

Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное
учреждение высшего образования
«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»
Факультет инфокоммуникационных технологий

ОТЧЕТ
О ЛАБОРАТОРНОЙ РАБОТЕ № 5
по теме: процедуры, функции, триггеры в PostgreSQL

Специальность:
09.03.03 Мобильные и сетевые технологии

Проверил:
Горова М.М. _____
Дата: «__» _____ 20__ г.
Оценка _____

Выполнил:
студент группы К3240
Козлов И.Д.

Санкт-Петербург 2022

ЦЕЛЬ РАБОТЫ

Овладеть практическими создания и использования процедур, функций и триггеров в базе данных PostgreSQL.

ПРАКТИЧЕСКОЕ ЗАДАНИЕ

Оборудование: компьютерный класс.

Программное обеспечение: СУБД PostgreSQL, SQL Shell (psql).

Практическое задание:

Вариант 1

1. Создать процедуры/функции согласно индивидуальному заданию и (согласно индивидуальному заданию, часть 4).
2. Создать триггер для логирования событий вставки, удаления, редактирования данных в базе данных PostgreSQL (согласно индивидуальному заданию, часть 5). Допустимо создать универсальный триггер или отдельные триггеры на логирование действий.

Вариант 2

Модифицировать триггер (триггерную функцию) на проверку корректности входа и выхода сотрудника (см. Практическое задание 1 Лабораторного практикума (Приложение)) с максимальным учетом «узких» мест некорректных данных по входу и выходу.

Указание. Работа выполняется в консоли SQL Shell (psql).

Вариант 1. БД «Отель»

Описание предметной области: Отели сети находятся в разных городах. Цены на номера одного типа во всех отелях одинаковы и зависят от типа номера и количества мест. Номер может быть забронирован, занят или свободен. При заезде в отель постояльцы проходят регистрацию. Информация о регистрации постояльцев отеля (выехавших из отеля) хранится в течение года и 1 января удаляется в архив.

БД должна содержать следующий минимальный набор сведений: Адрес отеля. Название отеля. Номер комнаты. Тип комнаты. Количество мест. Цена комнаты за сутки проживания. Имя постояльца. Фамилия постояльца. Отчество постояльца. Адрес постоянного проживания. Дата заезда. Дата отъезда.

Дополнить исходные данные информацией: по бронированию комнаты; по сотруднику, который регистрирует постояльца в отеле в день заезда; по оплате проживания; по составу удобств в комнате; по акциям, доступным при бронировании (скидки).

Задание 1. Создайте хранимые процедуры:

- для увеличения цены всех номеров на 5 %, если в отеле нет свободных номеров;
- для получения информации о свободных номерах отеля на завтрашний день;
- бронирования двухместного номера в гостинице на заданную дату и количество дней проживания.

Задание 2. Создайте необходимые триггеры.

ХОД РАБОТЫ

- 1) Для увеличения цены всех номеров на 5 %, если в отеле нет свободных номеров.

```
CREATE OR REPLACE PROCEDURE rise_price_on_five_percents()
LANGUAGE plpgsql AS
$$
BEGIN
UPDATE hotel.price SET price_for_one_day = CASE
WHEN (select COUNT(hotel.room.status_busyness) AS flags
from hotel.room, hotel.price where hotel.room.status_busyness = 'Свободно') = 0
THEN hotel.price.price_for_one_day * 1.05
ELSE hotel.price.price_for_one_day * 1
END;
END
$$;
```

```
1 CREATE OR REPLACE PROCEDURE rise_price_on_five_percents()
2 LANGUAGE plpgsql AS
3 $$
4 BEGIN
5 UPDATE hotel.price SET price_for_one_day = CASE
6 WHEN (select COUNT(hotel.room.status_busyness) AS flags
7 from hotel.room, hotel.price where hotel.room.status_busyness = 'Свободно') = 0
8 THEN hotel.price.price_for_one_day * 1.05
9 ELSE hotel.price.price_for_one_day * 1
10 END;
11 END
12 $$;
```

Создаю процедуру через консоль

```
[postgres=# \c hotel1
You are now connected to database "hotel1" as user "postgres".
hotel1=# CREATE OR REPLACE PROCEDURE rise_price_on_five_percents()
hotel1=# LANGUAGE plpgsql AS
hotel1=# $$
hotel1$# BEGIN
hotel1$# UPDATE hotel.price SET price_for_one_day = CASE
hotel1$# WHEN (select COUNT(hotel.room.status_busyness) AS flags
hotel1$# from hotel.room, hotel.price where hotel.room.status_busyness = 'Свободно') = 0
hotel1$# THEN hotel.price.price_for_one_day * 1.05
hotel1$# ELSE hotel.price.price_for_one_day * 1
hotel1$# END;
hotel1$# END
hotel1$# $$;

hotel1$# $$;
CREATE PROCEDURE
hotel1=# |
```

- 2) Для получения информации о свободных номерах отеля на завтрашний день.

```
CREATE OR REPLACE FUNCTION free_room_on_the_next_day()
RETURNS TABLE(room_number integer, id_hotel integer, room_number_real
integer)
LANGUAGE plpgsql AS
$$
BEGIN
    return query(
        select distinct hotel.room.room_number, hotel.room.id_hotel,
hotel.room.room_number_real from hotel.room, hotel.book
        where hotel.room.room_number NOT IN
        (select distinct hotel.book.room_number from hotel.room,
hotel.book
        where (current_date + 1) BETWEEN hotel.book.date_start_visit
AND hotel.book.date_end_visit
        and hotel.room.room_number = hotel.book.room_number));

END;
$;
```

```
1 CREATE OR REPLACE FUNCTION free_room_on_the_next_day()
2 RETURNS TABLE(room_number integer, id_hotel integer, room_number_real integer)
3 LANGUAGE plpgsql AS
4 $$
5 BEGIN
6     return query(
7         select distinct hotel.room.room_number, hotel.room.id_hotel, hotel.room.room_number_real from hotel.room, hotel.book
8         where hotel.room.room_number NOT IN
9         (select distinct hotel.book.room_number from hotel.room, hotel.book
10        where (current_date + 1) BETWEEN hotel.book.date_start_visit AND hotel.book.date_end_visit
11        and hotel.room.room_number = hotel.book.room_number));
12
13 END;
14 $;
```

Создаю функцию через консоль

```
CREATE FUNCTION
hotel1=# CREATE OR REPLACE FUNCTION free_room_on_the_next_day()
hotel1=# RETURNS TABLE(room_number integer, id_hotel integer, room_number_real integer)
hotel1=# LANGUAGE plpgsql AS
hotel1=# $$
hotel1=# BEGIN
hotel1=# return query(
hotel1=# select distinct hotel.room.room_number, hotel.room.id_hotel, hotel.room.room_number_real from hotel.room, hotel.book
hotel1=# where hotel.room.room_number NOT IN
hotel1=# (select distinct hotel.book.room_number from hotel.room, hotel.book
hotel1=# where (current_date + 1) BETWEEN hotel.book.date_start_visit AND hotel.book.date_end_visit
hotel1=# and hotel.room.room_number = hotel.book.room_number));
hotel1=# END;
hotel1=# $$;
CREATE FUNCTION
hotel1=# |
```

3) Бронирования двухместного номера в гостинице на заданную дату и количество дней проживания.

```
CREATE OR REPLACE PROCEDURE to_book_room
(
    type_room varchar(40),
    id_passport varchar(40),
    date_start date,
    date_end date
)
LANGUAGE plpgsql AS
$$
BEGIN
CASE
WHEN (
    select room_number AS cl_room from hotel.room where hotel.room.type_room
= 'двухместный' AND
    room_number NOT IN
    (
        select room_number from hotel.book
        WHERE date_start BETWEEN hotel.book.date_start_visit AND
hotel.book.date_end_visit AND
        date_end BETWEEN hotel.book.date_start_visit AND
hotel.book.date_end_visit
    )
    limit 1
) IN (select room_number from hotel.book )
    THEN INSERT INTO hotel.book (room_number, id_worker, id_passport, date_start_visit,
date_end_visit)
        SELECT -----
        (
            select room_number AS cl_room from hotel.room where hotel.room.type_room
= 'двухместный' AND
            room_number NOT IN
            (
                select room_number from hotel.book
                WHERE date_start BETWEEN hotel.book.date_start_visit AND
hotel.book.date_end_visit AND
                date_end BETWEEN hotel.book.date_start_visit AND
hotel.book.date_end_visit
            )
            limit 1
        ),
        6, id_passport, date_start, date_end;
END CASE;
END;
$$;
```

Создаю процедуру через консоль

```
hotel1=# CREATE OR REPLACE PROCEDURE to_book_room
hotel1=# (
hotel1(# type_room varchar(40),
hotel1(# id_passport varchar(40),
hotel1(# date_start date,
hotel1(# date_end date
hotel1(# )
hotel1=# LANGUAGE plpgsql AS
hotel1=# $$
hotel1=# BEGIN
hotel1=# CASE
hotel1=# WHEN (
hotel1$# select room_number AS cl_room from hotel.room where hotel.room.type_room = 'двухместный' AND
hotel1$# room_number NOT IN
hotel1$# (
hotel1$# select room_number from hotel.book
hotel1$# WHERE date_start BETWEEN hotel.book.date_start_visit AND hotel.book.date_end_visit AND
hotel1$# date_end BETWEEN hotel.book.date_start_visit AND hotel.book.date_end_visit
hotel1$# )
hotel1$# limit 1
hotel1$# ) IN (select room_number from hotel.book )
hotel1$# THEN INSERT INTO hotel.book (room_number, id_worker, id_passport, date_start_visit, date_end_visit)
hotel1$#
OVERRIDING SELECT TABLE VALUES
hotel1$#
OVERRIDING SELECT TABLE VALUES
hotel1$# SELECT -----
hotel1$# (
hotel1$# select room_number AS cl_room from hotel.room where hotel.room.type_room = 'двухместный' AND
hotel1$# room_number NOT IN
hotel1$# (
hotel1$# select room_number from hotel.book
hotel1$# WHERE date_start BETWEEN hotel.book.date_start_visit AND hotel.book.date_end_visit AND
hotel1$# date_end BETWEEN hotel.book.date_start_visit AND hotel.book.date_end_visit
hotel1$# )
hotel1$# limit 1
hotel1$# ),
hotel1$# 6, id_passport, date_start, date_end;
hotel1$# END CASE;
hotel1$# END;
hotel1$# $$;
CREATE PROCEDURE
hotel1=#
```

4) Создать триггер для логирования событий вставки, удаления, редактирования данных в базе данных PostgreSQL (согласно индивидуальному заданию, часть 5). Допустимо создать универсальный триггер или отдельные триггеры на логирование действий.

- 1) Создать таблицу log
- 2) Создать триггерную функцию через консоль

```

1 CREATE OR REPLACE FUNCTION add_to_log() RETURNS TRIGGER AS $$
2 DECLARE
3     mstr varchar(30);
4     astr varchar(100);
5     retstr varchar(254);
6 BEGIN
7     IF TG_OP = 'INSERT' THEN
8         astr = NEW;
9         mstr := 'Add new user ';
10        retstr := mstr||astr;
11        INSERT INTO logs(text,added) values (retstr,NOW());
12        RETURN NEW;
13    ELSIF TG_OP = 'UPDATE' THEN
14        astr = NEW;
15        mstr := 'Update user ';
16        retstr := mstr||astr;
17        INSERT INTO logs(text,added) values (retstr,NOW());
18        RETURN NEW;
19    ELSIF TG_OP = 'DELETE' THEN
20        astr = OLD;
21        mstr := 'Remove user ';
22        retstr := mstr || astr;
23        INSERT INTO logs(text,added) values (retstr,NOW());
24        RETURN OLD;
25    END IF;
26 END;
27 $$ LANGUAGE plpgsql;
28

```

```

hotel1=# CREATE OR REPLACE FUNCTION add_to_log() RETURNS TRIGGER AS $$
hotel1$# DECLARE
hotel1$#     mstr varchar(30);
hotel1$#     astr varchar(100);
hotel1$#     retstr varchar(254);
hotel1$# BEGIN
hotel1$#     IF TG_OP = 'INSERT' THEN
hotel1$#         astr = NEW;
hotel1$#         mstr := 'Add new user ';
hotel1$#         retstr := mstr||astr;
hotel1$#         INSERT INTO logs(text,added) values (retstr,NOW());
hotel1$#         RETURN NEW;
hotel1$#     ELSIF TG_OP = 'UPDATE' THEN
hotel1$#         astr = NEW;
hotel1$#         mstr := 'Update user ';
hotel1$#         retstr := mstr||astr;
hotel1$#         INSERT INTO logs(text,added) values (retstr,NOW());
hotel1$#         RETURN NEW;
hotel1$#     ELSIF TG_OP = 'DELETE' THEN
hotel1$#         astr = OLD;
hotel1$#         mstr := 'Remove user ';
hotel1$#         retstr := mstr || astr;
hotel1$#         INSERT INTO logs(text,added) values (retstr,NOW());
hotel1$#         RETURN OLD;
hotel1$#     END IF;
hotel1$# END;
hotel1$# $$ LANGUAGE plpgsql;
CREATE FUNCTION
hotel1=#

```


3) Создать триггер

```
1 CREATE TRIGGER t_hotel AFTER INSERT OR UPDATE OR DELETE ON hotel.book FOR EACH ROW EXECUTE PROCEDURE add_to_log ();
```

```
[hotel=# CREATE TRIGGER t_hotel AFTER INSERT OR UPDATE OR DELETE ON hotel.book FOR EACH ROW EXECUTE PROCEDURE add_to_log ();  
CREATE TRIGGER  
hotel=#
```

4) Проверка

Query Editor

```
1 UPDATE hotel.book  
2 SET date_end_visit = CASE  
3     WHEN (( hotel.book.date_end_visit + 2) NOT IN(SELECT date_end_visit FROM hotel.book) AND room_number =  
4         (SELECT room_number FROM hotel.book WHERE id_passport = '12345675813'))  
5     THEN hotel.book.date_end_visit + 2  
6     ELSE hotel.book.date_end_visit  
7 END;  
8  
9
```

План выполнения Сообщения Notifications История запросов Результат





UPDATE 5

Запрос завершён успешно, время выполнения: 106 msec.

Query Editor

```
1 SELECT * FROM hotel.logs  
2
```

План выполнения Результат Сообщения Notifications История запросов

	 text 	 added  timestamp with time zone	
1	Update...	2022-05-03 23:53:31.3594...	
2	Update...	2022-05-03 23:53:31.3594...	
3	Update...	2022-05-03 23:53:31.3594...	
4	Update...	2022-05-03 23:53:31.3594...	
5	Update...	2022-05-03 23:53:31.3594...	

Выводы:

Созданы процедуры/функции триггер для логирования событий вставки, удаления, редактирования данных в базе данных PostgreSQL