

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
УНИВЕРСИТЕТ ИТМО»**

Факультет инфокоммуникационных технологий

Дисциплина:

«Проектирование и реализация баз данных»

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №3

«Процедуры, функции, триггеры в PostgreSQL»

Выполнил:

студент группы К32392

Жаров Александр Павлович

(подпись)

Проверил(а):

Говорова Марина Михайловна

(отметка о выполнении)

(подпись)

Санкт-Петербург
2023 г.

Цель работы: овладеть практическими создания и использованияпроцедур, функций и триггеров в базе данных PostgreSQL.

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

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

2.

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.

Индивидуальное практическое задание:

База данных “Прокат автомобилей”

Задание 4:

1. Выполнить списание автомобилей, выпущенных ранее заданного года.
2. Выдачи автомобиля и расчета стоимости с учетом скидки постоянным клиентам.
3. Для вычисления количества автомобилей заданной марки

Схема базы данных:

	car_number [PK] integer	vin integer	year_of_issue date	car_model text	car_mileage integer
77	75	1234509	2019-01-01	XC90	100000
78	76	2345610	2020-01-01	XC60	110000
79	77	3456721	2021-01-01	Q5	120000
80	78	4567832	2010-01-01	Polo	130000
81	79	5678943	2011-01-01	XC90	140000
82	80	6789054	2012-01-01	XC60	150000
83	81	7890165	2013-01-01	Q5	160000
84	82	8901276	2014-01-01	Polo	170000
85	83	9012387	2015-01-01	XC90	180000
86	84	1234598	2016-01-01	XC60	190000
87	85	2345609	2017-01-01	Q5	200000
88	86	3456720	2018-01-01	Polo	210000
89	87	4567832	2019-01-01	XC90	220000
90	88	5678943	2020-01-01	XC60	230000
91	89	6789054	2021-01-01	Q5	240000
92	90	7890165	2010-01-01	Polo	250000

После вызова:

	car_number [PK] integer	vin integer	year_of_issue date	car_model text	car_mileage integer
71	73	8901234	2017-01-01	Q5	80000
72	74	9012345	2018-01-01	Polo	90000
73	75	1234509	2019-01-01	XC90	100000
74	76	2345610	2020-01-01	XC60	110000
75	77	3456721	2021-01-01	Q5	120000
76	79	5678943	2011-01-01	XC90	140000
77	80	6789054	2012-01-01	XC60	150000
78	81	7890165	2013-01-01	Q5	160000
79	82	8901276	2014-01-01	Polo	170000
80	83	9012387	2015-01-01	XC90	180000
81	84	1234598	2016-01-01	XC60	190000
82	85	2345609	2017-01-01	Q5	200000
83	86	3456720	2018-01-01	Polo	210000
84	87	4567832	2019-01-01	XC90	220000
85	88	5678943	2020-01-01	XC60	230000
86	89	6789054	2021-01-01	Q5	240000

2. Выдачи автомобиля и расчета стоимости с учетом скидки постоянным клиентам.

```

CREATE OR REPLACE FUNCTION "LR_2".car_issue(in_id_contract integer, in_client_id integer,
                                             car_number integer, in_car_model text, full_name text,
                                             in_start_of_rental date, in_end_of_rental date,
                                             in_discount integer)

RETURNS integer
LANGUAGE plpgsql
AS $$
DECLARE
    car_cost integer;
    total_cost integer;
    out_cost integer;
BEGIN
    SELECT cost_per_day INTO car_cost FROM "LR_2".model WHERE car_model = in_car_model;
    total_cost := (in_end_of_rental - in_start_of_rental) * car_cost;
    total_cost := total_cost - (total_cost * in_discount / 100);

    INSERT INTO "LR_2".contract(id_contract, id_client, car_number, workers_full_name, date_of_signing,
                                start_of_rental, end_of_rental, conformation_status)
    VALUES (in_id_contract, in_client_id, car_number, full_name, CURRENT_DATE,
            in_start_of_rental, in_end_of_rental, 'confirmed')

    RETURNING id_contract INTO out_cost;
    RETURN total_cost;
END;
$$;

```

После выполнения:

	car_issue integer
1	7200

3. Для вычисления количества автомобилей заданной марки

```

CREATE OR REPLACE FUNCTION "LR_2".count_cars_by_model(in_car_model text)
RETURNS integer
LANGUAGE plpgsql
AS $$
DECLARE
    out_count integer;
BEGIN
    SELECT COUNT(*) INTO out_count FROM "LR_2".car WHERE car_model = in_car_model;
    RETURN out_count;
END;
$$;

```

После выполнения:

	count_cars_by_model integer
1	23

4. Создаем триггер для логирования событий INSERT DELETE UPDATE

Создадим табличку для записи логов изменения таблицы contract:

```
CREATE TABLE "LR_2".contract_audit_log (
  id SERIAL PRIMARY KEY,
  operation VARCHAR(10) NOT NULL,
  id_contract INTEGER,
  id_client INTEGER,
  workers_full_name TEXT,
  car_number INTEGER,
  date_of_signing DATE,
  start_of_rental DATE,
  end_of_rental DATE,
  conformation_status TEXT,
  name_of_insurance TEXT,
  log_timestamp TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP
);
```

Создаем функцию, которая будет записывать логи в таблицу:

```
CREATE FUNCTION "LR_2".log_contract_changes() RETURNS TRIGGER
LANGUAGE plpgsql
AS $$
BEGIN
  IF TG_OP = 'INSERT' THEN
    INSERT INTO "LR_2".contract_audit_log (operation, id_contract, id_client, workers_full_name, car_number,
      date_of_signing, start_of_rental, end_of_rental, conformation_status,
      name_of_insurance)
    VALUES ('INSERT', NEW.id_contract, NEW.id_client, NEW.workers_full_name, NEW.car_number, NEW.date_of_signing,
      NEW.start_of_rental, NEW.end_of_rental, NEW.conformation_status, NEW.name_of_insurance);
    RETURN NEW;
  ELSIF TG_OP = 'UPDATE' THEN
    INSERT INTO "LR_2".contract_audit_log (operation, id_contract, id_client, workers_full_name, car_number,
      date_of_signing, start_of_rental, end_of_rental, conformation_status,
      name_of_insurance)
    VALUES ('UPDATE', NEW.id_contract, NEW.id_client, NEW.workers_full_name, NEW.car_number, NEW.date_of_signing,
      NEW.start_of_rental, NEW.end_of_rental, NEW.conformation_status, NEW.name_of_insurance);
    RETURN NEW;
  ELSIF TG_OP = 'DELETE' THEN
    INSERT INTO "LR_2".contract_audit_log (operation, id_contract, id_client, workers_full_name, car_number,
      date_of_signing, start_of_rental, end_of_rental, conformation_status, name_of_
    VALUES ('DELETE', OLD.id_contract, OLD.id_client, OLD.workers_full_name, OLD.car_number, OLD.date_of_signing,
      OLD.start_of_rental, OLD.end_of_rental, OLD.conformation_status, OLD.name_of_insurance);
    RETURN OLD;
  END IF;
END;
$$;
```

Создаем триггер, который будет срабатывать на UPDATE INSERT DELETE в таблицу contract:

```
CREATE TRIGGER log_contract_changes_after
AFTER INSERT OR UPDATE OR DELETE ON "LR_2".contract
FOR EACH ROW
EXECUTE FUNCTION "LR_2".log_contract_changes();
```

Проверяем его работу, выполнив несколько функций и сделав select таблицы логов:

	id [PK] integer	operation character varying (10)	id_contract integer	id_client integer	workers_full_name text	car_number integer	date_of_signing date	start_of_rental date	end_of_rental date	conformation_status text	name_of_insurance text
1	1	INSERT	23	1	Иван Иванович Иванов	123	2022-10-02	2022-10-20	2022-10-20	Обрабатывается	[null]
2	2	DELETE	23	1	Иван Иванович Иванов	123	2022-10-02	2022-10-20	2022-10-20	Обрабатывается	[null]

Выводы

В результате работы были изучены и применены различные функции и процедуры в PostgreSQL, а также разработан триггер для логирования операций INSERT, UPDATE и DELETE.