

Министерство науки и высшего образования Российской Федерации
федеральное государственное автономное образовательное учреждение
высшего образования
“Национальный исследовательский университет ИТМО”

Факультет инфокоммуникационных технологий

ЛАБОРАТОРНАЯ РАБОТА №5

Процедуры, функции, триггеры в PostgreSQL
по дисциплине:
«Проектирование и реализация баз данных»

Выполнил:

Бабичев Леонид Анатольевич
Группа К32422

Проверила:

Говорова Марина Михайловна

Санкт-Петербург
2023

ЦЕЛЬ РАБОТЫ:

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

Оборудование: компьютерный класс.

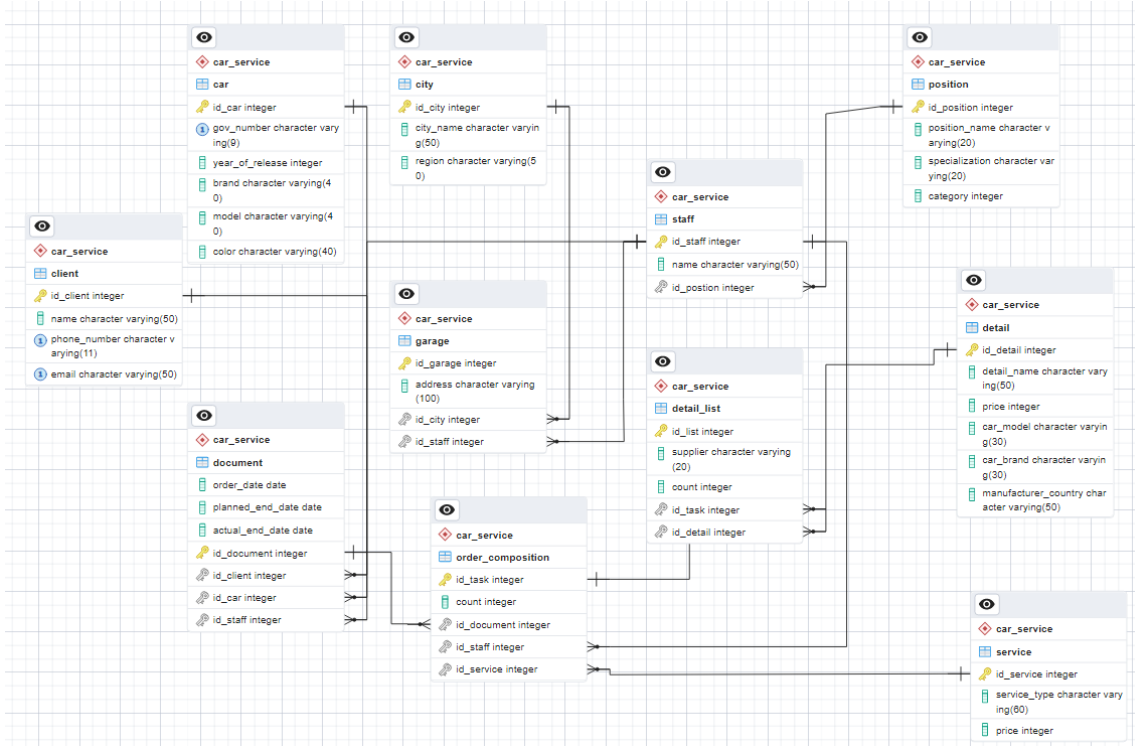
Программное обеспечение: СУБД PostgreSQL, SQL Shell (psql).

Вариант 1

Практическое задание:

- I. Создать процедуры/функции согласно индивидуальному заданию и (согласно индивидуальному заданию, часть 4).
- II. 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.

База данных:



Выполнение:

Задание 1. Создайте хранимые процедуры.

1) Повышения цены деталей на 10 %

```
car_service=# CREATE OR REPLACE PROCEDURE increase_ford_parts_price()
car_service=# LANGUAGE plpgsql
car_service=# AS $$
car_service$$ BEGIN
car_service$$   UPDATE car_service.detail
car_service$$   SET price = price * 1.1
car_service$$   WHERE car_brand = 'Subaru';
car_service$$ END;
car_service$$ $$;
CREATE PROCEDURE
car_service=# select * from car_service.detail;
```

id_detail	detail_name	price	car_model	car_brand	manufacturer_country
1	Львевьевы ЕюЁ	35000	Impreza Sport	Subaru	■яюэш
4	ЛрЁр	15000	V90 Cross Country	Volvo	†тхЎш
3	ТюЁьючэр ьюьюфър	50000	Valkyrie	Aston Martin	тхышьюсЁшЄрэш

(3 ёЁёюьш)

```
car_service=# CALL increase_ford_parts_price();
CALL
car_service=# select * from car_service.detail;
```

id_detail	detail_name	price	car_model	car_brand	manufacturer_country
4	ЛрЁр	15000	V90 Cross Country	Volvo	†тхЎш
3	ТюЁьючэр ьюьюфър	50000	Valkyrie	Aston Martin	тхышьюсЁшЄрэш
1	Львевьевы ЕюЁ	38500	Impreza Sport	Subaru	■яюэш

(3 ёЁёюьш)

2) Для повышения разряда тех мастеров, которые отремонтировали больше 3 автомобилей.

```
car_service=# CREATE OR REPLACE PROCEDURE increase_cat()
car_service=# LANGUAGE plpgsql
car_service=# AS $$
car_service$$ BEGIN
car_service$$ UPDATE car_service.position
car_service$$ SET category=category + 1
car_service$$ WHERE id_position in (select position.id_position from car_service.staff s join car_service.position on
car_service$$ position.id_position = s.id_position
car_service$$ join car_service.document d on
car_service$$ s.id_staff = d.id_staff
car_service$$ group by position.id_position
car_service$$ having count(distinct id_car) >= 3);
car_service$$ END;
car_service$$ $$;
CREATE PROCEDURE
car_service=# CALL increase_cat();
CALL
```

3) Сколько автомобилей отремонтировал каждый механик за истекший квартал.

```
SQL Shell (psql)
car_service=# CALL car_repair_count();
ОШИБКА: в запросе нет назначения для данных результата
ПОДСКАЗКА: Если вам нужно отбросить результаты SELECT, используйте PERFORM.
КОНТЕКСТ: функция PL/pgSQL car_repair_count(), строка 9, оператор SQL-оператор
car_service=# select car_repair_count();
ОШИБКА: car_repair_count() - процедура
СТРОКА 1: select car_repair_count();
ПОДСКАЗКА: Для вызова процедуры используйте CALL.
car_service=# CALL car_repair_count();
ОШИБКА: в запросе нет назначения для данных результата
ПОДСКАЗКА: Если вам нужно отбросить результаты SELECT, используйте PERFORM.
КОНТЕКСТ: функция PL/pgSQL car_repair_count(), строка 9, оператор SQL-оператор
car_service=# CREATE OR REPLACE FUNCTION cars_repaired_last_quarter(p_date DATE)
car_service=# RETURNS TABLE (
car_service=# "Табельный_номер" INTEGER,
car_service=# "ФИО" VARCHAR,
car_service=# "Количество_ремонтов" BIGINT
car_service=# )
car_service=# AS $$
car_service=# BEGIN
car_service=# RETURN QUERY
car_service=# SELECT
car_service=# s.id_staff AS "Табельный_номер",
car_service=# s.name AS "ФИО",
car_service=# COUNT(DISTINCT d.id_car) AS "Количество_ремонтов"
car_service=# FROM
car_service=# car_service.staff s
car_service=# JOIN car_service.document d ON s.id_staff = d.id_staff
car_service=# WHERE
car_service=# d.order_date BETWEEN DATE_TRUNC('quarter', p_date) AND (DATE_TRUNC('quarter', p_date) + INTERVAL '3 MONTHS' - INTERVAL '1 DAY')
car_service=# GROUP BY
car_service=# s.id_staff,
car_service=# s.name
car_service=# ORDER BY
car_service=# "Количество_ремонтов" DESC;
car_service=# END;
car_service=# $$ LANGUAGE plpgsql;
CREATE FUNCTION
car_service=# SELECT * FROM cars_repaired_last_quarter('2023-05-01');
Табельный_номер | ФИО | Количество_ремонтов
-----
2 | Илья Ильич | 2
1 | Ирина Сергеевна | 1
(2 строк)
car_service=#
```

Табельный_номер	ФИО	Количество_ремонтов
2	Илья Ильич	2
1	Ирина Сергеевна	1

Задание 2. Создать триггер для логирования событий вставки, удаления, редактирования

Создана таблица logs

```
car_service=# CREATE TABLE car_service.log_changes (id SERIAL PRIMARY KEY,
car_service(#  TABLE_NAME TEXT NOT NULL,
car_service(#  change_type TEXT NOT NULL,
car_service(#  change_time TIMESTAMP NOT NULL);
CREATE TABLE
car_service=# CREATE OR REPLACE FUNCTION log_changes() RETURNS TRIGGER AS $$
car_service$$ BEGIN
car_service$$ IF TG_OP = 'DELETE' THEN
car_service$$ INSERT INTO car_service.log_changes (table_name, change_type, change_time)
car_service$$ VALUES (TG_TABLE_NAME, 'DELETE', NOW());
car_service$$ ELSIF TG_OP = 'INSERT' THEN
car_service$$ INSERT INTO car_service.log_changes (table_name, change_type, change_time)
car_service$$ VALUES (TG_TABLE_NAME, 'INSERT', NOW());
car_service$$ ELSIF TG_OP = 'UPDATE' THEN
car_service$$ INSERT INTO car_service.log_changes (table_name, change_type, change_time)
car_service$$ VALUES (TG_TABLE_NAME, 'UPDATE', NOW());
car_service$$ END IF;
car_service$$ RETURN NEW;
car_service$$ END;
car_service$$ $$ LANGUAGE PLPGSQL;
CREATE FUNCTION
car_service=#
car_service=# CREATE TRIGGER log_changes_document_trigger AFTER
car_service-# INSERT
car_service-# OR
car_service-# UPDATE
car_service-# OR
car_service-# DELETE ON car_service.document
car_service-# FOR EACH ROW
car_service-# EXECUTE FUNCTION log_changes();
CREATE TRIGGER
car_service-#
```

Query

Query History

1

SELECT

2

* from car_service.log_changes

Data Output

Messages

Explain

×

Notifications

≡+

▼

	id [PK] integer	table_name text	change_type text	change_time timestamp without time zone
1	1	document	INSERT	2023-05-14 18:52:00.772929

Вывод:

В данной работе были изучены функции и процедуры, и созданы триггеры для корректного хранения данных.