

HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG

**BÁO CÁO BÀI TẬP LỚN**

**CƠ SỞ DỮ LIỆU PHÂN TÁN**

Giảng viên: Kim Ngọc Bách

|  |  |
| --- | --- |
| Sinh viên : | Tạ Quang Chiến - B22DCCN109  Nguyễn Mạnh Cường - B22DCCN100  Nguyễn Mậu Phi Hùng - B22DCCN366 |
| Nhóm lớp : | 9 |

Mục lục

[I) Yêu cầu bài tập 2](#_Toc199777867)

[1) Đề bài 2](#_Toc199777868)

[2) Giải pháp 2](#_Toc199777869)

[II) Thực hiện yêu cầu 2](#_Toc199777870)

[1) Tải tệp rating.dat 2](#_Toc199777871)

[2) Kết nối và tạo database 4](#_Toc199777872)

[3) Cài đặt hàm LoadRatings() 6](#_Toc199777873)

[4) Cài đặt hàm Range\_Partition() 13](#_Toc199777874)

[4.1) Thu thập thông tin test 13](#_Toc199777875)

[4.2) Cài đặt hàm 16](#_Toc199777876)

[4.3) Kiểm thử 19](#_Toc199777877)

[5) Cài đặt hàm Range\_ Insert() 20](#_Toc199777878)

[5.1) Thu thập thông tin test 20](#_Toc199777879)

[5.2) Cài đặt hàm 21](#_Toc199777880)

[5.3) Kiểm thử 23](#_Toc199777881)

[6) Cài đặt hàm RoundRobin\_Partition() 25](#_Toc199777882)

[6.1) Thu thập thông tin test 25](#_Toc199777883)

[6.2) Cài đặt hàm 27](#_Toc199777884)

[6.3) Kiểm thử 30](#_Toc199777885)

[7) Cài đặt hàm RoundRobin\_Insert() 31](#_Toc199777886)

[7.1) Thu thập thông tin test 31](#_Toc199777887)

[7.2) Cài đặt hàm 32](#_Toc199777888)

[7.3) Kiểm thử 34](#_Toc199777889)

[III) Tổng kết 35](#_Toc199777890)

# **I) Yêu cầu bài tập**

## **1) Đề bài**

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

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

## **2) Giải pháp**

a) Hệ quản trị cơ sở dữ liệu sử dụng

- Chọn PostgreSQL cho bài toán

b) Phương pháp

- Các bước cài đặt hàm sẽ được phân tích đánh giá qua các quá trình:

+ Đọc và hiểu cách thức mà code test hoạt động từ đó đưa ra giải pháp và ghi nhớ những điều cần chú ý

+ Cài đặt hàm phù hợp với yêu cầu ở bước đầu

+ Kết hợp hàm kiểm tra test và kiểm tra thủ công trên cơ sở dữ liệu để đảm báo tính đúng đắn của hàm cài đặt

* Các hàm sắp cài đặt dưới đây hầu hết sẽ trải qua các bước đã nêu trên

# **II) Thực hiện yêu cầu**

## **1) Tải tệp rating.dat**

- Xem xét file Assignment1Tester.py,

- File sau khi tải về có dạng như sauA dark blue rectangular object with white lines

AI-generated content may be incorrect.

A dark blue rectangular object with white text

AI-generated content may be incorrect.

Số bản ghi lên tới con số 10000054 (10 triệu +54 bản)

## **2) Kết nối và tạo database**

- Nhìn vào cấu trúc thư mục ta thấy được cấu trúc thư mục như sau  
A screenshot of a computer program

AI-generated content may be incorrect.

Thư mục chia làm 3 file python, trong đó file **Assignment1Tester.py** là file thực thi chính, file **testHelper.py** chứa code dùng để test cơ sở dữ liệu với các hàm được viết sẵn

Xem xét code file **Assignment1Tester.py**, cần điều chỉnh lại **INPUT\_FILE\_PATH**, và **ACTUAL\_ROWS\_IN\_INPUT\_FILE=10000054** cho phù hợp với số bản ghi trong file ratings

A screenshot of a computer program

AI-generated content may be incorrect.

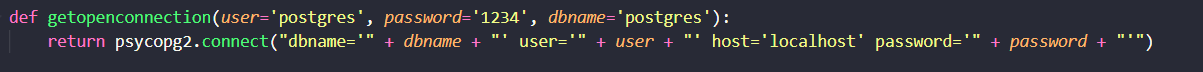
- Các tham số được khởi tạo mặc định, main gọi **testHelper.createdb(DATABASE\_NAME)**

- Đi vào hàm **createdb** của **testHelper**:

A screenshot of a computer code

AI-generated content may be incorrect.

* File test đã viết sẵn hàm kết nối với database, tạo database nếu chưa có



Hàm **getopenconnection**() kết nối mặc định đến database postgres với user postgres và password=”1234”

A computer screen shot of a program code

AI-generated content may be incorrect.

- Sau khi kết nối với database thì thực hiện xóa các bảng public

A computer code on a dark background

AI-generated content may be incorrect.

## **3) Cài đặt hàm LoadRatings()**

File **ratings.dat** sau khi được tải về có khoảng 10000054 (10 triệu +54 bản ghi)

A screenshot of a computer

AI-generated content may be incorrect.

- Từ file **Assignment1Tester.py,** ta thấy được để test hàm **LoadRatings**() cần thông qua câu lệnh **testHelper.testloadratings()**

A screen shot of a computer code

AI-generated content may be incorrect.

- Hàm **testloadratings**() nhận vào tham số là file bài tập(***MyAssignment***), tên bảng(***ratingstablename***), đường dẫn file dat(***filepath***), kết nối(***openconnection***) và số lượng bản ghi trong file dat(***rowsininpfile***)   
- Hàm test sẽ chạy hàm **LoadRatings**() sau đó kiểm tra số lượng bản ghi trong bảng có bằng với số lượng bản ghi trong file dat(***rowsininpfile****)* hay không, nếu không thì trả về kết quả false cùng với thông báo lỗi

=> Cần 1 hàm **LoadRatings**() phù hợp cho bài toán, ta có thể đề xuất 3 giải pháp

a) Chèn từng dòng, đọc từng dòng trong file ratings.dat rồi insert lần lượt vào trong bảng cơ sở dữ liệu

A computer screen shot of a program code

AI-generated content may be incorrect.

- Đầu tiên phải tạo bảng trong database nếu chưa có

- Tiếp theo, trong đoạn code trên, từng dòng được xử lý rồi thực hiện insert vào cơ sở dữ liệu, cứ 1000 bản ghi thì sẽ commit 1 lần

=> Dễ gây lỗi khi chèn, có thể lỗi giữa chừng, đặc biệt là chạy cực lâu khi thực hiện từng lệnh sql

=> Không sử dụng

b) Lưu vào file tạm rồi dùng copy\_from của psycopg2

A computer screen shot of a program code

AI-generated content may be incorrect.

- Đầu tiên phải tạo bảng trong database nếu chưa có

- Tiếp theo, trong đoạn code trên, từng dòng được xử lý (loại bỏ bớt 1 kí tự “:”) rồi ghi vào file tạm

=> Nhanh nhưng vẫn phải xử lý từng dòng trong file



=> Có thể sử dụng

c) Lợi dụng **copy\_from** của **psycopg2** để tạo bảng với các cột không liên quan rồi xóa cột

- Trong hàm **copy\_from** của **psycopg2** nhận đầu vào chỉ là 1 ký tự, nhưng phần input đầu vào lại phân tách các trường bằng 2 dấu “:”

=> Giải pháp: tạo các cột phụ bên cạnh các cột chính, sau khi thêm tất cả các bản ghi vào cơ sở dữ liệu thì drop những cột thừa ra đó/

Ví dụ: đầu vào là 1::1::3::912580553

Khi dùng copy\_from với tham số sep=”:” sẽ được phân tách thành [1,’’,1,’’,3,’’, 912580553]

=> Tạo bảng với 7 cột với tên lần lượt là [userid,extra1,movieid,extra2,rating,extra3,timestamp] như hình dưới

A computer screen shot of a program

AI-generated content may be incorrect.

Đọc file rating và dùng hàm **copy\_from**

Sau khi đọc xong sẽ drop các cột thừa

A screen shot of a computer code

AI-generated content may be incorrect.

A computer screen shot of a program

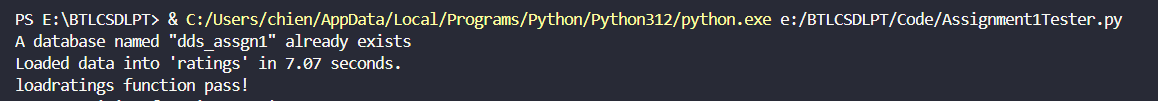
AI-generated content may be incorrect.

=> Nhanh



=> Sử dụng chiến lược này để load data vào cơ sở dữ liệu

- Sau khi chạy test đã thì đã có thể thấy thông báo thành công



Kiểm tra trong pgAdmin   
A screenshot of a computer

AI-generated content may be incorrect.

=> Đã có bảng ratings vơi đầy đủ các trường yêu cầu

Thực hiện truy vấn cũng ra kết quả tương tự  
A screenshot of a computer

AI-generated content may be incorrect.

=> Số lượng bản ghi hoàn toàn trùng khớp với file ban đầu là 10 triệu +54 bản ghi

## **4) Cài đặt hàm Range\_Partition()**

### **4.1) Thu thập thông tin test**

- Sơ lược về **Range partition** (phân vùng theo khoảng giá trị) là phương pháp chia một bảng lớn thành nhiều bảng (hoặc phân vùng) con dựa trên **giá trị của một cột** sao cho mỗi phân vùng chứa các bản ghi có giá trị nằm trong một **khoảng liên tiếp** (range) đã định trước. Trong bài tập này, bảng con được phân vùng dựa trên thuộc tính rating trải dài trên miền giá trị 0 đến 5

- Trong file **Assignment1Tester.py**, hàm **testHelper.testrangeinsert()** được gọi để test Range\_Partition.

A screen shot of a computer

AI-generated content may be incorrect.Đi vào file **testHelper.py** ta có hàm **testrangepartition**() nhận các tham số:

- *MyAssignment: file bài tập*

*- ratingstablename: bảng ratings*

*- n: số phân mảnh*

*- openconnection: kết nối*

*- partitionstartindex: giá trị đầu tiên của phần chia bảng*

*- ACTUAL\_ROWS\_IN\_INPUT\_FILE: số dòng trong file input*  
A computer screen shot of text

AI-generated content may be incorrect.

Hàm này gọi hàm rangepartition của file lời giải, sau đó test bằng 2 hàm **testrangeandrobinpartitioning**() và **testEachRangePartition**, ta cần đọc và kiểm tra những diều kiện cần để code đúng yêu cầu cho hàm **rangepartition**() trong file lời giải

a) Hàm **testrangeandrobinpartitioning**()

A screen shot of a computer program

AI-generated content may be incorrect.

Hàm này gọi 2 hàm con:  
- **checkpartitioncount**() : hàm này đếm số bảng có prefix được quy định, nếu số lượng bảng thực tế không khớp với số lượng bảng phân chia sẽ raise exceptionA computer code on a black background

AI-generated content may be incorrect.

- **totalrowsinallpartitions**(): kiểm tra và trả về tổng số lượng bản ghi trong từng bảng con sau khi phân chia (các bảng có prefix được quy định khi truyền vào hàm)

A computer code with colorful text

AI-generated content may be incorrect.

Trong hàm **testrangeandrobinpartitioning**() kiểm tra tổng số lượng bản ghi mà **totalrowsinallpartitions**() trả về có khớp với *ACTUAL\_ROWS\_IN\_INPUT\_FILE* hay không, nếu không thì sẽ raise exception tùy thuộc vào nó lớn hơn hay nhỏ hơn.

b) Hàm **testEachRangePartition**()

A computer code with colorful text

AI-generated content may be incorrect.

Hàm này gọi hàm con **getCountrangepartition**()

A computer screen shot of text

AI-generated content may be incorrect.

Khi xem xét ta thấy được điểm cần lưu ý và suy ra được từ phần test:  
+ Bảng con đầu tiên sẽ bắt đầu được chia theo điểm từ 0 đến nhỏ hơn hoặc bằng interval (hiệu lớn nhất giữa 2 diểm trong 1 bảng con). Biến interval sẽ là điểm xuất phát cho phần diểm nhỏ nhất của bảng con sau

+ Bắt đầu từ bảng con thứ 2, điểm sẽ giao động từ điểm lớn hơn điểm cao nhất của bảng con trước đến nhỏ hơn hoặc bằng điểm cao nhất của bảng con trước cộng với interval

* Cần lưu ý phần trên khi cài đặt hàm Range\_Patition()

Hàm **getCountrangepartition**() trả về 1 list lần lượt số bản ghi nên có trong các bảng con sau khi chia bảng, sau đó so sánh với số bản ghi có trong các bảng con (có prefix đúng yêu cầu) chỉ cần lệch 1 bảng thì sẽ raise exception

Kết luận: Hàm test chỉ đúng khi các hàm test không raise exception nào, lưu ý khi chia bảng sẽ được áp dụng để cài đặt hàm **Range\_Patition()**

### **4.2) Cài đặt hàm**

- Bắt đầu với các giá trị **RANGE\_TABLE\_PREFIX = 'range\_part'** theo yêu cầu đề bài

- **Range\_size** sẽ được tính bằng cách lấy 5 chia số số bảng muốn chia sẽ ra được khoảng cách lớn nhất giữa điểm lớn nhất và nhỏ nhất trong 1 bảng con

A screen shot of a computer code

AI-generated content may be incorrect.

- Đề bài cho phép sử dụng bảng meta-data, nên ta có tạo bảng tên là **partition\_metadata** để thuận tiện cho các phần sau này sẽ giải thích sau

A computer screen shot of a program

AI-generated content may be incorrect.

- Hàm **rangepartition()** gọi hàm tạo bảng **partition\_metadata** đồng thời chèn vào 1 bản ghi với thông tin như sau

* partition\_type: kiểu chia bảng hiện tại là ‘range’
* partition\_count: số bảng muốn chia
* range\_size: kích thước range

- Trong hàm sẽ không chèn giá trị cho hàm **current\_partition** vì nó được thiết kế để dành riêng cho phần round robin partition của phần sau

A computer screen with text

AI-generated content may be incorrect.

- Xử lý từng bảng con :

* Chạy vòng lặp từ bảng con đầu tiên bắt đầu là 0 đến (số bảng muốn chia -1)
* Tìm điểm thấp nhất và cao nhất có thể chèn vào trong bảng,
  + điểm thấp nhất là min\_rating= range\_size\* số thứ tự bảng con
  + điểm cao nhất là max\_rating = min\_rating + range\_size
* Đối với bảng con cuối cùng, điểm cao nhất có thể cho vào trong bảng phải là 5 vì phép chia và cộng số thực trong python có thể có sai số khiến cho phần max\_rating nhỏ hơn 5
* Tạo bảng con nếu chưa có: bằng Prefix theo yêu cầu đề bài + số thứ tự bảng con
* Tạo câu truy vấn insert các bản ghi thỏa mãn điều kiện từ bảng ratings (Bảng gốc) sang bảng con
* Riêng với bảng con đầu tiên, các bản ghi được lọc mỉn\_rating (ở bảng con đầu có giá trị bằng 0)<=rating<=max\_rating, các bảng con sau đó sẽ được lọc theo mỉn\_rating <rating<=max\_rating
* commit và kết thúc hàm

A computer screen shot of a program

AI-generated content may be incorrect.

- **Nhưng:** dễ thấy, bảng con và bảng cha có cấu trúc bảng giống nhau chỉ khác nhau tên bảng

=> có thể gộp bước tạo bảng con và insert thành 1 câu truy vấn, sử dụng kĩ thuật CTAS(Create Table As Select). Code sẽ trở thành như sau  
A computer screen shot of text

AI-generated content may be incorrect.

Toàn bộ hàm

A computer screen shot of a program

AI-generated content may be incorrect.

### **4.3) Kiểm thử**

- Từ bảng ratings gốc và số phân mảnh là 5 các bảng đã được phân chia như hình dưới

A screenshot of a computer

AI-generated content may be incorrect.

|  |  |
| --- | --- |
| A screenshot of a computer  AI-generated content may be incorrect. | A screenshot of a computer  AI-generated content may be incorrect. |
| A screenshot of a computer  AI-generated content may be incorrect. | A screenshot of a computer  AI-generated content may be incorrect. |
| A screenshot of a computer  AI-generated content may be incorrect. |  |

- Hoặc thử với số phân mảnh là 2  
A close-up of a computer screen

AI-generated content may be incorrect.

|  |  |
| --- | --- |
| A screenshot of a computer  AI-generated content may be incorrect. | A screenshot of a computer  AI-generated content may be incorrect. |

- Các truy vấn trên hình kết hợp với các test có trong phần hàm test có thể cho thấy được thuật toán phân mảnh trên là đúng đắn

## **5) Cài đặt hàm Range\_ Insert()**

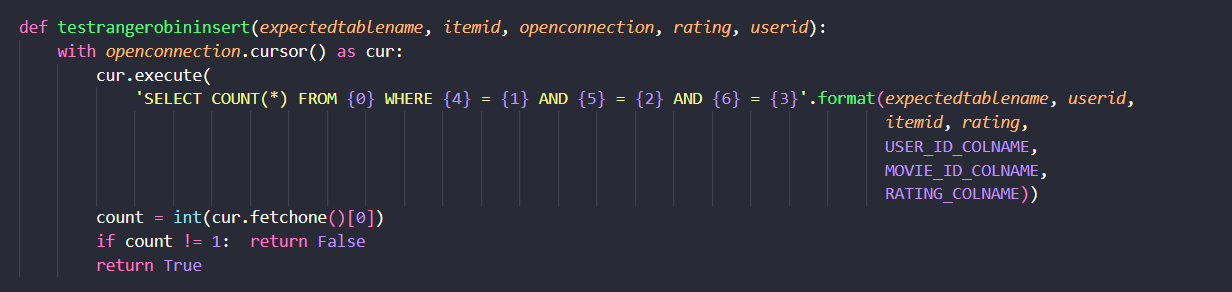
### **5.1) Thu thập thông tin test**

- Hàm **testrangeinsert()** là hàm kiểm tra cho phần cài đặt **Range\_ Insert()**

A screenshot of a computer program

AI-generated content may be incorrect.

- Trong hàm này **testrangeandrobinpartitioning()** được dùng để kiểm tra xem bản ghi mới được thêm vào có tồn tại trong bảng con hay không



- Nếu không tồn tại trong bảng con thì raise exception

=> Phải cài đặt hàm với đầu vào là 1 bản ghi mới và nó phải được điền vào đúng bảng con

### **5.2) Cài đặt hàm**

- Trong phần interface có sẵn có thể thấy được số bảng được phân chia được đếm thông   
A screen shot of a computer

AI-generated content may be incorrect.

=> Có thể thấy được số bảng được phân chia được đếm thông qua hàm **count\_partitions()**A screenshot of a computer program

AI-generated content may be incorrect.

Nhưng ở phần **rangepartition()** trước đó, chúng ta đã tạo 1 bảng partition\_metadata để lưu trữ thông tin của các phiên bản phân chia cơ sở dữ liệu, từ đó ta có thể dễ dàng lấy được số bảng cùng với đó là range\_size của nó mà không cần tính toán lại

Vì vậy hàm **rangeinsert()** của chúng ta sẽ như sau:

A computer screen shot of a program code

AI-generated content may be incorrect.

- Lấy **range\_size** từ metadata, tính toán bảng con mà bản ghi sẽ được chèn vào bằng cách lấy điểm cho bộ phim của bản ghi đó chia cho **range\_size** và chỉ lấy phần nguyên

- Lưu ý với phép tính chia hết, ta phải giảm 1 bậc số thứ tự của bảng con sau khi chia vì ở phần **rangepartition** trước đó đã để điều kiện **min\_rating< rating <=max\_rating** (trong đó rating là điểm của bản ghi mới định cho vào)

- Sau khi tính toán bảng mà bản ghi sẽ được thêm vào thì tất nhiên phải thêm bản ghi đó vào bảng gốc trước rồi mới thêm nó vào bảng con sau, cuối cùng là commit, kết thúc hàm

A computer screen shot of text

AI-generated content may be incorrect.

=> Toàn bộ hàm như sau  
A screen shot of a computer program

AI-generated content may be incorrect.

### **5.3) Kiểm thử**

- Với số lượng bảng phân chia là 5 vậy range\_size là 1:

A computer screen shot of a program

AI-generated content may be incorrect.

+ range\_part0: từ 0<=rating<=1

+ range\_part1: từ 1<rating<=2

+ range\_part2: từ 2<rating<=3

+ range\_part3: từ 3<rating<=4

+ range\_part4: từ 4<rating<=5

- Vậy với rating 2.5, 3, 0, 5 bảng mong muốn sẽ vào các bảng con có thứ tự 2, 2, 0, 4

+Rating=2.5 tìm trong bảng range\_part2 đã ra đúng kết quả

A screenshot of a computer

AI-generated content may be incorrect.

+Rating=3 tìm trong bảng range\_part2 đã ra đúng kết quả

A screenshot of a computer

AI-generated content may be incorrect.

+Rating=0 tìm trong bảng range\_part0 đã ra đúng kết quả

A screenshot of a computer

AI-generated content may be incorrect.

+Rating=5 tìm trong bảng range\_part4 đã ra đúng kết quả

A screenshot of a computer

AI-generated content may be incorrect.

Và điều chắc chắn rằng bảng con khác (các bảng từ 0 đến 3) sẽ không chứa bản ghi này  
A screenshot of a computer

AI-generated content may be incorrect.

Tương tự với các giá trị còn lại

## **6) Cài đặt hàm RoundRobin\_Partition()**

### **6.1) Thu thập thông tin test**

- Sơ lược về Round Robin Partition (phân vùng vòng tròn): là một kỹ thuật chia bảng lớn thành nhiều bảng con (partition) bằng cách phân phối các bản ghi theo thứ tự luân phiên, không dựa vào giá trị cụ thể của bất kỳ cột nào.

- Hàm **testroundrobinpartition()** có logic như sau

A screenshot of a computer program

AI-generated content may be incorrect.

- Phần **testrangeandrobinpartitioning()** đã được nói ở phần a) của phần 4.1), cần xem xét xử lý, kiểm tra phần test của hàm **testEachRoundrobinPartition()**

A screen shot of a computer code

AI-generated content may be incorrect.

- Hàm **testEachRoundrobinPartition()** gọi hàm **getCountroundrobinpartition()**

A screen shot of a computer program

AI-generated content may be incorrect.

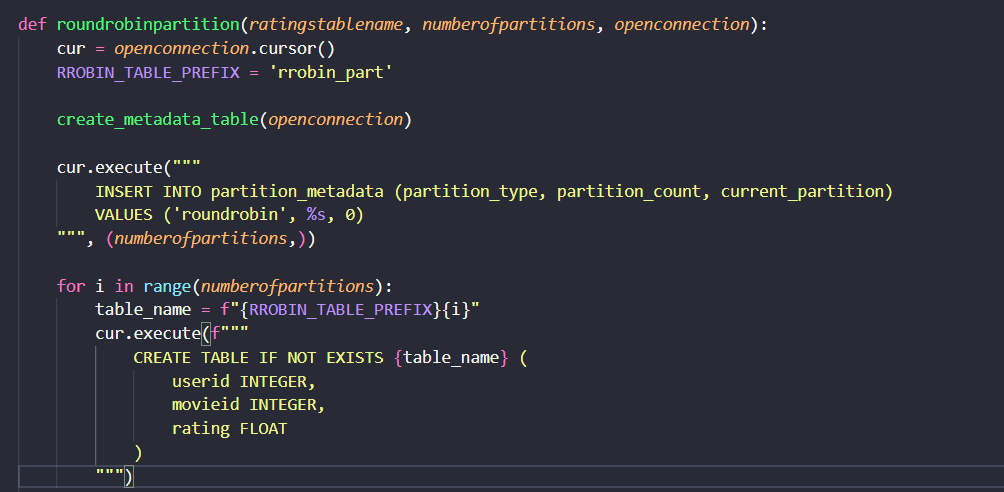
- Hàm trả về 1 danh sách số dòng mà các bảng con nên có, số thứ tự dòng phải trừ đi 1 vì số thứ tự của bảng con bắt đầu từ 0,nếu chỉ xét về mặt số lượng bản ghi thực chất chỉ là phân chia đồng đều số lượng bản ghi vào các bảng con sao cho đồng đều hết mức có thể (chênh lệch số bản ghi lớn nhất giữa các bảng là 1 hoặc 0).

- Sau khi có danh sách số lượng bản ghi mong muốn, hàm **testEachRoundrobinPartition()** so sánh với số lượng bản ghi thực tế có trong bảng con, nếu có sai khác thì raise exception

=> Cài đặt hàm chỉ đúng khi hàm test không raise exception

### **6.2) Cài đặt hàm**

- Định nghĩa RROBIN\_TABLE\_PREFIX = 'rrobin\_part'



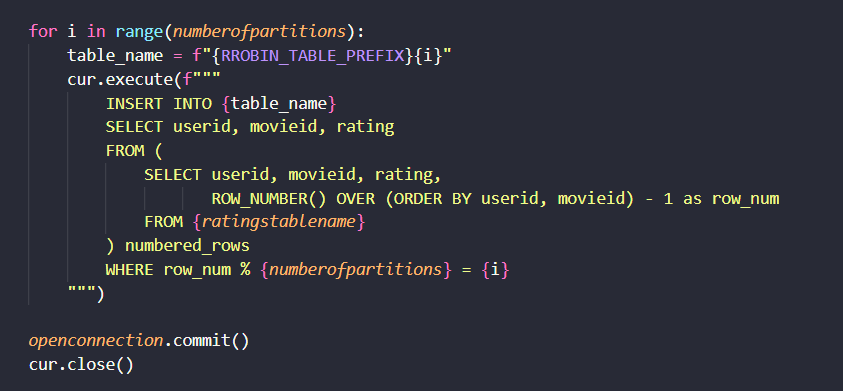
- Gọi hàm **create\_metadata\_table()** (đã nói ở phần rangepatition) để tạo bảng partition\_metadata

- Trước tiên phải thêm metadata cho phần chia bảng theo round robin:

* partition\_type: là ‘roundrobin’
* partition\_count: số lượng phân mảnh tùy thuộc vào đầu vào
* current\_partition: đặt là 0, sẽ được thay đổi sau này mỗi lần insert

Tương tự với kiểu **partition\_type** là ‘range’ không init trường **current\_partition**,với bản ghi có **partition\_type** là ‘roundrobin’ trường **range\_size** sẽ không được init vì đó là trường dành riêng cho kiểu phân chia bảng bằng ‘range’

- Tạo các bảng con có prefix là rrobin\_part + số thứ tự của phân mảnh bắt đầu từ 0



- Duyệt qua từng phân mảnh con, thực hiện truy vấn bảng gốc và đánh số từng dòng trong bản ghi  
- ROW\_NUMBER() OVER (ORDER BY userid, movieid) đảm bảo mỗi lần thực hiện đánh số dòng được đồng nhất vì đã được sắp xếp theo userid và movieid, nếu không sắp xếp (không ORDER) , mỗi bản ghi sẽ có số thứ tự dòng khác nhau trong mỗi lần truy xuất (vì truy vấn con

SELECT userid, movieid, rating,

ROW\_NUMBER() OVER () - 1 as row\_num

FROM {*ratingstablename*}

được gọi lại nhiều lần bằng với số bảng con )

- Chỉ chèn những bản ghi có row\_num chia lấy dư cho số bảng con bằng với số thứ tự bảng con(bắt đầu từ 0)

**Nhưng:** mỗi lần 1 bảng con lại phải sắp đánh số cho bảng gốc theo thứ tự sắp xếp, nếu có 5 phân mảnh thì phải sắp xếp + đánh số 5 lần, nếu phân mảnh sẽ làm bước này lặp lại nhiều lần vì việc sắp xếp và đánh số đảm bảo bảo việc 1 bản ghi không có 2 số thứ tự khác nhau trong các lần đánh số ở mỗi vòng lặp gây ra việc chèn thiếu.

=> Nên tạo bảng đã đánh số từ bảng gốc rồi chèn các bản ghi vào bảng con từ bảng đã đánh số đó (tránh được việc sắp xếp và đánh số nhiều lần) kết hợp với kỹ thuật CTAS. Như vậy code sẽ như sau

A computer screen shot of a program code

AI-generated content may be incorrect.

- Bảng **temp\_numbered** sẽ chứa toàn bộ bản ghi của bảng gốc cùng với số thứ tự hàng với mỗi bản ghi

- Lặp n lần (số phân mảnh cần tạo) tạo bảng và chèn vào theo đúng yêu cầu

- Thêm bản ghi vào bảng **partition\_metadata** trong 1 câu truy vấn thay vì tạo 1 bản ghi rồi cập nhật như code cũ

=> Giải quyết được vấn đề đặt ra đồng thời tối ưu việc chia bảng khi mà không phải sắp xếp đánh số nhiều lần trên bảng gốc

### **6.3) Kiểm thử**

|  |  |
| --- | --- |
| A screenshot of a computer  AI-generated content may be incorrect. | A screenshot of a computer  AI-generated content may be incorrect. |

- Thực hiện kiểm tra những bản ghi có trong bảng gốc nhưng không có trong những phân mảnh gộp lại và ngược lại đều cho ra kết quả rỗng (không có dòng hay bản ghi nào thỏa mãn)

A screenshot of a computer

AI-generated content may be incorrect.

- Các phân mảnh con cũng có đúng với bản ghi mà nó nên có

=> Phân mảnh có thể coi như thành công

## **7) Cài đặt hàm RoundRobin\_Insert()**

### **7.1) Thu thập thông tin test**

- Hàm **testroundrobininsert()** được sử dụng để test phần chèn bản ghi vào phân mảnh round robin

A screen shot of a computer program

AI-generated content may be incorrect.

- Hàm trên lại gọi testrangerobininsert() chỉ đơn giản là kiểm tra xem bản ghi mới chèn có nằm trong bảng mong muốn không, nếu không sẽ raise exception

A screen shot of a computer

AI-generated content may be incorrect.

- Yêu cầu test không có công thức chuẩn để nhận biết được rẳng nên chèn bản ghi này vào bảng nào tiếp theo, đầu vào ví dụ chỉ là

A screen shot of a computer

AI-generated content may be incorrect.

với số phân mảnh là 5 khi đã có sẵn 20 bản ghi trong bảng gốc

- Thì ra thuật toán chạy sẽ thêm lần lượt vào các bảng con theo số thứ tự bảng tăng dần, nhưng phải bắt đầu từ bảng sau bảng trước đó mới có bản ghi thêm vào. Nghĩa là nếu bản ghi trước đó được chèn vào phân mảnh 2 thì bản ghi tiếp theo sẽ phải được chèn vào phân mảnh 3

=> Cần có phương án để tìm phân mảnh tiếp theo, đề xuất 3 giải pháp:

+ Đếm số lượng bản ghi gốc trong bảng chính để tính phân mảnh tiếp theo được chèn vào

* Nguy cơ khi mà xóa 1 bản ghi trong bảng gốc, khi có bản ghi mới, hệ thống tính tổng bản ghi ở bảng gốc rồi chia modul cho số bảng thì bảng con được chọn để chèn vào lại trùng với bảng con được chọn của bản ghi trước đó => không đúng logic.
* Nguy cơ khi việc chèn là liên tục, xảy ra cùng 1 thời điểm, 2 lần chèn xảy ra cùng 1 thời điểm sẽ đọc bảng gốc cùng 1 thời điểm cho ra cùng 1 kết quả số lượng bản ghi trong bảng gốc, khi đó, 2 bản ghi này sẽ cùng 1 lúc được chèn vào cùng 1 bảng con => không hợp lệ

+Tính số lượng bản ghi trong từng bảng con, chọn bảng con có số lượng bản ghi thấp nhất và có số thứ tự bảng thấp nhất để chèn: Nguy cơ tương tự với phương án đầu tiên khi xóa 1 bản ghi ở 1 bảng con nào đó hoặc đồng thời đọc số lượng bản ghi ở các bảng con ở cùng 1 thời điểm khi có 2 yêu cầu chèn cùng lúc

+ Dùng metadata: như đã trình bày ở các bước cài đặt hàm trước đó, bảng partition\_metadata lưu trữ bảng ghi số lượng phân mảnh, riêng với kiểu round robin đã có 1 trường current\_partition dùng để lưu trữ bảng con sẽ được chèn tiếp theo nếu có bản ghi mới

=> Sử dụng phương án 3 để giải quyết cho phần cài đặt hàm roundrobininsert

### **7.2) Cài đặt hàm**

- Hàm thiết lập những tham số đầu vào RROBIN\_TABLE\_PREFIX = 'rrobin\_part' theo yêu cầu

- Tất nhiên cần phải thêm bản ghi vào trong bảng gốc trước

A screen shot of a computer program

AI-generated content may be incorrect.

- Như đã trình bày ở bước phân tích, việc chèn bản ghi theo chiến lược round robin cần sử dụng bảng partition\_metadata để lấy thông tin index của bảng sẽ được chọn để chèn thêm bảng như mới

A computer code with colorful text

AI-generated content may be incorrect.

Để đảm bảo không có lần gọi hàm chèn cùng 1 lúc, ta sẽ không select rồi update mà thay vào đó sẽ Update và Returning, điều này đảm bảo mỗi lần gọi hàm chèn sẽ lấy được current\_partition khác nhau

- current\_partition chính là index của bảng được chọn để chèn bản ghi hiện tại vào, thực hiện insert vào bảng phân mảnh tương ứng qua prefix+index

A computer screen shot of text

AI-generated content may be incorrect.

- Toàn bộ hàm   
A computer screen shot of a program code

AI-generated content may be incorrect.

### **7.3) Kiểm thử**

- Với phần phân mảnh con của phần trước (phân mảnh thành 5 phần) đã có kết quả như sau

A screenshot of a computer

AI-generated content may be incorrect.

- Dễ dàng thầy nếu có bản ghi mới, nó sẽ được chèn vào part4, sau khi chèn vào part4 thì quay trở về chèn bắt đầu từ part0 -> part1 ->part2 ->part3 ...

A screen shot of a computer program

AI-generated content may be incorrect.

Thử chèn 3 bản ghi, vậy thứ tự chèn phải tuân thủ chèn từ part4 rồi part0 rồi part1

A screenshot of a computer

AI-generated content may be incorrect.

Vậy thuật toán đã tuân thủ đúng quy tắc,chèn vào bảng con số 4 rồi bắt đầu chèn đến bảng 0 và 1, phần kiểm tra bản ghi có tồn tại trong bảng con không thì hàm test đã thực hiện điều đó

# **III) Tổng kết**

Thực hiện bài test tổng

- Vì số lượng bản ghi thực sự trong file ratings khác với file ví dụ nên cần điều chỉnh 1 chút ở file Assignment1Tester.py

A screenshot of a computer program

AI-generated content may be incorrect.

- Phần test đầu vào sẽ được thay đổi theo vùng màu đỏ, kết quả cho ra được phần giống hình ảnh bên dưới

A computer screen shot

AI-generated content may be incorrect.

- Nhìn chung phần cài đặt thuật toán đã thành công.

Trong đồ án này, nhóm em đã triển khai hai phương pháp phân mảnh ngang trên PostgreSQL: Range Partition và Round Robin Partition, cùng với hai hàm chèn động tương ứng (Range\_Insert và RoundRobin\_Insert). Đầu tiên, dữ liệu MovieLens (10 triệu+ 54 đánh giá) được nạp vào bảng gốc Ratings qua hàm LoadRatings().

Với Range\_Partition(N), thực hiện chia khoảng giá trị rating thành N vùng đồng đều, tạo N bảng con range\_part0 … range\_partN-1. Mỗi bản ghi được phân vào vùng phụ thuộc giá trị rating. Hàm Range\_Insert() sau đó đảm bảo chèn bản ghi mới vào đúng phân vùng bằng cách tính partition\_num = floor((rating – MIN) / range\_size).

Với RoundRobin\_Partition(N), dùng ROW\_NUMBER() OVER () để đánh số các bản ghi hiện có và phân luồng vào N bảng con rrobin\_part0 … rrobin\_partN-1 bằng công thức (rn - 1) % N. Đối với dữ liệu mới, RoundRobin\_Insert() dùng câu UPDATE … RETURNING để lấy chỉ số partition hiện tại và tăng mod N, sau đó chèn đánh giá vào bảng con tương ứng. Thử nghiệm với bộ dữ liệu mẫu và tập test do giảng viên cung cấp đều cho kết quả đúng, các testcase tự động pass 100%.

Trong quá trình thực hiện, nhóm nhận thấy metadata (current\_partition) trong Round Robin trở thành điểm “cổ chai” khi có nhiều client chèn song song, dù đã dùng UPDATE … RETURNING. Đối với Range Partition, khi dữ liệu phân phối không đồng đều (ví dụ rating tập trung ở 4–5 sao), một số partition chứa quá nhiều dữ liệu (skew), do đó cần cân nhắc nâng cấp thành Hash Partition hoặc giải pháp tự cân bằng.

Tóm lại, nhóm đã hoàn thành đầy đủ yêu cầu:

1. Load dữ liệu từ ratings.dat.
2. Tạo Range/ Round Robin Partition với N phân vùng.
3. Cài đặt hàm chèn động.
4. Kiểm thử toàn bộ test cases do giảng viên cung cấp.

- Phân chia công việc:

* Nguyễn Mậu Phi Hùng: làm Load Rating
* Nguyễn Mạnh Cường: làm Range Partition và Range Insert
* Tạ Quang Chiến: làm Round Robin Partition và Round Robin Insert