Санкт-Петербургский национальный исследовательский университет
ИТМО

Факультет Инфокоммуникационных технологий

Лабораторная работа №3 по теме
“процедуры, функции, триггеры в PostgreSQL”
по дисциплине “Проектирование и реализация баз данных”

Выполнил:
студент 2 курса группы K32421
Безгин Алексей Геннадьевич
Преподаватель:
Говорова Марина Михайловна

Санкт-Петербург
2023

1. **Цель работы:** овладеть практическими создания и использования процедур, функций и триггеров в базе данных PostgreSQL.

2. **Практическое задание:**

   Вариант 1 (я выбрал его)

   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.

   3. Ход выполнения работы:

1) Напишем необходимые функции:

FUNCTION 1

Вывод данных о пассажирах, которые заказывали такси

в заданном, как параметр, временном интервале.

```
CREATE FUNCTION passengers_by_time_period(time_from timestamp, time_until timestamp)
RETURNS SETOF "Такси"."Пассажир" AS
$$
	SELECT * FROM "Такси"."Пассажир"
	WHERE "Телефон" IN
		(SELECT "Телефон_пассажира" FROM "Такси"."Вызов"
		 WHERE "Дата" BETWEEN time_from AND time_until)
$$ LANGUAGE SQL;
```

Пример работы:

```
SELECT * FROM passengers_by_time_period('2023-04-02 10:00:00', '2023-04-03 22:00:00')
```

```
Taxi=# CREATE FUNCTION passengers_by_time_period(time_from timestamp, time_until timestamp)
Taxi-# RETURNS SETOF "Такси"."Пассажир" AS
Taxi-# $$
Taxi$# SELECT * FROM "Такси"."Пассажир"
Taxi$# WHERE "Телефон" IN
Taxi$# (SELECT "Телефон_пассажира" FROM "Такси"."Вызов"
Taxi$#  WHERE "Дата" BETWEEN time_from AND time_until)
Taxi$# $$ LANGUAGE SQL;
CREATE FUNCTION
Taxi=# SELECT * FROM passengers_by_time_period('2023-04-02 10:00:00', '2023-04-03 22:00:00')
Taxi-# ;
   Телефон    |    Имя
--------------+-----------
 +77773332004 | Михаил Х.
 +79145627465 | Николай С.
 +79136767287 | Юрий Д.
 +73445627689 | Илья Я.
(4 rows)
```

FUNCTION 2

Вывести сведения о том, куда был доставлен пассажир

по заданному номеру телефона пассажира.

```
CREATE FUNCTION destinations_by_telephone(telephone varchar)
RETURNS table(destination varchar) AS
$$
        SELECT "Куда" FROM "Такси"."Вызов"
                WHERE "Статус" = 'Завершен' AND "Телефон_пассажира" = telephone
$$ LANGUAGE SQL;
```

Пример работы:

```
SELECT * FROM destinations_by_telephone('+73445627689')
```

```
Taxi=# CREATE FUNCTION destinations_by_telephone(telephone varchar)
Taxi-# RETURNS table(destination varchar) AS
Taxi-# $$
Taxi$# SELECT "Куда" FROM "Такси"."Вызов"
Taxi$#  WHERE "Статус" = 'Завершен' AND "Телефон_пассажира" = telephone
Taxi$# $$ LANGUAGE SQL;
CREATE FUNCTION
Taxi=# SELECT * FROM destinations_by_telephone('+73445627689')
Taxi-# ;
      destination
----------------------
 ул. Большая Морская 4
(1 row)
```

FUNCTION 3

Вычисление суммарного дохода таксопарка за истекший месяц.

```
CREATE FUNCTION last_month_profit()
RETURNS table(profit numeric) AS
$$
	SELECT ROUND(SUM("Наценка" + "Расстояние" *
		(SELECT "Цена_за_км" FROM "Такси"."Автомобиль" JOIN "Такси"."Тариф"
			ON "Автомобиль"."Код_тарифа" = "Тариф"."Код_тарифа"
			WHERE "Автомобиль"."Госномер" = "Вызов"."Госномер_автомобиля"))::numeric, 2)
	FROM "Такси"."Вызов"
	WHERE ("Такси"."Вызов"."Дата" > now() - interval'1 month')
		AND "Вызов"."Статус" = 'Завершен'
$$ LANGUAGE SQL;
```

Пример работы:

```
SELECT * FROM last_month_profit()
```

```
Taxi=# CREATE FUNCTION last_month_profit()
Taxi-# RETURNS table(profit numeric) AS
Taxi-# $$
Taxi$# SELECT ROUND(SUM("Наценка" + "Расстояние" *
Taxi$#    (SELECT "Цена_за_км" FROM "Такси"."Автомобиль" JOIN "Такси"."Тариф"
Taxi$# ON "Автомобиль"."Код_тарифа" = "Тариф"."Код_тарифа"
Taxi$# WHERE "Автомобиль"."Госномер" = "Вызов"."Госномер_автомобиля"))::numeric, 2)
Taxi$# FROM "Такси"."Вызов"
Taxi$#  WHERE ("Такси"."Вызов"."Дата" > now() - interval'1 month')
Taxi$#   AND "Вызов"."Статус" = 'Завершен'
Taxi$# $$ LANGUAGE SQL;
CREATE FUNCTION
Taxi=# SELECT * FROM last_month_profit();
 profit
---------
 9452.00
(1 row)
```

2) **Напишем триггер-логировщик согласно индивидуальному заданию.**

Создадим таблицу для логирования:

```
CREATE TABLE passenger_logs
(action_time timestamp without time zone,
 action_type varchar, telephone_old varchar,
 telephone_new varchar, name_old varchar, name_new varchar);
```

Создадим функцию, которая будет триггерить события:

```
CREATE OR REPLACE FUNCTION add_to_passenger_logs() RETURNS TRIGGER AS
$$
DECLARE
        oldtel varchar(12);
   newtel varchar(12);
        oldname varchar(50);
   newname varchar(50);
BEGIN
   IF TG_OP = 'INSERT' THEN
      newtel = NEW."Телефон";
      newname = NEW."Имя";
      INSERT INTO passenger_logs(action_time, action_type,
                                 telephone_old, telephone_new,
                                 name_old, name_new)
                VALUES (now(), TG_OP, oldtel, newtel, oldname, newname);
      RETURN NEW;
   ELSIF TG_OP = 'UPDATE' THEN
                oldtel = OLD."Телефон";
      oldname = OLD."Имя";
      newtel = NEW."Телефон";
```

```
        newname = NEW."Имя";
        INSERT INTO passenger_logs(action_time, action_type,
                                   telephone_old, telephone_new,
                                   name_old, name_new)
                VALUES (now(), TG_OP, oldtel, newtel, oldname, newname);
        RETURN NEW;
    ELSIF TG_OP = 'DELETE' THEN
        oldtel = OLD."Телефон";
        oldname = OLD."Имя";
        INSERT INTO passenger_logs(action_time, action_type,
                                   telephone_old, telephone_new,
                                   name_old, name_new)
                VALUES (now(), TG_OP, oldtel, newtel, oldname, newname);
        RETURN OLD;
    END IF;
END;
$$ LANGUAGE plpgsql;
```

Наконец, создадим триггер:

```
CREATE TRIGGER passenger_logs_trigger AFTER INSERT OR UPDATE OR DELETE
ON "Такси"."Пассажир" FOR EACH ROW EXECUTE PROCEDURE add_to_passenger_logs();
```

Протестируем его работу:

```
INSERT INTO "Такси"."Пассажир" VALUES('+79132638490', 'Александр Н.');
UPDATE "Такси"."Пассажир" SET "Имя" = 'Ирина Ш.' WHERE "Телефон" = '+79132638490';
DELETE FROM "Такси"."Пассажир" WHERE "Телефон" = '+79132638490';
```

SELECT * FROM passenger_logs;

Скриншот процедуры создания в SHELL:

```
Taxi=# create table passenger_logs (action_time timestamp without time zone, action_type varchar, telephone_old varchar, telephone_new varchar, name_old varchar, name_new varchar);
CREATE TABLE
Taxi=# CREATE OR REPLACE FUNCTION add_to_passenger_logs() RETURNS TRIGGER AS $$
Taxi$# DECLARE
Taxi$# oldtel varchar(12);
Taxi$#     newtel varchar(12);
Taxi$# oldname varchar(50);
Taxi$#     newname varchar(50);
Taxi$# BEGIN
Taxi$#     IF TG_OP = 'INSERT' THEN
Taxi$#         newtel = NEW."Телефон";
Taxi$#         newname = NEW."Имя";
Taxi$#         INSERT INTO passenger_logs(action_time, action_type,
Taxi$#    telephone_old, telephone_new,
Taxi$#    name_old, name_new)
Taxi$#
OVERRIDING  SELECT      TABLE       VALUES
Taxi$# VALUES (now(), TG_OP, oldtel, newtel, oldname, newname);
Taxi$#         RETURN NEW;
Taxi$#     ELSIF TG_OP = 'UPDATE' THEN
Taxi$# oldtel = OLD."Телефон";
Taxi$#         oldname = OLD."Имя";
Taxi$#         newtel = NEW."Телефон";
Taxi$#         newname = NEW."Имя";
Taxi$#         INSERT INTO passenger_logs(action_time, action_type,
Taxi$#    telephone_old, telephone_new,
Taxi$#    name_old, name_new)
Taxi$#
OVERRIDING  SELECT      TABLE       VALUES
Taxi$# VALUES (now(), TG_OP, oldtel, newtel, oldname, newname);
Taxi$#         RETURN NEW;
Taxi$#     ELSIF TG_OP = 'DELETE' THEN
Taxi$#         oldtel = OLD."Телефон";
Taxi$#         oldname = OLD."Имя";
Taxi$#         INSERT INTO passenger_logs(action_time, action_type,
Taxi$#    telephone_old, telephone_new,
Taxi$#    name_old, name_new)
Taxi$#
OVERRIDING  SELECT      TABLE       VALUES
Taxi$# VALUES (now(), TG_OP, oldtel, newtel, oldname, newname);
Taxi$#         RETURN OLD;
Taxi$#     END IF;
Taxi$# END;
Taxi$# $$ LANGUAGE plpgsql;
CREATE FUNCTION
Taxi=# CREATE TRIGGER passenger_logs_trigger AFTER INSERT OR UPDATE OR DELETE
Taxi-# ON "Такси"."Пассажир" FOR EACH ROW EXECUTE PROCEDURE add_to_passenger_logs();
CREATE TRIGGER
```

Результат работы:

```
Taxi=# INSERT INTO "Такси"."Пассажир" VALUES('+79132638490', 'Александр Н.');
INSERT 0 1
Taxi=# UPDATE "Такси"."Пассажир" SET "Имя" = 'Ирина Ш.' WHERE "Телефон" = '+79132638490';
UPDATE 1
Taxi=# DELETE FROM "Такси"."Пассажир" WHERE "Телефон" = '+79132638490';
DELETE 1
Taxi=# SELECT * FROM passenger_logs;
        action_time         | action_type | telephone_old | telephone_new |   name_old   |   name_new
----------------------------+-------------+---------------+---------------+--------------+--------------
 2023-04-26 23:13:53.298769 | INSERT      |               | +79132638490  |              | Александр Н.
 2023-04-26 23:14:00.365123 | UPDATE      | +79132638490  | +79132638490  | Александр Н. | Ирина Ш.
 2023-04-26 23:14:07.033705 | DELETE      | +79132638490  |               | Ирина Ш.     |
(3 rows)
```

5. Выводы.

В ходе выполнения лабораторной работы №3 мы овладели теоретическими и практическими навыками создания и использования процедур, функций и триггеров в базе данных PostgreSQL. Создали 3 функции и один триггер и проверили их работоспособность в контексте спроектированной ранее базы данных TAXI.