

HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG

KHOA CÔNG NGHỆ THÔNG TIN I

BỘ MÔN CƠ SỞ DỮ LIỆU PHÂN TÁN



BÀI TẬP LỚN CƠ SỞ DỮ LIỆU PHÂN TÁN

Giảng viên hướng dẫn

Tên thành viên

Nguyễn Tùng Dương

Nguyễn Hải Sơn

Lớp

Nhóm

Kim Ngọc Bách

Mã sinh viên

B22DCCN168

B22DCCN694

D22 - 10

01

Hà Nội – 2024

I. YÊU CẦU

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

Dữ liệu đầu vào

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

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

`UserID::MovieID::Rating::Timestamp`

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

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

II. THỰC HIỆN

1. Kết nối đến Cơ sở dữ liệu

Để làm việc với cơ sở dữ liệu, đầu tiên cần cấu hình kết nối trong hai file `testHelper.py` và `Interface.py`. Cụ thể, trong hàm `getOpenConnection`, phải thay đổi thông tin người dùng và mật khẩu mặc định bằng thông tin đăng nhập PostgreSQL trên máy cá nhân.

Ví dụ:

```
def getopenconnection(user='postgres', password='1234', dbname='postgres'):
    return psycopg2.connect("dbname='" + dbname + "' user='" + user +
        "localhost" + password + "'")
```

2. Cài đặt hàm LoadRatings

Hàm `loadratings` có nhiệm vụ đọc dữ liệu từ một tệp văn bản chứa thông tin đánh giá phim và lưu trữ vào một bảng trong cơ sở dữ liệu PostgreSQL. Dữ liệu được lưu dưới định dạng `userid:movieid:rating:timestamp` – với dấu `:` là dấu phân cách giữa các trường.

Cú pháp hàm:

```
def loadratings(ratingstablename, ratingsfilepath, openconnection):
```

- ratingtablename: Tên bảng cần tạo để lưu dữ liệu đánh giá.
- ratingsfilepath: Đường dẫn tới tệp dữ liệu đánh giá (dạng text).
- openconnection: Kết nối mở tới cơ sở dữ liệu PostgreSQL.

Các bước thực hiện:

B1. Tạo bảng tạm để lưu dữ liệu thô:

```
cur.execute("create table " + ratingtablename + "(userid integer, extra1 char, movieid integer, extra2 char, rating float, extra3 char, timestamp bigint);")
```

Vì dữ liệu trong file sử dụng dấu : làm dấu phân cách, PostgreSQL cần định nghĩa đủ số cột tạm để lưu tất cả các thành phần, bao gồm các dấu : không cần thiết (extra1, extra2, extra3).

B2. Chèn dữ liệu từ file vào bảng:

```
cur.copy_from(open(ratingsfilepath), ratingtablename, sep=':')
```

B3. Xoá các cột thừa (extra1, extra2, extra3, timestamp):

```
cur.execute("alter table " + ratingtablename + " drop column extra1, drop column extra2, drop column extra3, drop column timestamp;")
```

B4. Lưu thay đổi và đóng kết nối:

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

Kết quả:

Sau khi thực thi hàm, bảng ratingtablename sẽ chứa dữ liệu dạng:

	userid integer	movieid integer	rating double precision
1	1	122	5
2	1	185	5
3	1	231	5
4	1	292	5
5	1	316	5
6	1	329	5

3. Cài đặt hàm Range_Partition

Tính khoảng chia delta đại diện cho độ rộng của rating trên mỗi phân vùng.

Tạo vòng lặp từ 0 đến số lượng phân vùng – 1, với mỗi vòng lặp:

- Tính khoảng rating bắt đầu bằng cách lấy delta nhân với biến đếm vòng lặp.

- Tính khoảng rating kết thúc bằng cách lấy khoảng bắt đầu cộng với delta.
- Tạo bảng mới đại diện cho phân vùng.
- Chèn dữ liệu từ bảng gốc vào phân vùng với điều kiện rating lớn hơn khoảng bắt đầu và nhỏ hơn hoặc bằng khoảng kết thúc (Với phân vùng đầu tiên sẽ dùng điều kiện rating lớn hơn hoặc bằng khoảng bắt đầu để không bỏ sót rating = 0).

4. Cài đặt hàm RoundRobin_Partition

Tạo vòng lặp từ 0 đến `numberofpartitions - 1`, với mỗi vòng lặp:

- Tạo bảng mới đại diện cho phân vùng.
- Chèn dữ liệu từ bảng gốc vào phân vùng theo nguyên tắc:
 - Đánh số thứ tự tất cả các dòng của bảng gốc bằng cách sử dụng hàm `ROW_NUMBER()`.
 - Với mỗi dòng, tính chỉ số phân vùng bằng công thức:
 $(row_number - 1) \bmod numberofpartitions$
 - Chỉ những dòng có kết quả bằng `i` mới được chèn vào phân vùng thứ `i`.

5. Cài đặt hàm Range_Insert

Chèn dòng dữ liệu mới vào bảng chính.

Lấy số lượng phân vùng hiện tại bằng cách gọi hàm `count_partitions(...)` với tiền tố `range_part`.

Tính độ rộng của mỗi phân vùng (delta): $5/numberofpartitions$.

Xác định chỉ số phân vùng (index) dựa trên giá trị rating:

Index = $\text{int}(\text{rating}/\text{delta})$

Xử lý biên trái: Nếu rating nằm đúng ở ranh giới chia và `index != 0`, thì giảm index đi 1 để đảm bảo tính đúng phạm vi.

Tạo tên bảng phân vùng theo dạng: `range_part{index}`.

Chèn dòng dữ liệu mới vào bảng phân vùng tương ứng.

6. Cài đặt hàm RoundRobin_Insert

Chèn dòng mới vào bảng chính.

Đếm tổng số dòng hiện có trong bảng chính (sau khi thêm dòng mới).

Lấy số lượng phân vùng hiện tại bằng cách gọi hàm `count_partitions(...)` với tiền tố `rrobin_part`.

Xác định chỉ số phân vùng (index) cần thêm dòng vào bằng công thức:

Index = $(\text{total_rows} - 1) \bmod numberofpartitions$

Tạo tên bảng phân vùng theo dạng `rrobin_part{index}`.

Chèn dòng dữ liệu mới vào bảng phân vùng tương ứng.

7. Bổ sung

Chỉnh sửa code cho phép nhập số lượng phân mảnh mong muốn cho các hàm tạo phân mảnh và cho phép nhập vào dữ liệu trước khi thực hiện các hàm insert.

PHÂN CHIA CÔNG VIỆC

STT	Công việc/Nhiệm vụ	Sinh viên thực hiện
1	Kết nối CSDL, hàm Range_Partition(), hàm Range_Insert()	Nguyễn Hải Sơn
2	hàm LoadRatings(), hàm RoundRobin_Partition(), hàm RoundRobin_Insert()	Nguyễn Tùng Dương