Санкт-Петербургский Политехнический Университет Петра Великого

Институт компьютерных наук и технологий

Кафедра компьютерных систем и программных технологий

ОТЧЕТ
по лабораторной работе

«Хранимые процедуры»

Базы данных

Работу выполнил студент

группа 43501/3 Дьячков В.В.

Работу принял преподаватель

_____ Мяснов А.В.

Санкт-Петербург

2018

Содержание

1. Цель работы

Познакомить студентов с возможностями реализации более сложной обработки данных на стороне сервера с помощью хранимых процедур.

2. Программа работы

1. Изучение возможностей языка PL/pgSQL.
2. Создание двух хранимых процедур в соответствии с индивидуальным заданием, полученным у преподавателя.
3. Выкладывание скрипта с созданными сущностями в репозиторий.
4. Демонстрация результатов преподавателю.

3. Хранимые процедуры

3.1. Рекомендации по изменению количества номеров

Задание: Для заданного отеля и количества лет реализовать расчет рекомендаций по увеличению/уменьшению количества номеров каждого типа. Решение принимать на основе превышения/снижения значения ниже порогового.

Выберем следующий критерий эффективности заданного типа номера: если среднее число бронирований такого типа номеров выше заданного порога, то такой тип номера считается эффективным и можно рекомендовать увеличить число таких номеров. Неэффективные номера находятся по аналогии. Будем формировать итоговую процедуру поэтапно, демонстрируя промежуточные результаты для отеля с ID = 5:

1. Определение количества бронирований номеров каждого типа для выбранного отеля:

```sql
SELECT room_type_id AS room_type_id,
       COUNT(*)     AS reservations_count
FROM reservation
    JOIN room ON reservation.room_id = room.id
    JOIN room_type ON room.room_type_id = room_type.id
WHERE date_part('Y', now()) - date_part('Y', reservation.from) <= 2
    AND hotel_id = 1
GROUP BY room_type_id
```

Листинг 1: `reservatrions_per_room_type.sql`

```
 room_type_id | reservations_count
--------------+--------------------
            1 |                103
            2 |                171
            3 |                230
            4 |                218
```

```
            5 |                250
            6 |                213
            7 |                126
            8 |                249
            9 |                166
(9 rows)
```

2. Определение количества номеров соответствующих каждому типу номеров:

```sql
SELECT room_type_id AS room_type_id, COUNT(*) AS room_count
FROM room
    JOIN room_type ON room.room_type_id = room_type.id
WHERE hotel_id = 5
GROUP BY room_type_id
ORDER BY room_type_id
```

Листинг 2: `rooms_per_room_type.sql`

```
 room_type_id | room_count
--------------+------------
            1 |          8
            2 |         12
            3 |         11
            4 |         12
            5 |          9
            6 |         22
            7 |         14
            8 |         12
            9 |         15
(9 rows)
```

3. Формирование итоговой рекомендации по увеличению или уменьшению количества номеров данного типа в заданном отеле:

```sql
CREATE OR REPLACE FUNCTION recommend_rooms_optimization(p_hotel_id        INTEGER,
                                                        p_years           INTEGER,
                                                        p_upper_threshold INTEGER,
                                                        p_lower_threshold INTEGER)
    RETURNS TABLE(room_type_id BIGINT, ratio BIGINT, increase BOOLEAN, decrease BOOLEAN)
LANGUAGE SQL
AS $$
WITH reservatrions_per_room_type AS (
    SELECT room_type_id AS room_type_id,
           COUNT(*)     AS reservations_count
    FROM reservation
        JOIN room ON reservation.room_id = room.id
        JOIN room_type ON room.room_type_id = room_type.id
    WHERE
        date_part('Y', now()) - date_part('Y', reservation.from) <= p_years
        AND hotel_id = p_hotel_id
    GROUP BY room_type_id
),
rooms_per_room_type AS (
    SELECT room_type_id AS room_type_id, COUNT(*) AS room_count
    FROM room
        JOIN room_type ON room.room_type_id = room_type.id
    WHERE hotel_id = p_hotel_id
    GROUP BY room_type_id
)
```

```
SELECT summary.room_type_id              AS room_type_id,
       summary.ratio                     AS ratio,
       summary.ratio > p_upper_threshold AS increase,
       summary.ratio < p_lower_threshold AS decrease
FROM (
    SELECT rooms_per_room_type.room_type_id AS room_type_id,
        reservations_count / room_count  AS ratio
    FROM reservatrions_per_room_type
        JOIN rooms_per_room_type
            ON rooms_per_room_type.room_type_id = reservatrions_per_room_type.room_type_id
        JOIN room_type ON room_type.id = rooms_per_room_type.room_type_id
    ORDER BY room_type_id
) AS summary
$$;

SELECT *
FROM recommend_rooms_optimization(5, 2, 22, 19);
```

Листинг 3: `recommend_rooms_optimization.sql`

```
CREATE FUNCTION
 room_type_id | ratio | increase | decrease
--------------+-------+----------+----------
            1 |    23 | t        | f
            2 |    16 | f        | t
            3 |    18 | f        | t
            4 |    19 | f        | f
            5 |    19 | f        | f
            6 |    20 | f        | f
            7 |    21 | f        | f
            8 |    20 | f        | f
            9 |    19 | f        | f
(9 rows)
```

В результирующем отчете выводится список типов номеров в отеле и соответствующие им рекомендации: необходимо ли увеличить количество номеров такого типа (`increase`) или уменьшить (`decrease`).

Измерение времени: увеличим количество бронирований до 100 тысяч с помощью генератора и измерим время исполнения итогового запроса:

1. execution time: 48 ms, fetching time: 22 ms
2. execution time: 39 ms, fetching time: 5 ms
3. execution time: 37 ms, fetching time: 9 ms

Видно, что после первого запуска время выполнения запроса немного уменьшилось, что объясняется кэшированием результатов внутри базы данных.

3.2. Рекомендации на следующую поездку

Задание: На основе известных данных о госте (страны, длительности, отзывы и пр.) сформировать рекомендации на следующую поездку.
Будем рекомендовать отели пользователю по следующему критерию:

- найдем бронирования пользователя, в отзыве к которым он указал высокую оценку;
- найдем пользователей, которым понравился тот же номер в отеле;
- найдем у этих пользователей другие бронирования, в отзывах к которым был также указан высокий рейтинг, и будем рекомендовать пользователю отели, в которых были оставлены эти отзывы.

Будем формировать итоговую процедуру поэтапно, демонстрируя промежуточные результаты для пользователя с ID = 3:

1. Определение номеров, забронировав которые пользователь указал высокую оценку:

```sql
SELECT room.id AS room_id
FROM reservation
    JOIN room ON reservation.room_id = room.id
    JOIN review ON review.reservation_id = reservation.id
WHERE reservation.user_id = 6
    AND review.rating >= 4
```

Листинг 4: `liked_rooms.sql`

```
 room_id
---------
    2968
     478
    3041
    1499
(4 rows)
```

2. Определение пользователей, которые также ставили высокую оценку этим номерам:

```sql
WITH liked_rooms AS (
    SELECT room.id AS room_id
    FROM reservation
        JOIN room ON reservation.room_id = room.id
        JOIN review ON review.reservation_id = reservation.id
    WHERE reservation.user_id = 6
        AND review.rating >= 4
)
SELECT user_id AS user_id
FROM reservation
    JOIN room ON reservation.room_id = room.id
    JOIN hotel ON hotel.id = room.hotel_id
    JOIN liked_rooms ON liked_rooms.room_id = room.id
    JOIN review ON reservation.id = review.reservation_id
WHERE review.rating >= 4
    AND user_id != 6
```

Листинг 5: `similar_users.sql`

```
 user_id
---------
    4257
    4124
    3364
    3502
     367
    1927
     451
     355
    1064
(9 rows)
```

3. Формирование итоговой рекомендации на следующую поездку:

```sql
CREATE OR REPLACE FUNCTION recommend_hotel(p_user_id INTEGER, p_rating_threshold INTEGER)
    RETURNS TABLE(hotel_id INTEGER)
LANGUAGE SQL
AS $$
WITH similiar_users AS (
    WITH liked_rooms AS (
        SELECT room.id AS room_id
        FROM reservation
            JOIN room ON reservation.room_id = room.id
            JOIN review ON review.reservation_id = reservation.id
        WHERE reservation.user_id = p_user_id
            AND review.rating >= p_rating_threshold
    )
    SELECT user_id AS user_id
    FROM reservation
        JOIN room ON reservation.room_id = room.id
        JOIN hotel ON hotel.id = room.hotel_id
        JOIN liked_rooms ON liked_rooms.room_id = room.id
        JOIN review ON reservation.id = review.reservation_id
    WHERE review.rating >= p_rating_threshold
        AND user_id != p_user_id
)
SELECT hotel_id AS hotel_id
FROM reservation
    JOIN similiar_users ON similiar_users.user_id = reservation.user_id
    JOIN room ON room.id = reservation.room_id
    JOIN review ON review.reservation_id = reservation.id
WHERE review.rating >= p_rating_threshold
    AND room.id NOT IN (
        SELECT room_id
        FROM reservation
        WHERE reservation.user_id = p_user_id
    )
ORDER BY review.rating DESC
LIMIT 5
$$;

SELECT hotel.name AS hotel_name, city.name AS city_name, country.name AS country_name
FROM hotel
     JOIN recommend_hotel(6, 4) AS recommendation ON recommendation.hotel_id = hotel.id
     JOIN city ON city.id = hotel.city_id
     JOIN country ON country.id = city.country_id;
```

Листинг 6: `recommend_hotel.sql`

```
CREATE FUNCTION
        hotel_name          |  city_name   |   country_name
----------------------------+--------------+-------------------
 Jenkins Inc                | Dortmund     | Republic of Korea
 Emard Inc                  | Ekaterinburg | Russia
 Kulas LLC                  | Paris        | France
 Wisoky, Wisoky and Wisoky | Kiev         | Ukraine
 Homenick-Homenick          | Miami        | USA
(5 rows)
```

Измерение времени: увеличим количество бронирований до 100 тысяч с помощью генератора и измерим время исполнения итогового запроса:

1. execution time: 91 ms, fetching time: 8 ms
2. execution time: 68 ms, fetching time: 9 ms
3. execution time: 60 ms, fetching time: 9 ms

Видно, что после первого запуска время исполнения также несильно уменьшилось.

4. Выводы

В процессе выполнения данной работы:

- изучены возможности языка `PL/pgSQL`;
- реализованы хранимые процедуры по заданию преподавателя: рекомендации по изменению количества номеров и рекомендации на следующую поездку.