Министерство науки и высшего образования Российской Федерации
федеральное государственное автономное образовательное учреждение высшего образования
«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»

Отчет

по лабораторной работе №5 «ПРОЦЕДУРЫ, ФУНКЦИИ, ТРИГГЕРЫ В POSTGRESQL »

по дисциплине **«Проектирование и реализация баз данных»**

Автор: Оспельников А. В.

Факультет: ИКТ

Группа: К3240

Преподаватель: Говорова М.М.

Санкт-Петербург 2023

Оглавление

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).

Практическое задание:

Вариант 1 (max - 6 баллов)

1. Создать процедуры/функции 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 (max - 8 баллов)

1. Создать процедуры/функции согласно индивидуальному заданию (часть 4).

2.1. 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).

2.2. Создать авторский триггер по варианту индивидуального задания.

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

Вариант 18. БД «ГИБДД»

Описание предметной области:

Описание предметной области: ГИБДД производит регистрацию автомобилей и следит за безопасностью дорожного движения. БД служит для ведения статистики нарушений правил дорожного движения и аварий. В одной аварии водитель может нарушить несколько ПДД. У одной аварии может быть несколько участников – виновников и потерпевших. Статус участника аварии может быть неопределенным. В системе должна храниться история штрафов водителей за нарушения ППД и статус их оплаты.

БД должна содержать следующий минимальный набор сведений: Номер водительского удостоверения. ФИО водителя. Адрес. Номер телефона. Номер автомобиля. Марка автомобиля. Модель автомобиля. Год выпуска. Дата регистрации в ГИБДД. Код нарушения. Вид нарушения. Сумма штрафа. Срок лишения прав управления автомобилем. Дата нарушения. Время нарушения. Район аварии/нарушения. Улица аварии/нарушения. Личный номер инспектора. ФИО инспектора. Дата аварии. Виновность владельца. Описание аварии.

Выполнение

Функции согласно заданию

1) Вывести все сведения о владельце автомобиля по заданному, как параметр номеру автомобиля.

```
CREATE OR REPLACE FUNCTION show_driver_data(car_reg_number character(9))
RETURNS TABLE (
	full_name character varying(50),
	phone_number bigint, adress text,
	passport_number bigint,
	driving_licence_number bigint
)
LANGUAGE SQL AS $$
SELECT full_name, phone_number, adress, passport_number, driving_licence_number
FROM "Police-schema"."Car owner"
WHERE driving_licence_number IN (SELECT driving_licence_number
                                 FROM "Police-schema"."Registred car"
                                 WHERE registration_number = car_reg_number
	);
$$;
```

2) Вывести данные инспектора, оштрафовавшего одного и того же водителя более одного раза.

```
CREATE OR REPLACE FUNCTION show_inspector_data()
RETURNS TABLE (
	full_name character varying(50),
```

```
    phone_number bigint
    rank character varying(30)
)
LANGUAGE SQL AS $$
    SELECT full_name, phone_number, adress, rank
    FROM "Police-schema"."Traffic inspector"
    WHERE traffic_inspector_personal_code IN (
        SELECT traffic_inspector_personal_code
        FROM "Police-schema"."Traffic accident"
        GROUP BY car_code, traffic_inspector_personal_code
        HAVING COUNT(traffic_inspector_personal_code) > 1

$$;
```

3) Вывести количество нарушений, повлекших лишение прав в заданном, как параметр районе.

```
CREATE OR REPLACE FUNCTION show_takeoff_licence_data(violation_district text)
RETURNS integer
LANGUAGE SQL AS $$
    SELECT COUNT(driving_licence_number)
    FROM "Police-schema"."Registred car"
    WHERE car_code IN (
        SELECT DISTINCT vehicle_code
        FROM "Police-schema"."Registred car"
        WHERE car_code IN (
            SELECT DISTINCT car_code
            FROM "Police-schema"."Traffic laws violation"
            WHERE violation_place = violation_district
            GROUP BY violation_code, car_code
            HAVING (violation_code IN (
                SELECT violation_code FROM "Police-schema"."Violation manual" WHERE punishment LIKE '%taking out driving licence%')
                ) AND
            Count(car_code) > 1
        )
    );
```

$$;

Авторские триггеры

1) Если водитель попадает в аварию с просроченной регистрацией, ему автоматически начисляется штраф

```
CREATE OR REPLACE FUNCTION outdate_registration()
RETURNS TRIGGER AS $$
BEGIN
	IF EXISTS (SELECT car_code FROM "Police-schema"."Registred car" WHERE end_registration_date < current_date - 10 AND
				(car_code = NEW.car_code OR car_code = NULL)
				)
	THEN
		INSERT INTO "Police-schema"."Traffic laws violation" (violation_place, car_code, violation_code, violation_date)
		VALUES (
			NEW.accident_place,
			NEW.car_code,
			(SELECT violation_code FROM "Police-schema"."Violation manual" WHERE violation_name = 'Outdate registration'),
			NEW.accident_date
		);
	END IF;
	RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE OR REPLACE TRIGGER add_outdate_registration
AFTER INSERT ON "Police-schema"."Traffic accident"
FOR EACH ROW EXECUTE PROCEDURE outdate_registration();
```

2) Как только водитель попадает в аварию, его машине автоматически добавляется участие в 1 аварии

```
CREATE OR REPLACE FUNCTION add_accident()
RETURNS TRIGGER AS $$
BEGIN
	UPDATE "Police-schema"."Car"
```

```
        SET accident_amount = accident_amount + 1
        WHERE vehicle_code IN (SELECT vehicle_code
                                    FROM "Police-schema"."Registred car"
                                    WHERE car_code IN (SELECT car_code
                                                                FROM "Police-schema"."Traffic accident"
                                                                WHERE car_code = NEW.car_code));
        RETURN NEW;
END
$$ LANGUAGE plpgsql;

CREATE OR REPLACE TRIGGER update_accident_amount
AFTER INSERT ON "Police-schema"."Traffic accident"
FOR EACH ROW EXECUTE PROCEDURE add_accident ();
```

Вывод

В данной лабораторной работе я научился работать с функциями и триггерами в POSGRESQL, написал функции по идивидуальному заданию и создал 2 своих триггера, которые автоматизируют работу с базой данных.