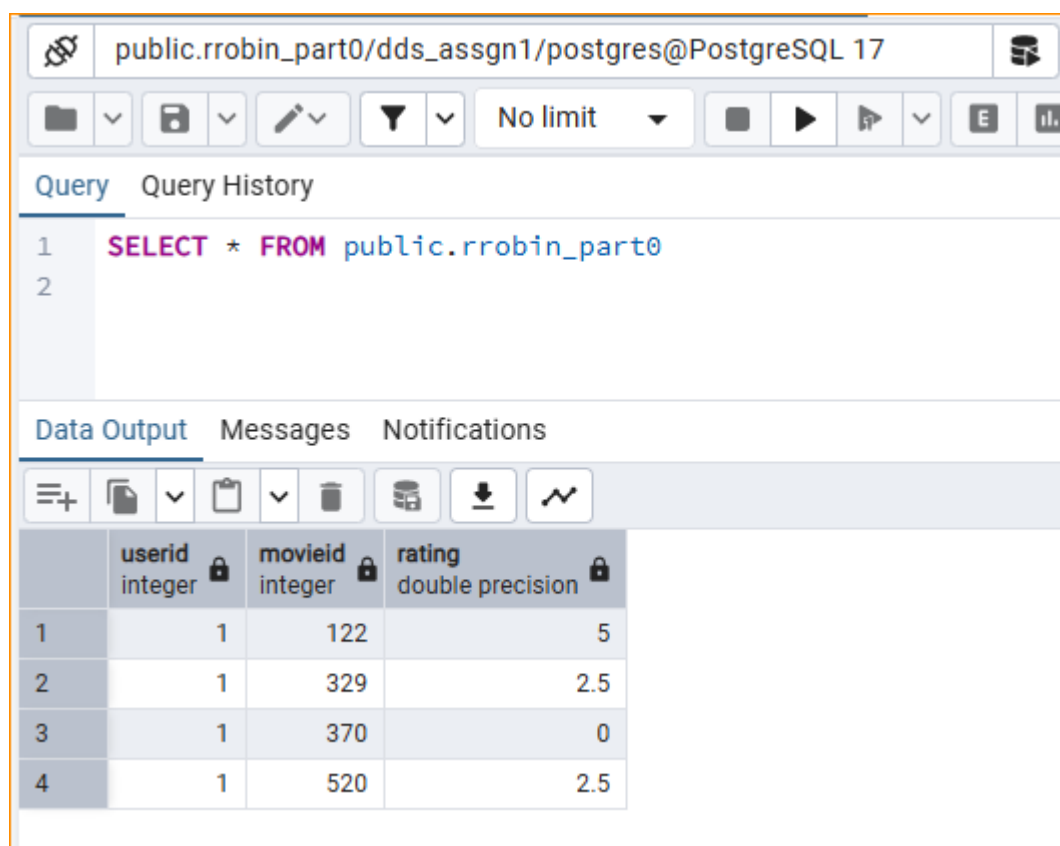
```
C:\Users\truong\AppData\Local\Programs\Python\Python312\python.exe "F:\code\python_learn\Range_Robin_fragmentation\Range-Robin_fragmentation_ddb\c
Cơ sở dữ liệu 'dds_assgn1' đã tồn tại. Không cần tạo lại.
loadratings function pass!

Tổng thời gian load xong: 0.102 giây

Chọn thuật toán phân mảnh để test (range / roundrobin): roundrobin

Testing ROUND ROBIN partitioning...
roundrobinpartition function pass!
Nhập Enter để tiếp tục sau khi đã kiểm tra roundrobinpartition...
```

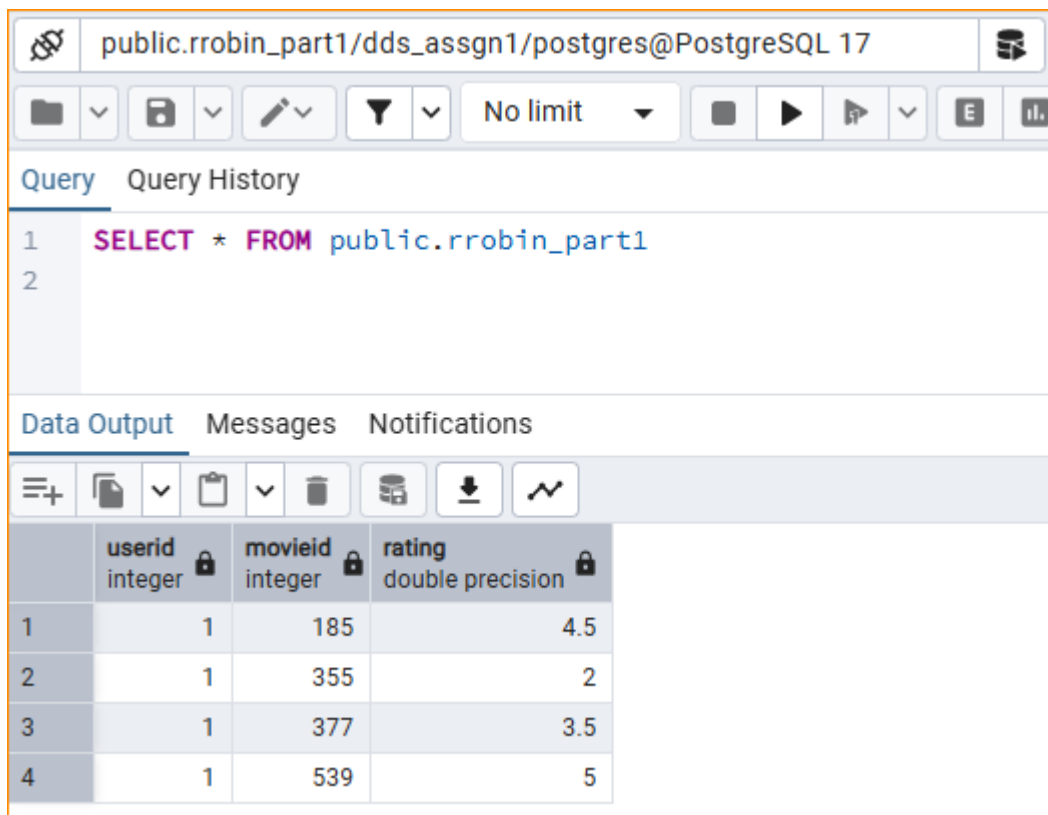
Hình 4.12. Log thể hiện function roundrobinpartition đã pass.



The screenshot shows a PostgreSQL query editor interface. The query bar at the top contains the query: `SELECT * FROM public.rrobin_part0`. Below the query bar, the 'Data Output' tab is selected, displaying a table with 4 rows and 3 columns: `userid` (integer), `movieid` (integer), and `rating` (double precision). The data is as follows:

	userid integer	movieid integer	rating double precision
1	1	122	5
2	1	329	2.5
3	1	370	0
4	1	520	2.5

Hình 4.13. Kết quả bảng rrobin_part0 trong cơ sở dữ liệu.



public.rrobin_part1/dds_assgn1/postgres@PostgreSQL 17

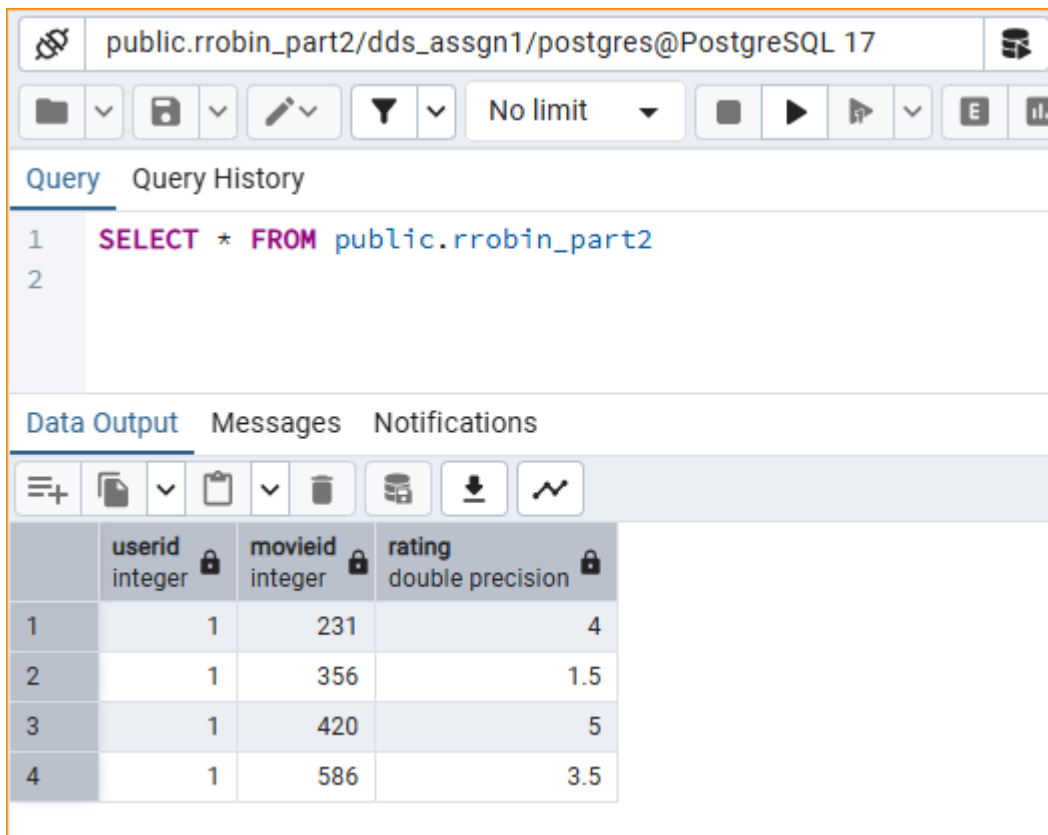
Query Query History

```
1 SELECT * FROM public.rrobin_part1
2
```

Data Output Messages Notifications

	userid integer	movieid integer	rating double precision
1	1	185	4.5
2	1	355	2
3	1	377	3.5
4	1	539	5

Hình 4.14. Kết quả bảng rrobin_part1 trong cơ sở dữ liệu.



public.rrobin_part2/dds_assgn1/postgres@PostgreSQL 17

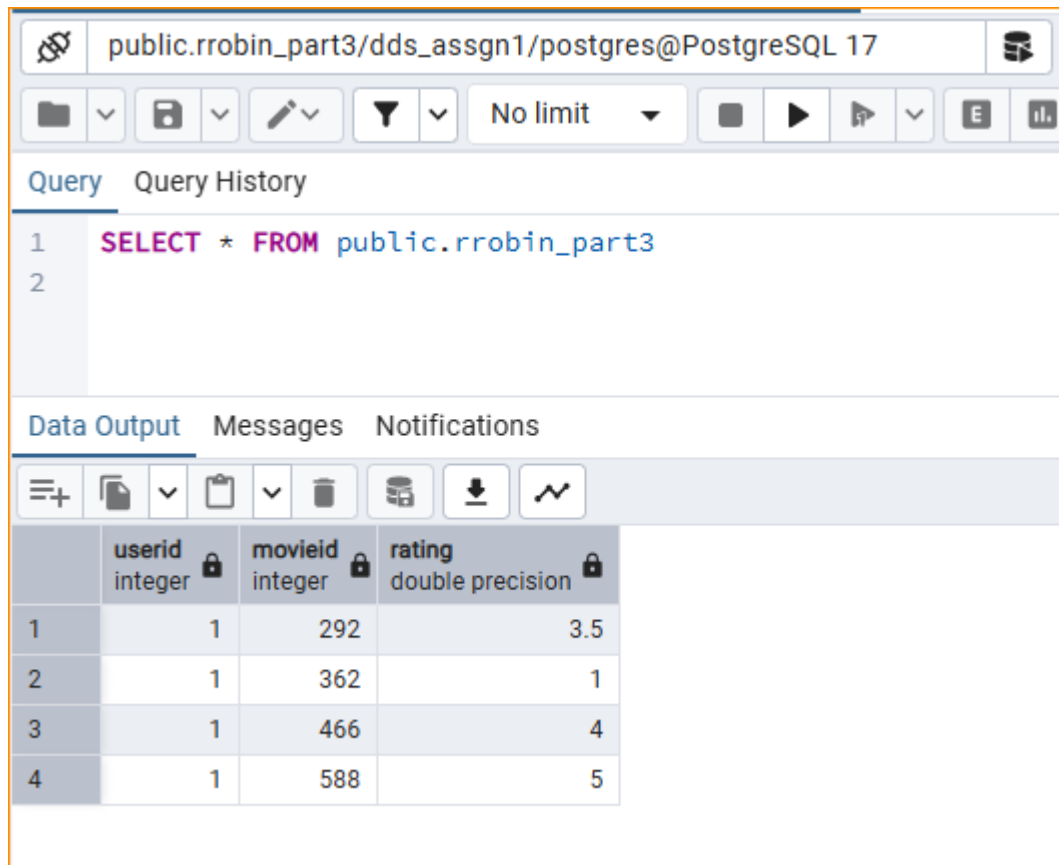
Query Query History

```
1 SELECT * FROM public.rrobin_part2
2
```

Data Output Messages Notifications

	userid integer	movieid integer	rating double precision
1	1	231	4
2	1	356	1.5
3	1	420	5
4	1	586	3.5

Hình 4.15. Kết quả bảng rrobin_part2 trong cơ sở dữ liệu.



The screenshot shows a PostgreSQL query editor interface. The top bar displays the connection string: `public.rrobin_part3/dds_assgn1/postgres@PostgreSQL 17`. Below the toolbar, the query editor shows the following SQL query:

```
1 SELECT * FROM public.rrobin_part3
2
```

The "Data Output" tab is active, showing the results of the query in a table format. The table has four columns: `userid` (integer), `movieid` (integer), and `rating` (double precision). The results are as follows:

	userid integer	movieid integer	rating double precision
1	1	292	3.5
2	1	362	1
3	1	466	4
4	1	588	5

Hình 4.16. Kết quả bảng rrobin_part3 trong cơ sở dữ liệu.

The screenshot shows a PostgreSQL query editor interface. The top bar indicates the connection to 'public.rrobin_part4/dds_assgn1/postgres@PostgreSQL 17'. Below the toolbar, the 'Query' tab is active, displaying a SQL query: `SELECT * FROM public.rrobin_part4`. The 'Data Output' tab is also visible, showing the results of the query in a table format. The table has four columns: 'userid' (integer), 'movieid' (integer), and 'rating' (double precision). The results are as follows:

	userid integer	movieid integer	rating double precision
1	1	316	3
2	1	364	0.5
3	1	480	5
4	1	589	1.5

Hình 4.17. Kết quả bảng rrobin_part4 trong cơ sở dữ liệu.

The screenshot shows a PostgreSQL query editor interface. The top bar indicates the connection to 'public.meta_rrobin_part_info/dds_assgn1/postgres@PostgreS...'. Below the toolbar, the 'Query' tab is active, displaying a SQL query: `SELECT * FROM public.meta_rrobin_part_info`. The 'Data Output' tab is also visible, showing the results of the query in a table format. The table has two columns: 'partition_number' (integer) and 'no_of_partitions' (integer). The results are as follows:

	partition_number integer	no_of_partitions integer
1	4	5

Hình 4.18. Kết quả bảng metadata hay meta_rrobin_part_info trong cơ sở dữ liệu.

4.1.3.5 Kết quả hàm roundrobininsert()

Chạy file Assignment1 Test, dữ liệu được thêm vào sẽ là 1 bản ghi với thông tin:

- userid: 100
- movieid: 1
- rating: 3

Vậy sẽ chỉ có bảng ratings, metadata (meta_rrobin_part_info) và rrobin_part0 bị ảnh hưởng.

Kết quả log ra là function đã pass, cụ thể:

```
C:\Users\truong\AppData\Local\Programs\Python\Python312\python.exe "F:\code\python_learn\Range_Robin_fra
Cơ sở dữ liệu 'dds_assgn1' đã tồn tại. Không cần tạo lại.
loadratings function pass!

Tổng thời gian load xong: 0.102 giây

Chọn thuật toán phân mảnh để test (range / roundrobin): roundrobin

Testing ROUND ROBIN partitioning...
roundrobinpartition function pass!
Nhập Enter để tiếp tục sau khi đã kiểm tra roundrobinpartition...
roundrobininsert function pass!
Press enter to Delete all tables? |
```

Hình 4.19. Log thể hiện function roundrobinpartition đã pass.

The screenshot shows a PostgreSQL query editor interface. The top bar indicates the connection to 'public.meta_rrabin_part_info/dds_assgn1/postgres@PostgreS...'. Below the toolbar, the 'Query' tab is active, displaying a SQL query: `SELECT * FROM public.meta_rrabin_part_info`. The 'Data Output' tab is also visible, showing the results of the query in a table format. The table has two columns: 'partition_number' (integer) and 'no_of_partitions' (integer). The results show one row with 'partition_number' 0 and 'no_of_partitions' 5.

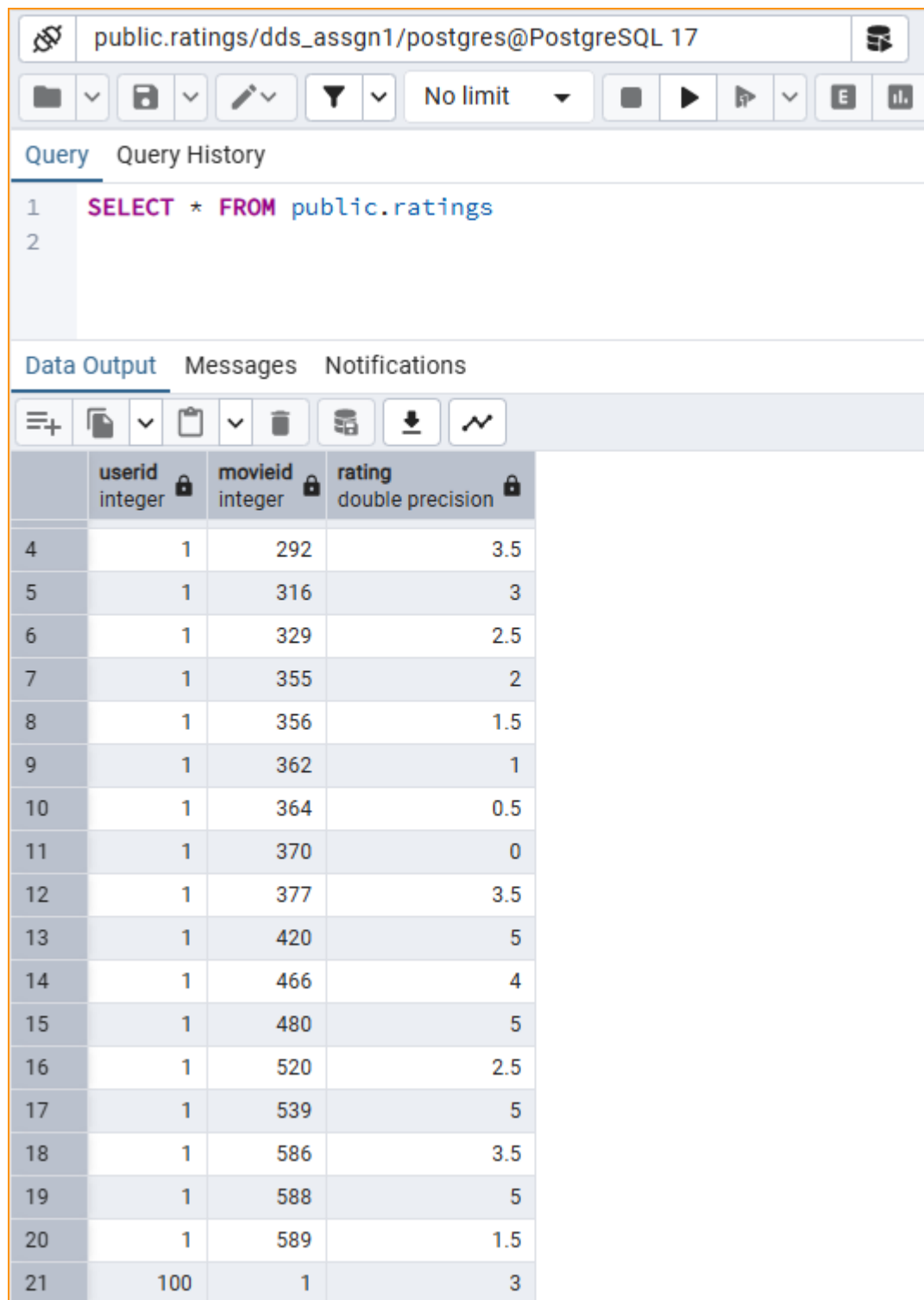
	partition_number integer	no_of_partitions integer
1	0	5

Hình 4.20. Kết quả bảng metadata hay meta_rrabin_part_info trong cơ sở dữ liệu.

The screenshot shows a PostgreSQL query editor interface. The top bar indicates the connection to 'public.rrabin_part0/dds_assgn1/postgres@PostgreSQL 17'. Below the toolbar, the 'Query' tab is active, displaying a SQL query: `SELECT * FROM public.rrabin_part0`. The 'Data Output' tab is also visible, showing the results of the query in a table format. The table has three columns: 'userid' (integer), 'movieid' (integer), and 'rating' (double precision). The results show five rows of data.

	userid integer	movieid integer	rating double precision
1	1	122	5
2	1	329	2.5
3	1	370	0
4	1	520	2.5
5	100	1	3

Hình 4.21. Kết quả bảng rrobin_part0 trong cơ sở dữ liệu.



The screenshot shows a PostgreSQL query editor interface. The top bar indicates the connection to 'public.ratings/dds_assgn1/postgres@PostgreSQL 17'. Below the toolbar, the 'Query' tab is active, displaying the SQL query: `SELECT * FROM public.ratings`. The 'Data Output' tab is also visible, showing the results of the query in a table format. The table has four columns: 'userid' (integer), 'movieid' (integer), and 'rating' (double precision). The results are displayed as a list of rows, numbered 4 to 21. The last row (21) shows a rating of 3 for user 100 and movie 1.

	userid integer	movieid integer	rating double precision
4	1	292	3.5
5	1	316	3
6	1	329	2.5
7	1	355	2
8	1	356	1.5
9	1	362	1
10	1	364	0.5
11	1	370	0
12	1	377	3.5
13	1	420	5
14	1	466	4
15	1	480	5
16	1	520	2.5
17	1	539	5
18	1	586	3.5
19	1	588	5
20	1	589	1.5
21	100	1	3

Hình 4.22. Kết quả bảng ratings trong cơ sở dữ liệu.

4.2 Kiểm thử trên bộ dữ liệu thực tế

4.2.1 Miêu tả bộ dữ liệu thực tế

- **Quy mô dữ liệu:** 10 triệu đánh giá từ 72.000 người dùng cho 10.000 bộ phim.
- **Định dạng mỗi dòng:** UserID::MovieID::Rating::Timestamp.
 - **UserID:** Mã định danh người dùng (kiểu int).
 - **MovieID:** Mã định danh bộ phim (kiểu int).
 - **Rating:** Điểm đánh giá từ 0 đến 5.
 - **Timestamp:** Thời gian đánh giá.

4.2.2 Cấu hình bộ dữ liệu test

- Đối với thuật toán phân mảnh theo khoảng (RangePartition), số phân mảnh sẽ là 5 và index bảng bắt đầu trên dữ liệu là 0.
- Đối với thuật toán phân mảnh vòng tròn (RoundRobinPartition), số phân mảnh sẽ là 5 và index bảng bắt đầu trên dữ liệu là 0.

4.2.3 Kết quả trên bộ dữ liệu thực tế

4.2.3.1 Kết quả hàm load_ratings

Chạy file Assignment1 Test:

- Kết quả log ra là function đã pass
- Tổng thời gian để chạy xong hàm hay load xong dữ liệu vào bảng ratings là 19 giây

```
C:\Users\truong\AppData\Local\Programs\Python\Python312\python.exe "F:\code\python_learn\Range_Robin_fragmentation\Range-Robin_fragmentation_ddb\cd
A database named "dds_assgn1" already exists
Cơ sở dữ liệu 'dds_assgn1' đã tồn tại. Không cần tạo lại.
loadratings function pass!

Tổng thời gian load xong: 18.726 giây
```

Hình 4.23. Log thể hiện function load_ratings đã pass.

public.ratings/dds_assgn1/postgres@PostgreSQL 17

Query Query History

```
1 SELECT * FROM public.ratings
2
```

Data Output Messages Notifications

	userid integer	movieid integer	rating double precision
10000037	71567	1805	4
10000038	71567	1833	3
10000039	71567	1876	3
10000040	71567	1909	2
10000041	71567	1917	4
10000042	71567	1920	4
10000043	71567	1982	1
10000044	71567	1983	1
10000045	71567	1984	1
10000046	71567	1985	1
10000047	71567	1986	1
10000048	71567	2012	3
10000049	71567	2028	5
10000050	71567	2107	1
10000051	71567	2126	2
10000052	71567	2294	5
10000053	71567	2338	2
10000054	71567	2384	2

Hình 4.24. Kết quả bảng ratings trong cơ sở dữ liệu.

4.2.3.2 Kết quả hàm rangepartitions()

Chạy file Assignment1 Test:

- Kết quả log ra là function đã pass.
- Tổng thời gian để chạy xong hàm rangepartitions hay phân mảnh theo đoạn là 24 giây.

```
C:\Users\truong\AppData\Local\Programs\Python\Python312\python.exe "F:\code\python_learn\Range-Robin_fragmentation\Range-Robin_fragmentation_ddb\c
A database named "dds_assgn1" already exists
Cơ sở dữ liệu 'dds_assgn1' đã tồn tại. Không cần tạo lại.
loadratings function pass!

Tổng thời gian load xong: 18.726 giây

Chọn thuật toán phân mảnh để test (range / roundrobin): range

Testing RANGE partitioning...
rangepartition function pass!

Tổng thời gian rangepartition: 24.388 giây
Nhập Enter để tiếp tục sau khi đã kiểm tra rangepartition...
```

Hình 4.25. Log thể hiện function rangepartitions đã pass.

public.range_part0/dds_assgn1/postgres@PostgreSQL 17

<

Hình 4.26. Kết quả bảng range_part0 trong cơ sở dữ liệu.

public.range_part1/dds_assgn1/postgres@PostgreSQL 17

<

Hình 4.27. Kết quả bảng range_part1 trong cơ sở dữ liệu.

public.range_part2/dds_assgn1/postgres@PostgreSQL 17																																																																															
No limit																																																																															
Query Query History																																																																															
<pre> 1 SELECT * FROM public.range_part2 2 </pre>																																																																															
Data Output Messages Notifications																																																																															
<table> <thead> <tr> <th></th><th>userid integer</th><th>movieid integer</th><th>rating double precision</th></tr> </thead> <tbody> <tr><td>2726789</td><td>71564</td><td>5872</td><td>3</td></tr> <tr><td>2726790</td><td>71564</td><td>6333</td><td>3</td></tr> <tr><td>2726791</td><td>71564</td><td>6537</td><td>3</td></tr> <tr><td>2726792</td><td>71564</td><td>7257</td><td>2.5</td></tr> <tr><td>2726793</td><td>71564</td><td>8528</td><td>2.5</td></tr> <tr><td>2726794</td><td>71564</td><td>8596</td><td>3</td></tr> <tr><td>2726795</td><td>71564</td><td>8636</td><td>3</td></tr> <tr><td>2726796</td><td>71565</td><td>6</td><td>3</td></tr> <tr><td>2726797</td><td>71565</td><td>21</td><td>3</td></tr> <tr><td>2726798</td><td>71565</td><td>527</td><td>3</td></tr> <tr><td>2726799</td><td>71565</td><td>903</td><td>3</td></tr> <tr><td>2726800</td><td>71565</td><td>1027</td><td>3</td></tr> <tr><td>2726801</td><td>71565</td><td>1077</td><td>3</td></tr> <tr><td>2726802</td><td>71565</td><td>1267</td><td>3</td></tr> <tr><td>2726803</td><td>71565</td><td>1280</td><td>3</td></tr> <tr><td>2726804</td><td>71565</td><td>1380</td><td>3</td></tr> <tr><td>2726805</td><td>71565</td><td>1547</td><td>3</td></tr> <tr><td>2726806</td><td>71565</td><td>1959</td><td>3</td></tr> </tbody> </table>					userid integer	movieid integer	rating double precision	2726789	71564	5872	3	2726790	71564	6333	3	2726791	71564	6537	3	2726792	71564	7257	2.5	2726793	71564	8528	2.5	2726794	71564	8596	3	2726795	71564	8636	3	2726796	71565	6	3	2726797	71565	21	3	2726798	71565	527	3	2726799	71565	903	3	2726800	71565	1027	3	2726801	71565	1077	3	2726802	71565	1267	3	2726803	71565	1280	3	2726804	71565	1380	3	2726805	71565	1547	3	2726806	71565	1959	3
	userid integer	movieid integer	rating double precision																																																																												
2726789	71564	5872	3																																																																												
2726790	71564	6333	3																																																																												
2726791	71564	6537	3																																																																												
2726792	71564	7257	2.5																																																																												
2726793	71564	8528	2.5																																																																												
2726794	71564	8596	3																																																																												
2726795	71564	8636	3																																																																												
2726796	71565	6	3																																																																												
2726797	71565	21	3																																																																												
2726798	71565	527	3																																																																												
2726799	71565	903	3																																																																												
2726800	71565	1027	3																																																																												
2726801	71565	1077	3																																																																												
2726802	71565	1267	3																																																																												
2726803	71565	1280	3																																																																												
2726804	71565	1380	3																																																																												
2726805	71565	1547	3																																																																												
2726806	71565	1959	3																																																																												
Total rows: 2726854		Query complete 00:00:01.200																																																																													

Hình 4.28. Kết quả bảng range_part2 trong cơ sở dữ liệu.

public.range_part3/ddsgn1/postgres@PostgreSQL 17

Query Query History

```
1 SELECT * FROM public.range_part3
2
```

Data Output Messages Notifications

	userid integer	movieid integer	rating double precision
3755493	71564	6218	3.5
3755494	71564	6539	4
3755495	71564	6636	3.5
3755496	71564	6796	3.5
3755497	71564	7193	4
3755498	71564	7325	3.5
3755499	71565	110	4
3755500	71565	111	4
3755501	71565	246	4
3755502	71565	260	4
3755503	71565	265	4
3755504	71565	318	4
3755505	71565	480	4
3755506	71565	608	4
3755507	71565	661	4
3755508	71565	704	4
3755509	71565	899	4
3755510	71565	908	4

Total rows: 3755614 Query complete 00:00:01.917

Hình 4.29. Kết quả bảng range_part3 trong cơ sở dữ liệu.

public.range_part4/dds_assgn1/postgres@PostgreSQL 17																																																																															
No limit																																																																															
Query Query History																																																																															
<pre>1 SELECT * FROM public.range_part4 2</pre>																																																																															
Data Output Messages Notifications																																																																															
<table> <thead> <tr> <th></th><th>userid integer</th><th>movieid integer</th><th>rating double precision</th></tr> </thead> <tbody> <tr><td>2129781</td><td>71564</td><td>1394</td><td>4.5</td></tr> <tr><td>2129782</td><td>71564</td><td>2858</td><td>4.5</td></tr> <tr><td>2129783</td><td>71564</td><td>2997</td><td>4.5</td></tr> <tr><td>2129784</td><td>71564</td><td>6807</td><td>5</td></tr> <tr><td>2129785</td><td>71564</td><td>8622</td><td>4.5</td></tr> <tr><td>2129786</td><td>71565</td><td>34</td><td>5</td></tr> <tr><td>2129787</td><td>71565</td><td>73</td><td>5</td></tr> <tr><td>2129788</td><td>71565</td><td>541</td><td>5</td></tr> <tr><td>2129789</td><td>71565</td><td>589</td><td>5</td></tr> <tr><td>2129790</td><td>71565</td><td>593</td><td>5</td></tr> <tr><td>2129791</td><td>71565</td><td>750</td><td>5</td></tr> <tr><td>2129792</td><td>71565</td><td>923</td><td>5</td></tr> <tr><td>2129793</td><td>71565</td><td>924</td><td>5</td></tr> <tr><td>2129794</td><td>71565</td><td>1080</td><td>5</td></tr> <tr><td>2129795</td><td>71565</td><td>1104</td><td>5</td></tr> <tr><td>2129796</td><td>71565</td><td>1178</td><td>5</td></tr> <tr><td>2129797</td><td>71565</td><td>1214</td><td>5</td></tr> <tr><td>2129798</td><td>71565</td><td>1221</td><td>5</td></tr> </tbody> </table>					userid integer	movieid integer	rating double precision	2129781	71564	1394	4.5	2129782	71564	2858	4.5	2129783	71564	2997	4.5	2129784	71564	6807	5	2129785	71564	8622	4.5	2129786	71565	34	5	2129787	71565	73	5	2129788	71565	541	5	2129789	71565	589	5	2129790	71565	593	5	2129791	71565	750	5	2129792	71565	923	5	2129793	71565	924	5	2129794	71565	1080	5	2129795	71565	1104	5	2129796	71565	1178	5	2129797	71565	1214	5	2129798	71565	1221	5
	userid integer	movieid integer	rating double precision																																																																												
2129781	71564	1394	4.5																																																																												
2129782	71564	2858	4.5																																																																												
2129783	71564	2997	4.5																																																																												
2129784	71564	6807	5																																																																												
2129785	71564	8622	4.5																																																																												
2129786	71565	34	5																																																																												
2129787	71565	73	5																																																																												
2129788	71565	541	5																																																																												
2129789	71565	589	5																																																																												
2129790	71565	593	5																																																																												
2129791	71565	750	5																																																																												
2129792	71565	923	5																																																																												
2129793	71565	924	5																																																																												
2129794	71565	1080	5																																																																												
2129795	71565	1104	5																																																																												
2129796	71565	1178	5																																																																												
2129797	71565	1214	5																																																																												
2129798	71565	1221	5																																																																												
Total rows: 2129834		Query complete 00:00:01.008																																																																													

Hình 4.30. Kết quả bảng range_part4 trong cơ sở dữ liệu.

4.2.3.3 Kết quả hàm rangeinsert()

Chạy file Assignment1Test, dữ liệu được thêm vào sẽ là 1 bản ghi với thông tin:

- userid: 100
- movieid: 2
- rating: 3

Vậy sẽ chỉ có bảng ratings và range_part2 bị ảnh hưởng.

Kết quả log ra là function đã pass và thời gian để insert dữ liệu là <1 giây.

```
Chọn thuật toán phân mảnh để test (range / roundrobin): range

Testing RANGE partitioning...
rangepartition function pass!

Tổng thời gian rangepartition: 24.388 giây
Nhập Enter để tiếp tục sau khi đã kiểm tra rangepartition...
rangeinsert function pass!

Tổng thời gian rangeinsert: 0.334 giây
Press enter to Delete all tables?
```

Hình 4.31. Log thể hiện function rangepartitions đã pass.

Query Query History

```

1 SELECT * FROM public.range_part2
2 WHERE userid = 100 AND movieid = 2 AND rating = 3
3

```

Data Output Messages Notifications

	userid integer	movieid integer	rating double precision
1	100	2	3

Hình 4.32. Kết quả bảng range_part2 trong cơ sở dữ liệu.

Query Query History

```

1 SELECT * FROM public.ratings
2 WHERE userid = 100 AND movieid = 2 AND rating = 3 |

```

Data Output Messages Notifications

	userid integer	movieid integer	rating double precision
1	100	2	3

Hình 4.33. Kết quả bảng ratings trong cơ sở dữ liệu.

4.2.3.4 Kết quả hàm roundrobinpartition()

Chạy file Assignment1 Test:

- Kết quả log ra là function đã pass.
- Tổng thời gian để chạy xong hàm roundrobinpartition hay phân mảnh vòng tròn là 47 giây.

```
C:\Users\truong\AppData\Local\Programs\Python\Python312\python.exe "F:\code\python learn\Range
A database named "dds_assgn1" already exists
Cơ sở dữ liệu 'dds_assgn1' đã tồn tại. Không cần tạo lại.
loadratings function pass!

Tổng thời gian load xong: 20.354 giây

Chọn thuật toán phân mảnh để test (range / roundrobin): roundrobin

Testing ROUND ROBIN partitioning...
roundrobinpartition function pass!

Tổng thời gian roundrobinpartition: 47.661 giây
Nhập Enter để tiếp tục sau khi đã kiểm tra roundrobinpartition...
```

Hình 4.34. Log thể hiện function roundrobinpartition đã pass.

public.rrobin_part0/dds_assgn1/postgres@PostgreSQL 17																																																																															
No limit																																																																															
Query Query History																																																																															
<pre>1 SELECT * FROM public.rrobin_part0 2</pre>																																																																															
Data Output Messages Notifications																																																																															
<table border="1"> <thead> <tr> <th></th><th>userid integer</th><th>movieid integer</th><th>rating double precision</th></tr> </thead> <tbody> <tr><td>1</td><td>1</td><td>122</td><td>5</td></tr> <tr><td>2</td><td>1</td><td>329</td><td>5</td></tr> <tr><td>3</td><td>1</td><td>370</td><td>5</td></tr> <tr><td>4</td><td>1</td><td>520</td><td>5</td></tr> <tr><td>5</td><td>1</td><td>594</td><td>5</td></tr> <tr><td>6</td><td>2</td><td>376</td><td>3</td></tr> <tr><td>7</td><td>2</td><td>733</td><td>3</td></tr> <tr><td>8</td><td>2</td><td>858</td><td>2</td></tr> <tr><td>9</td><td>2</td><td>1391</td><td>3</td></tr> <tr><td>10</td><td>3</td><td>590</td><td>3.5</td></tr> <tr><td>11</td><td>3</td><td>1288</td><td>3</td></tr> <tr><td>12</td><td>3</td><td>1674</td><td>4.5</td></tr> <tr><td>13</td><td>3</td><td>4995</td><td>4.5</td></tr> <tr><td>14</td><td>3</td><td>6287</td><td>3</td></tr> <tr><td>15</td><td>3</td><td>8529</td><td>4</td></tr> <tr><td>16</td><td>4</td><td>21</td><td>3</td></tr> <tr><td>17</td><td>4</td><td>153</td><td>5</td></tr> <tr><td>18</td><td>4</td><td>253</td><td>3</td></tr> </tbody> </table>					userid integer	movieid integer	rating double precision	1	1	122	5	2	1	329	5	3	1	370	5	4	1	520	5	5	1	594	5	6	2	376	3	7	2	733	3	8	2	858	2	9	2	1391	3	10	3	590	3.5	11	3	1288	3	12	3	1674	4.5	13	3	4995	4.5	14	3	6287	3	15	3	8529	4	16	4	21	3	17	4	153	5	18	4	253	3
	userid integer	movieid integer	rating double precision																																																																												
1	1	122	5																																																																												
2	1	329	5																																																																												
3	1	370	5																																																																												
4	1	520	5																																																																												
5	1	594	5																																																																												
6	2	376	3																																																																												
7	2	733	3																																																																												
8	2	858	2																																																																												
9	2	1391	3																																																																												
10	3	590	3.5																																																																												
11	3	1288	3																																																																												
12	3	1674	4.5																																																																												
13	3	4995	4.5																																																																												
14	3	6287	3																																																																												
15	3	8529	4																																																																												
16	4	21	3																																																																												
17	4	153	5																																																																												
18	4	253	3																																																																												
Total rows: 200011		Query complete 00:00:00.915																																																																													

Hình 4.35. Kết quả bảng rrobin_part0 trong cơ sở dữ liệu.

public.rrobin_part1/dds_assgn1/postgres@PostgreSQL 17																																																																															
No limit																																																																															
Query Query History																																																																															
<pre> 1 SELECT * FROM public.rrobin_part1 2 </pre>																																																																															
Data Output Messages Notifications																																																																															
<table border="1"> <thead> <tr> <th></th><th>userid integer</th><th>movieid integer</th><th>rating double precision</th></tr> </thead> <tbody> <tr><td>1</td><td>1</td><td>185</td><td>5</td></tr> <tr><td>2</td><td>1</td><td>355</td><td>5</td></tr> <tr><td>3</td><td>1</td><td>377</td><td>5</td></tr> <tr><td>4</td><td>1</td><td>539</td><td>5</td></tr> <tr><td>5</td><td>1</td><td>616</td><td>5</td></tr> <tr><td>6</td><td>2</td><td>539</td><td>3</td></tr> <tr><td>7</td><td>2</td><td>736</td><td>3</td></tr> <tr><td>8</td><td>2</td><td>1049</td><td>3</td></tr> <tr><td>9</td><td>2</td><td>1544</td><td>3</td></tr> <tr><td>10</td><td>3</td><td>1148</td><td>4</td></tr> <tr><td>11</td><td>3</td><td>1408</td><td>3.5</td></tr> <tr><td>12</td><td>3</td><td>3408</td><td>4</td></tr> <tr><td>13</td><td>3</td><td>5299</td><td>3</td></tr> <tr><td>14</td><td>3</td><td>6377</td><td>4</td></tr> <tr><td>15</td><td>3</td><td>8533</td><td>4.5</td></tr> <tr><td>16</td><td>4</td><td>34</td><td>5</td></tr> <tr><td>17</td><td>4</td><td>161</td><td>5</td></tr> <tr><td>18</td><td>4</td><td>266</td><td>5</td></tr> </tbody> </table>					userid integer	movieid integer	rating double precision	1	1	185	5	2	1	355	5	3	1	377	5	4	1	539	5	5	1	616	5	6	2	539	3	7	2	736	3	8	2	1049	3	9	2	1544	3	10	3	1148	4	11	3	1408	3.5	12	3	3408	4	13	3	5299	3	14	3	6377	4	15	3	8533	4.5	16	4	34	5	17	4	161	5	18	4	266	5
	userid integer	movieid integer	rating double precision																																																																												
1	1	185	5																																																																												
2	1	355	5																																																																												
3	1	377	5																																																																												
4	1	539	5																																																																												
5	1	616	5																																																																												
6	2	539	3																																																																												
7	2	736	3																																																																												
8	2	1049	3																																																																												
9	2	1544	3																																																																												
10	3	1148	4																																																																												
11	3	1408	3.5																																																																												
12	3	3408	4																																																																												
13	3	5299	3																																																																												
14	3	6377	4																																																																												
15	3	8533	4.5																																																																												
16	4	34	5																																																																												
17	4	161	5																																																																												
18	4	266	5																																																																												
Total rows: 2000011		Query complete 00:00:00.894																																																																													

Hình 4.36. Kết quả bảng rrobin_part1 trong cơ sở dữ liệu.

public.rrobin_part2/dds_assgn1/postgres@PostgreSQL 17			
No limit			
Query Query History			
1 SELECT * FROM public.rrobin_part2			
2			
Data Output Messages Notifications			
	userid integer	movieid integer	rating double precision
1	1	231	5
2	1	356	5
3	1	420	5
4	1	586	5
5	2	110	5
6	2	590	5
7	2	780	3
8	2	1073	3
9	3	110	4.5
10	3	1246	4
11	3	1552	2
12	3	3684	4.5
13	3	5505	2
14	3	6539	5
15	3	8783	5
16	4	39	3
17	4	165	5
18	4	292	3
Total rows: 200011		Query complete 00:00:01.071	

Hình 4.37. Kết quả bảng rrobin_part2 trong cơ sở dữ liệu.

public.rrobin_part3/dds_assgn1/postgres@PostgreSQL 17																																																																															
No limit																																																																															
Query Query History																																																																															
<pre>1 SELECT * FROM public.rrobin_part3 2</pre>																																																																															
Data Output Messages Notifications																																																																															
<table border="1"> <thead> <tr> <th></th><th>userid integer</th><th>movieid integer</th><th>rating double precision</th></tr> </thead> <tbody> <tr><td>1</td><td>1</td><td>292</td><td>5</td></tr> <tr><td>2</td><td>1</td><td>362</td><td>5</td></tr> <tr><td>3</td><td>1</td><td>466</td><td>5</td></tr> <tr><td>4</td><td>1</td><td>588</td><td>5</td></tr> <tr><td>5</td><td>2</td><td>151</td><td>3</td></tr> <tr><td>6</td><td>2</td><td>648</td><td>2</td></tr> <tr><td>7</td><td>2</td><td>786</td><td>3</td></tr> <tr><td>8</td><td>2</td><td>1210</td><td>4</td></tr> <tr><td>9</td><td>3</td><td>151</td><td>4.5</td></tr> <tr><td>10</td><td>3</td><td>1252</td><td>4</td></tr> <tr><td>11</td><td>3</td><td>1564</td><td>4.5</td></tr> <tr><td>12</td><td>3</td><td>4535</td><td>4</td></tr> <tr><td>13</td><td>3</td><td>5527</td><td>4.5</td></tr> <tr><td>14</td><td>3</td><td>7153</td><td>4</td></tr> <tr><td>15</td><td>3</td><td>27821</td><td>4.5</td></tr> <tr><td>16</td><td>4</td><td>110</td><td>5</td></tr> <tr><td>17</td><td>4</td><td>208</td><td>3</td></tr> <tr><td>18</td><td>4</td><td>316</td><td>5</td></tr> </tbody> </table>					userid integer	movieid integer	rating double precision	1	1	292	5	2	1	362	5	3	1	466	5	4	1	588	5	5	2	151	3	6	2	648	2	7	2	786	3	8	2	1210	4	9	3	151	4.5	10	3	1252	4	11	3	1564	4.5	12	3	4535	4	13	3	5527	4.5	14	3	7153	4	15	3	27821	4.5	16	4	110	5	17	4	208	3	18	4	316	5
	userid integer	movieid integer	rating double precision																																																																												
1	1	292	5																																																																												
2	1	362	5																																																																												
3	1	466	5																																																																												
4	1	588	5																																																																												
5	2	151	3																																																																												
6	2	648	2																																																																												
7	2	786	3																																																																												
8	2	1210	4																																																																												
9	3	151	4.5																																																																												
10	3	1252	4																																																																												
11	3	1564	4.5																																																																												
12	3	4535	4																																																																												
13	3	5527	4.5																																																																												
14	3	7153	4																																																																												
15	3	27821	4.5																																																																												
16	4	110	5																																																																												
17	4	208	3																																																																												
18	4	316	5																																																																												
Total rows: 2000011		Query complete 00:00:00.911																																																																													

Hình 4.38. Kết quả bảng rrobin_part3 trong cơ sở dữ liệu.

public.rrobin_part4/dds_assgn1/postgres@PostgreSQL 17			
No limit			
Query Query History			
1 SELECT * FROM public.rrobin_part4			
2			
Data Output Messages Notifications			
	userid integer	movieid integer	rating double precision
1	1	316	5
2	1	364	5
3	1	480	5
4	1	589	5
5	2	260	5
6	2	719	3
7	2	802	2
8	2	1356	3
9	3	213	5
10	3	1276	3.5
11	3	1597	4.5
12	3	4677	4
13	3	5952	3.5
14	3	7155	3.5
15	3	33750	3.5
16	4	150	5
17	4	231	1
18	4	317	5
Total rows: 2000010		Query complete 00:00:00.916	

Hình 4.39. Kết quả bảng rrobin_part4 trong cơ sở dữ liệu.

The screenshot shows a PostgreSQL query editor interface. At the top, the connection string is 'public.meta_rrobin_part_info/dds_assgn1/postgres@Postgres...'. Below the connection bar, there are icons for file operations, a filter icon, and a dropdown menu set to 'No limit'. The 'Query' tab is active, showing a SQL query: `SELECT * FROM public.meta_rrobin_part_info`. Below the query, the 'Data Output' tab is active, displaying the results of the query in a table format. The table has two columns: 'partition_number' (integer) and 'no_of_partitions' (integer). The first row of data shows '1' for partition_number and '5' for no_of_partitions.

	partition_number integer	no_of_partitions integer
1	3	5

Hình 4.40. Kết quả bảng metadata hay meta_rrobin_part_info trong cơ sở dữ liệu.

4.2.3.5 Kết quả hàm roundrobininsert()

Chạy file Assignment1 Test, dữ liệu được thêm vào sẽ là 1 bản ghi với thông tin:

- userid: 100
- movieid: 1
- rating: 3

Vậy sẽ chỉ có bảng ratings, metadata (meta_rrobin_part_info) và rrobin_part0 bị ảnh hưởng.

Kết quả log ra là function đã pass và thời gian để insert dữ liệu là <1 giây.

```
Tổng thời gian roundrobinpartition: 47.002 giây
Nhập Enter để tiếp tục sau khi đã kiểm tra roundrobinpartition...
roundrobininsert function pass!

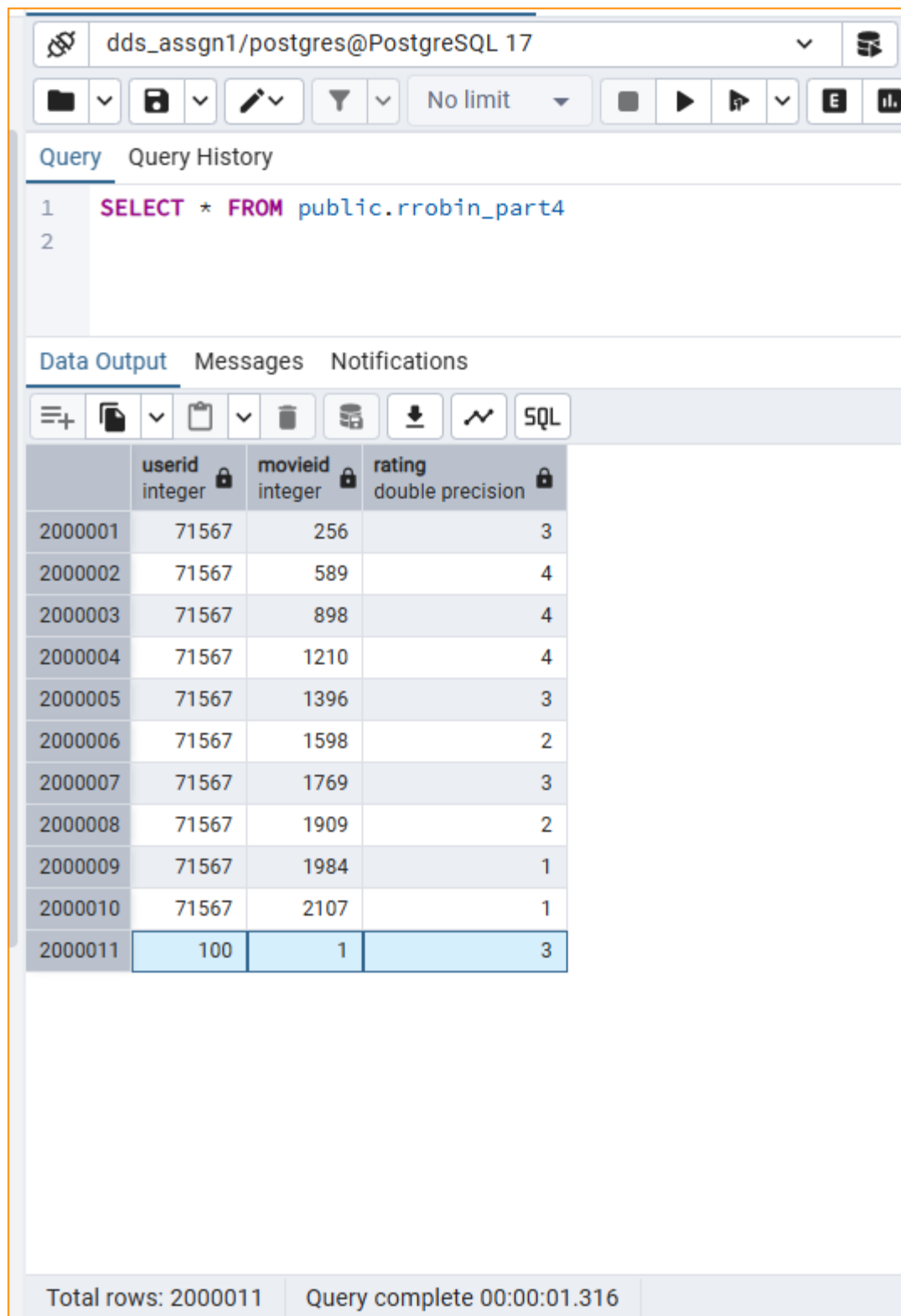
Tổng thời gian roundrobininsert: 0.127 giây
Press enter to Delete all tables?
```

Hình 4.41. Log thể hiện function roundrobinpartition đã pass.

The screenshot shows a PostgreSQL query editor interface. At the top, the connection is identified as 'dds_assgn1/postgres@PostgreSQL 17'. Below the connection bar is a toolbar with icons for file operations, query execution, and settings. The 'Query' tab is active, displaying the SQL query: `SELECT * FROM public.meta_rrobin_part_info`. Below the query editor, the 'Data Output' tab is selected, showing the results of the query in a table format. The table has two columns: 'partition_number' (integer) and 'no_of_partitions' (integer). The results show a single row with the values 1 and 5 respectively.

	partition_number integer	no_of_partitions integer
1	4	5

Hình 4.42. Kết quả bảng metadata hay meta_rrobin_part_info trong cơ sở dữ liệu.



Query: `SELECT * FROM public.rrobin_part4`

Data Output Messages Notifications

	userid integer	movieid integer	rating double precision
2000001	71567	256	3
2000002	71567	589	4
2000003	71567	898	4
2000004	71567	1210	4
2000005	71567	1396	3
2000006	71567	1598	2
2000007	71567	1769	3
2000008	71567	1909	2
2000009	71567	1984	1
2000010	71567	2107	1
2000011	100	1	3

Total rows: 2000011 Query complete 00:00:01.316

Hình 4.43. Kết quả bảng rrobin_part4 trong cơ sở dữ liệu.

V. KẾT LUẬN

Qua quá trình tìm hiểu và thực hiện bài tập lớn này, nhóm đã có cơ hội tiếp cận và áp dụng các phương pháp phân mảnh dữ liệu ngang trên cơ sở dữ liệu quan hệ bằng thuật toán Range Partition và Round Robin Partition. Việc triển khai các hàm `rangepartition`, `rangeinsert`, `roundrobinpartition`, và `roundrobininsert` giúp nhóm hiểu rõ hơn về cách thức phân mảnh dữ liệu để tối ưu hóa truy vấn cũng như quản lý dữ liệu lớn trong môi trường phân tán.

Bài tập lớn này không chỉ giúp nhóm củng cố kiến thức về cơ sở dữ liệu phân tán, mà còn rèn luyện các kỹ năng lập trình Python kết hợp với PostgreSQL, kỹ năng làm việc nhóm và quản lý dự án. Qua đó, nhóm đã học được cách thiết kế các hàm tối ưu, tránh thực thi lặp lại không cần thiết và sử dụng các kỹ thuật như CTE và bảng tạm để nâng cao hiệu suất.