

**HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG**

-----□□□□-----



# **BÁO CÁO BÀI TẬP LỚN**

**Môn học: Cơ sở dữ liệu phân tán**

**Giảng viên hướng dẫn: Kim Ngọc Bách**

<b>Sinh viên thực hiện</b>	<b>Mã sinh viên</b>
----------------------------	---------------------

<b>Nguyễn Văn Đạt</b>	<b>B22DCCN199</b>
-----------------------	-------------------

<b>Bùi Xuân Trường</b>	<b>B22DCCN878</b>
------------------------	-------------------

<b>Phan Hữu Dương</b>	<b>B22DCCN171</b>
-----------------------	-------------------

## Hà Nội – 2025

### 1. Cài đặt Mysql

```
import mysql.connector

DATABASE_NAME = 'dds_assgn1'

def getopenconnection(user='root', password='123456', dbname='mysql'):
    return mysql.connector.connect(
        host='localhost',
        user='root',
        password='123456',
        database=dbname
    )
```

Hình 1: Đoạn mã Python thiết lập kết nối tới cơ sở dữ liệu MySQL

Hình 1 trình bày đoạn mã Python sử dụng thư viện **mysql.connector** để thiết lập kết nối đến cơ sở dữ liệu MySQL.

Cụ thể, hàm **getopenconnection()** được định nghĩa với các tham số như **user**, **password**, và **dbname**, cho phép tái sử dụng khi cần mở kết nối đến cơ sở dữ liệu. Trong ví dụ, kết nối được thiết lập với **localhost**, sử dụng tài khoản **root** và mật khẩu **123456**.

### 2. Cài đặt hàm Python LoadRatings()

Hàm **LoadRatings()** có nhiệm vụ nhận vào một đường dẫn tuyệt đối đến tệp **rating.data** nhưng do dữ liệu lớn nên trong chương trình này sẽ sử dụng dữ liệu mẫu là **test\_data.dat** và tải dữ liệu vào một bảng Mysql có tên **ratings**.

```

def loadratings(ratingtablename, ratingsfilepath, openconnection):

    con = openconnection
    cur = con.cursor()

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

    with open(ratingsfilepath, 'r') as f:
        for line in f:
            userid, _, movieid, _, rating, _, _ = line.strip().split(':')
            cur.execute("INSERT INTO " + ratingtablename +
                        " (userid, movieid, rating) VALUES (%s, %s, %s)",
                        (int(userid), int(movieid), float(rating)))

    cur.close()
    con.commit()

```

Hình 2: Hàm LoadRating()

Cụ thể:

- **con = openconnection** là đối tượng kết nối đến CSDL .
- **cur = con.cursor()** là cursor để thực thi các lệnh SQL.
- **DROP TABLE IF EXISTS:** xóa bảng cũ nếu đã tồn tại, tránh lỗi khi gọi lại hàm nhiều lần.
- **CREATE TABLE:** tạo bảng mới chỉ gồm ba cột cần thiết:
  - userid: kiểu interger.
  - movieid: kiểu interger.
  - rating: kiểu float.
- **with open (ratingsfilepath, 'r') as f:** đọc dữ liệu từ file ratingsfilepath.

Sau đó với mỗi dòng dữ liệu thì tách chuỗi theo dấu : và dùng **unpacking** để lấy ra các trường userid, movieid , rating

- Ví dụ: với dữ liệu dạng **1::122::5::838985046** thì nó **unpacking** thành **['1', '122', '5', '838985046']** và gán cho **userid = 1, movieid = 122 và rating = 5** bỏ qua timestamp.
- Tiếp theo thực thi câu lệnh chèn dữ liệu vào SQL sau đó commit lưu tất cả thay đổi vào database và đóng kết nối.

### 3. Cài đặt hàm Python Range\_Partition()

Hàm **rangepartition()** có nhiệm vụ phân chia dữ liệu từ bảng ratings chính thành nhiều bảng con (partitions) dựa trên khoảng giá trị rating từ 0 đến 5.

```
def rangepartition(ratingtablename, numberofpartitions, openconnection):

    con = openconnection
    cur = con.cursor()
    delta = 5.0 / numberofpartitions
    RANGE_TABLE_PREFIX = 'range_part'

    for i in range(numberofpartitions):
        cur.execute("DROP TABLE IF EXISTS " + RANGE_TABLE_PREFIX + str(i))

    for i in range(numberofpartitions):
        minRange = i * delta
        maxRange = minRange + delta
        table_name = RANGE_TABLE_PREFIX + str(i)

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

        if i == 0:
            cur.execute("INSERT INTO " + table_name +
                        " SELECT userid, movieid, rating FROM " + ratingtablename +
                        " WHERE rating >= %s AND rating <= %s", (minRange, maxRange))
        else:
            cur.execute("INSERT INTO " + table_name +
                        " SELECT userid, movieid, rating FROM " + ratingtablename +
                        " WHERE rating > %s AND rating <= %s", (minRange, maxRange))

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

Hình 3: Đoạn mã Python cài đặt hàm rangepartition()

## Cách hoạt động của hàm rangepartition

### a) Tính toán khoảng chia:

```
delta = 5.0 / numberofpartitions
```

- Chia khoảng rating [0-5] thành các phần bằng nhau
- Ví dụ: 5 partitions → delta = 1.0

### b) Xóa các bảng partition cũ:

```
for i in range(numberofpartitions):  
    cur.execute("DROP TABLE IF EXISTS " + RANGE_TABLE_PREFIX + str(i))
```

- Đảm bảo không có conflict với dữ liệu cũ
- Sử dụng IF EXISTS để tránh lỗi nếu bảng chưa tồn tại

### c) Tạo và phân chia dữ liệu:

```
for i in range(numberofpartitions):  
    minRange = i * delta  
    maxRange = minRange + delta  
    table_name = RANGE_TABLE_PREFIX + str(i)
```

- Lặp từ 0 đến (numberofpartitions - 1)
- Ví dụ: numberofpartitions = 5 → i = 0, 1, 2, 3, 4
- minRange: khoảng rating tối thiểu.
- maxRange: khoảng rating tối đa.

### d) Phân chia dữ liệu theo điều kiện:

Partition đầu tiên (i=0):

```
WHERE rating >= %s AND rating <= %s", (minRange, maxRange))
```

- Bao gồm cả giá trị biên dưới (>=)

Các partition còn lại:

```
WHERE rating > %s AND rating <= %s", (minRange, maxRange))
```

- Không bao gồm giá trị biên dưới (>) để tránh trùng lặp

### Ví dụ cụ thể:

Với numberofpartitions = 5:

Partition	Tên bảng	Khoảng rating	Điều kiện
0	range_part0	[0.0 - 1.0]	rating >= 0.0 AND rating <= 1.0
1	range_part1	(1.0 - 2.0]	rating > 1.0 AND rating <= 2.0

Partition	Tên bảng	Khoảng rating	Điều kiện
2	range_part2	(2.0 - 3.0]	rating > 2.0 AND rating <= 3.0
3	range_part3	(3.0 - 4.0]	rating > 3.0 AND rating <= 4.0
4	range_part4	(4.0 - 5.0]	rating > 4.0 AND rating <= 5.0

#### 4. Cài đặt hàm Python RoundRobin\_Partition()

Hàm **roundrobinpartition()** được thiết kế để phân chia dữ liệu từ một bảng gốc thành nhiều bảng con (partitions) sử dụng thuật toán Round Robin.

```
def roundrobinpartition(ratingtablename, numberofpartitions, openconnection):

    con = openconnection
    cur = con.cursor()
    RROBIN_TABLE_PREFIX = 'rrobin_part'

    for i in range(numberofpartitions):
        table_name = RROBIN_TABLE_PREFIX + str(i)
        cur.execute("DROP TABLE IF EXISTS " + table_name)
        cur.execute("CREATE TABLE " + table_name +
            " (userid INTEGER, movieid INTEGER, rating FLOAT)")
        sql_insert = (
            "INSERT INTO `" + table_name + "` (userid, movieid, rating) "
            "SELECT userid, movieid, rating "
            "FROM ("
            "    SELECT userid, movieid, rating, "
            "    ROW_NUMBER() OVER (ORDER BY userid ASC) AS rnum "
            "    FROM `" + ratingtablename + "` "
            ") AS numbered "
            "WHERE MOD(numbered.rnum - 1, " +
            str(numberofpartitions) + ") = " + str(i) + ";"
        )
        cur.execute(sql_insert)

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

### Cách chia dữ liệu theo Round-Robin

Với mỗi bảng con thì hàm thực hiện chèn dữ liệu vào bảng table\_name với ba cột userid, movieid, rating với dữ liệu được lấy ở truy vấn con bên dưới.

Cụ thể:

- Lấy toàn bộ dòng từ bảng **ratingtablename**.
- Sử dụng hàm của số ROW\_NUMBER() OVER (ORDER BY userid ASC) để **đánh số thứ tự từng dòng** theo thứ tự userid tăng dần.
- Gán số thứ tự này vào cột mới tên là rnum.
- Lọc với điều kiện: **(rnum - 1) MOD numberofpartitions == i**
- Những dòng nào thỏa mãn sẽ được thêm vào bảng.

### Ví dụ:

Dữ liệu gốc

userid	movieid	rating
1	122	5
1	185	4.5
1	231	4
1	292	3.5
1	329	3
1	355	2

**BƯỚC 1:** Đánh số thứ tự từng dòng với ROW\_NUMBER() OVER (ORDER BY userid ASC)

userid	movieid	rating	rnum
1	122	5	1
1	185	4.5	2
1	231	4	3
1	292	3.5	4
1	329	3	5
1	355	2	6

**BƯỚC 2:** Chia thành 5 phân mảnh theo:

- Với  $i = 0$  (partition 0)
  - $\text{MOD}(\text{rnum} - 1, 5) = 0$
  - $\text{rnum} = 1 \rightarrow (1-1)\%5 = 0$
  - $\text{rnum} = 6 \rightarrow (6-1)\%5 = 0$

- Với  $i = 1$  (partition 1)
  - $\text{MOD}(\text{rnum} - 1, 5) = 1$
  - $\text{rnum} = 2 \rightarrow (2-1)\%5 = 1$
- Với  $i = 2$  (partition 2)
  - $\text{MOD}(\text{rnum} - 1, 5) = 2$
  - $\text{rnum} = 1 \rightarrow (3-1)\%5 = 2$
- Với  $i = 3$  (partition 3)
  - $\text{MOD}(\text{rnum} - 1, 5) = 3$
  - $\text{rnum} = 1 \rightarrow (4-1)\%5 = 3$
- Với  $i = 4$  (partition 4)
  - $\text{MOD}(\text{rnum} - 1, 5) = 4$
  - $\text{rnum} = 1 \rightarrow (5-1)\%5 = 4$

## 5. Python RoundRobin\_Insert()

Hàm **roundrobininsert** thực hiện việc chèn dữ liệu rating vào cả bảng chính và các bảng phân mảnh theo thuật toán Round Robin.

```
def roundrobininsert(ratingtablename, userid, itemid, rating, openconnection):

    con = openconnection
    cur = con.cursor()
    RROBIN_TABLE_PREFIX = 'rrobin_part'

    cur.execute("INSERT INTO " + ratingtablename +
                " (userid, movieid, rating) VALUES (%s, %s, %s)",
                (userid, itemid, rating))

    cur.execute("SELECT COUNT(*) FROM " + ratingtablename)
    total_rows = cur.fetchone()[0]

    cur.execute("SELECT COUNT(*) FROM information_schema.tables WHERE table_schema = DATABASE() AND table_name LIKE %s",
                (RROBIN_TABLE_PREFIX + '%',))
    numberofpartitions = cur.fetchone()[0]

    index = (total_rows - 1) % numberofpartitions
    table_name = RROBIN_TABLE_PREFIX + str(index)

    cur.execute("INSERT INTO " + table_name +
                " (userid, movieid, rating) VALUES (%s, %s, %s)",
                (userid, itemid, rating))

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

### Cách hoạt động:

#### Bước 1: Khởi tạo kết nối và cursor

```
con = openconnection
cur = con.cursor()
RROBIN_TABLE_PREFIX = 'rrobin_part'
```

#### Chức năng:

- Thiết lập kết nối cơ sở dữ liệu
- Tạo cursor object để thực thi các câu lệnh SQL
- Định nghĩa hằng số tiền tố cho các bảng phân vùng



Kết quả:

- **con**: Object kết nối database
- **cur**: Cursor để thực thi SQL commands
- **RROBIN\_TABLE\_PREFIX**: Chuỗi "rrobin\_part" làm prefix

## Bước 2: Chèn dữ liệu vào bảng chính

```
cur.execute("INSERT INTO " + ratingstablename +  
            " (userid, movieid, rating) VALUES (%s, %s, %s)",  
            (userid, itemid, rating))
```

Chức năng:

- Thực hiện câu lệnh INSERT vào bảng chính
- Chèn một bản ghi mới với 3 trường: userid, movieid, rating

## Bước 3: Đếm tổng số bản ghi

```
cur.execute("SELECT COUNT(*) FROM " + ratingstablename)  
total_rows = cur.fetchone()[0]
```

Chức năng:

- Truy vấn số lượng bản ghi hiện tại trong bảng chính
- Lấy kết quả từ câu truy vấn COUNT
- **total\_rows**: Số nguyên biểu thị tổng số bản ghi (bao gồm bản ghi vừa chèn)

## Bước 4: Xác định số lượng bảng phân vùng

```
cur.execute("SELECT COUNT(*) FROM information_schema.tables WHERE table_schema = DATABASE() AND table_name LIKE %s",  
            (RROBIN_TABLE_PREFIX + '%',))  
numberofpartitions = cur.fetchone()[0]
```

Chức năng:

- Truy vấn metadata của database để đếm số bảng phân vùng
- Sử dụng **information\_schema.tables** để lấy thông tin về các bảng
- Tìm tất cả bảng có tên bắt đầu bằng 'rrobin\_part'
- **numberofpartitions**: Số lượng bảng phân vùng có sẵn (ví dụ: 3, 5, 10...)

## Bước 5: Tính toán chỉ số phân vùng đích

Chức năng:

- Sử dụng thuật toán Round Robin để xác định phân vùng đích
- Phép chia lấy dư đảm bảo phân phối đều
- Tạo tên bảng phân vùng cụ thể

**Ví dụ tính toán:**

- Nếu **total\_rows** = 7 và **numberofpartitions** = 3:
  - $\text{index} = (7 - 1) \% 3 = 6 \% 3 = 0$
  - **table\_name** = "rrobin\_part0"

## Bước 6: Chèn vào partition tương ứng

Mục đích:

- Chèn dữ liệu vào bảng phân mảnh đã tính toán
- Thực hiện phân tán dữ liệu

## 6. Python Range\_Insert()

Hàm **rangeinsert** thực hiện việc chèn dữ liệu đánh giá phim vào cơ sở dữ liệu theo phương pháp phân vùng Range Partitioning, phân phối dữ liệu dựa trên giá trị đánh giá (rating) vào các phân vùng tương ứng.

```
def rangeinsert(ratingtablename, userid, itemid, rating, openconnection):  
  
    con = openconnection  
    cur = con.cursor()  
    RANGE_TABLE_PREFIX = 'range_part'  
  
    cur.execute("SELECT COUNT(*) FROM information_schema.tables WHERE table_schema = DATABASE() AND table_name LIKE %s",  
                (RANGE_TABLE_PREFIX + '%',))  
    numberofpartitions = cur.fetchone()[0]  
  
    delta = 5.0 / numberofpartitions  
    index = int(rating / delta)  
    if rating % delta == 0 and index != 0:  
        index = index - 1  
  
    table_name = RANGE_TABLE_PREFIX + str(index)  
    cur.execute("INSERT INTO " + ratingtablename +  
                " (userid, movieid, rating) VALUES (%s, %s, %s)",  
                (userid, itemid, rating))  
    cur.execute("INSERT INTO " + table_name +  
                " (userid, movieid, rating) VALUES (%s, %s, %s)",  
                (userid, itemid, rating))  
  
    cur.close()  
    con.commit()
```

## Bước 1: Khởi tạo môi trường

```
con = openconnection  
cur = con.cursor()  
RANGE_TABLE_PREFIX = 'range_part'
```

Chức năng:

- Thiết lập kết nối cơ sở dữ liệu
- Tạo cursor object để thực thi các câu lệnh SQL
- Định nghĩa hằng số tiền tố cho các bảng phân vùng range

Kết quả:

- con: Object kết nối database
- cur: Cursor để thực thi SQL commands
- RANGE\_TABLE\_PREFIX: Chuỗi "range\_part" làm prefix

## Bước 2: Xác định số lượng bảng phân vùng

```
cur.execute("SELECT COUNT(*) FROM information_schema.tables WHERE table_schema = DATABASE() AND table_name LIKE %s",  
            (RANGE_TABLE_PREFIX + '%',))  
numberofpartitions = cur.fetchone()[0]
```

Chức năng:

- Truy vấn metadata của database để đếm số bảng phân vùng range
- Sử dụng **information\_schema.tables** để lấy thông tin về các bảng
- Tìm tất cả bảng có tên bắt đầu bằng 'range\_part'
- numberofpartitions: Số lượng bảng phân vùng có sẵn (ví dụ: 3, 5, 10...)

## Bước 3: Tính toán khoảng chia (delta)

```
delta = 5.0 / numberofpartitions
```

Chức năng:

- Tính khoảng chia đều thang điểm từ 0 đến 5
- Mỗi phân vùng sẽ chứa một khoảng giá trị rating nhất định

Ví dụ tính toán:

- Nếu numberofpartitions = 5:
  - $\text{delta} = 5.0 / 5 = 1.0$
  - Phân vùng 0: [0.0, 1.0]
  - Phân vùng 1: (1.0, 2.0]
  - Phân vùng 2: (2.0, 3.0]
  - Phân vùng 3: (3.0, 4.0]
  - Phân vùng 4: (4.0, 5.0]
- delta: Kích thước khoảng giá trị cho mỗi phân vùng

## Bước 4: Tính toán chỉ số phân vùng

```
index = int(rating / delta)  
if rating % delta == 0 and index != 0:  
    index = index - 1
```

Chức năng:

- Xác định phân vùng dựa trên giá trị rating
- Xử lý trường hợp đặc biệt khi rating chia hết cho delta

Logic chi tiết:

### a) Tính index ban đầu

- $\text{index} = \text{int}(\text{rating} / \text{delta})$ 
  - Chia rating cho delta và lấy phần nguyên
  - Ví dụ:  $\text{rating} = 3.7, \text{delta} = 1.0 \rightarrow \text{index} = \text{int}(3.7/1.0) = 3$

### b) Xử lý trường hợp biên

- if  $\text{rating} \% \text{delta} == 0$  and  $\text{index} != 0$ :  $\text{index} = \text{index} - 1$ 
  - Điều kiện:  $\text{rating} \% \text{delta} == 0$  (rating chia hết cho delta)
  - Và  $\text{index} != 0$  (không phải phân vùng đầu tiên)
  - Hành động: Giảm index đi 1

### Ví dụ xử lý biên:

- $\text{rating} = 2.0, \text{delta} = 1.0, \text{numberofpartitions} = 5$ :
  - $\text{index} = \text{int}(2.0/1.0) = 2$
  - $2.0 \% 1.0 = 0$  và  $\text{index} != 0 \rightarrow \text{index} = 2 - 1 = 1$
  - Kết quả: rating 2.0 thuộc phân vùng 1 (khoảng (1.0, 2.0])

## Bước 5: Tạo tên bảng phân vùng

```
table_name = RANGE_TABLE_PREFIX + str(index)
cur.execute("INSERT INTO " + ratingtablename +
            " (userid, movieid, rating) VALUES (%s, %s, %s)",
            (userid, itemid, rating))
cur.execute("INSERT INTO " + table_name +
            " (userid, movieid, rating) VALUES (%s, %s, %s)",
            (userid, itemid, rating))
```

### Chức năng:

- Tạo tên bảng phân vùng cụ thể dựa trên index
- Kết hợp prefix với chỉ số phân vùng
- Đưa dữ liệu vào đúng phân vùng
- Thực hiện câu lệnh INSERT dữ liệu vào bảng chính
- Lưu trữ dữ liệu gốc trong bảng tổng hợp

## 7. Kết quả






Với  $\text{numberofpartitions} = 5$






```
loadratings function pass!
rangepartition function pass!
rangeinsert function pass!
roundrobinpartition function pass!
roundrobininsert function pass!
Press enter to Delete all tables?
```

Dữ liệu được load:

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







Các phân mảnh được tạo ra:

 range_part0
 range_part1
 range_part2
 range_part3
 range_part4

 rrobin_part0
 rrobin_part1
 rrobin_part2
 rrobin_part3
 rrobin_part4

Phân vùng 0: [0.0, 1.0]	<table><tr><th>userid</th><th>movieid</th><th>rating</th></tr><tr><td>1</td><td>362</td><td>1</td></tr><tr><td>1</td><td>364</td><td>0.5</td></tr><tr><td>1</td><td>370</td><td>0</td></tr></table>	userid	movieid	rating	1	362	1	1	364	0.5	1	370	0	rrobin_part0	<table><tr><th>userid</th><th>movieid</th><th>rating</th></tr><tr><td>1</td><td>122</td><td>5</td></tr><tr><td>1</td><td>329</td><td>2.5</td></tr><tr><td>1</td><td>370</td><td>0</td></tr><tr><td>1</td><td>520</td><td>2.5</td></tr><tr><td>100</td><td>1</td><td>3</td></tr></table>	userid	movieid	rating	1	122	5	1	329	2.5	1	370	0	1	520	2.5	100	1	3						
	userid	movieid	rating																																				
	1	362	1																																				
	1	364	0.5																																				
	1	370	0																																				
userid	movieid	rating																																					
1	122	5																																					
1	329	2.5																																					
1	370	0																																					
1	520	2.5																																					
100	1	3																																					
Phân vùng 1: (1.0, 2.0]	<table><tr><th>userid</th><th>movieid</th><th>rating</th></tr><tr><td>1</td><td>355</td><td>2</td></tr><tr><td>1</td><td>356</td><td>1.5</td></tr><tr><td>1</td><td>589</td><td>1.5</td></tr></table>	userid	movieid	rating	1	355	2	1	356	1.5	1	589	1.5	rrobin_part1	<table><tr><th>userid</th><th>movieid</th><th>rating</th></tr><tr><td>1</td><td>185</td><td>4.5</td></tr><tr><td>1</td><td>355</td><td>2</td></tr><tr><td>1</td><td>377</td><td>3.5</td></tr><tr><td>1</td><td>539</td><td>5</td></tr></table>	userid	movieid	rating	1	185	4.5	1	355	2	1	377	3.5	1	539	5									
	userid	movieid	rating																																				
	1	355	2																																				
	1	356	1.5																																				
1	589	1.5																																					
userid	movieid	rating																																					
1	185	4.5																																					
1	355	2																																					
1	377	3.5																																					
1	539	5																																					
Phân vùng 2: (2.0, 3.0]	<table><tr><th>userid</th><th>movieid</th><th>rating</th></tr><tr><td>1</td><td>316</td><td>3</td></tr><tr><td>1</td><td>329</td><td>2.5</td></tr><tr><td>1</td><td>520</td><td>2.5</td></tr><tr><td>100</td><td>2</td><td>3</td></tr></table>	userid	movieid	rating	1	316	3	1	329	2.5	1	520	2.5	100	2	3	rrobin_part2	<table><tr><th>userid</th><th>movieid</th><th>rating</th></tr><tr><td>1</td><td>231</td><td>4</td></tr><tr><td>1</td><td>356</td><td>1.5</td></tr><tr><td>1</td><td>420</td><td>5</td></tr><tr><td>1</td><td>586</td><td>3.5</td></tr></table>	userid	movieid	rating	1	231	4	1	356	1.5	1	420	5	1	586	3.5						
	userid	movieid	rating																																				
	1	316	3																																				
	1	329	2.5																																				
	1	520	2.5																																				
100	2	3																																					
userid	movieid	rating																																					
1	231	4																																					
1	356	1.5																																					
1	420	5																																					
1	586	3.5																																					
Phân vùng 3: (3.0, 4.0]	<table><tr><th>userid</th><th>movieid</th><th>rating</th></tr><tr><td>1</td><td>231</td><td>4</td></tr><tr><td>1</td><td>292</td><td>3.5</td></tr><tr><td>1</td><td>377</td><td>3.5</td></tr><tr><td>1</td><td>466</td><td>4</td></tr><tr><td>1</td><td>586</td><td>3.5</td></tr></table>	userid	movieid	rating	1	231	4	1	292	3.5	1	377	3.5	1	466	4	1	586	3.5	rrobin_part3	<table><tr><th>userid</th><th>movieid</th><th>rating</th></tr><tr><td>1</td><td>292</td><td>3.5</td></tr><tr><td>1</td><td>362</td><td>1</td></tr><tr><td>1</td><td>466</td><td>4</td></tr><tr><td>1</td><td>588</td><td>5</td></tr></table>	userid	movieid	rating	1	292	3.5	1	362	1	1	466	4	1	588	5			
	userid	movieid	rating																																				
	1	231	4																																				
	1	292	3.5																																				
	1	377	3.5																																				
	1	466	4																																				
1	586	3.5																																					
userid	movieid	rating																																					
1	292	3.5																																					
1	362	1																																					
1	466	4																																					
1	588	5																																					
Phân vùng 4: (4.0, 5.0]	<table><tr><th>userid</th><th>movieid</th><th>rating</th></tr><tr><td>1</td><td>122</td><td>5</td></tr><tr><td>1</td><td>185</td><td>4.5</td></tr><tr><td>1</td><td>420</td><td>5</td></tr><tr><td>1</td><td>480</td><td>5</td></tr><tr><td>1</td><td>539</td><td>5</td></tr><tr><td>1</td><td>588</td><td>5</td></tr></table>	userid	movieid	rating	1	122	5	1	185	4.5	1	420	5	1	480	5	1	539	5	1	588	5	rrobin_part4	<table><tr><th>userid</th><th>movieid</th><th>rating</th></tr><tr><td>1</td><td>316</td><td>3</td></tr><tr><td>1</td><td>364</td><td>0.5</td></tr><tr><td>1</td><td>480</td><td>5</td></tr><tr><td>1</td><td>589</td><td>1.5</td></tr></table>	userid	movieid	rating	1	316	3	1	364	0.5	1	480	5	1	589	1.5
	userid	movieid	rating																																				
	1	122	5																																				
	1	185	4.5																																				
	1	420	5																																				
	1	480	5																																				
	1	539	5																																				
1	588	5																																					
userid	movieid	rating																																					
1	316	3																																					
1	364	0.5																																					
1	480	5																																					
1	589	1.5																																					

Với `numberofpartitions = 3`  
 Các phân mảnh được tạo ra:

 range_part0	 robin_part0
 range_part1	 robin_part1
 range_part2	 robin_part2

Phân vùng 0: [0, 1.67]				rrobin_part0			
	userid	movieid	rating		userid	movieid	rating
	1	356	1.5		1	122	5
	1	362	1		1	292	3.5
	1	364	0.5		1	355	2
	1	370	0		1	364	0.5
	1	589	1.5		1	420	5
				1	520	2.5	
				1	588	5	

Phân vùng 1: (1.67, 3.34]				rrobin_part1			
	userid	movieid	rating		userid	movieid	rating
	1	316	3		1	185	4.5
	1	329	2.5		1	316	3
	1	355	2		1	356	1.5
	1	520	2.5		1	370	0
	100	2	3		1	466	4
				1	539	5	
				1	589	1.5	

Phân vùng 2: (3.34, 5]	userid	movieid	rating	rrobin_part2	userid	movieid	rating
	1	122	5		1	231	4
	1	185	4.5		1	329	2.5
	1	231	4		1	362	1
	1	292	3.5		1	377	3.5
	1	377	3.5		1	480	5
	1	420	5		1	586	3.5
	1	466	4		100	1	3
	1	480	5				
	1	539	5				
	1	586	3.5				
	1	588	5				