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

ОТЧЕТ
ПО ЛАБОРАТОРНОЙ РАБОТЕ № 3

по теме: процедуры, функции, триггеры в PostgreSQL
по дисциплине: Проектирование и реализация баз данных

Специальность:

09.03.03 Мобильные и сетевые технологии

Проверила:

Говорова М.М.
Дата: «__» _____ 2023г.
Оценка _____

Выполнила:

студент группы К32391

Тюлюкин Игорь

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

Практическое задание:

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.

Индивидуальное задание:

Вариант 14. БД «Служба заказа такси»

Описание предметной области: Система должна фиксировать все вызовы такси и распределять их между водителями.

Каждому водителю ежедневно начисляется заработная плата в зависимости от количества вызовов и их тарифа (50% от заработанной им суммы). Автомобили могут быть собственностью компании или таксиста.

Заказ принимает дежурный администратор и передает его водителю. В заказе фиксируется тип оплаты – наличными или онлайн.

В системе необходимо хранить график работы водителей.

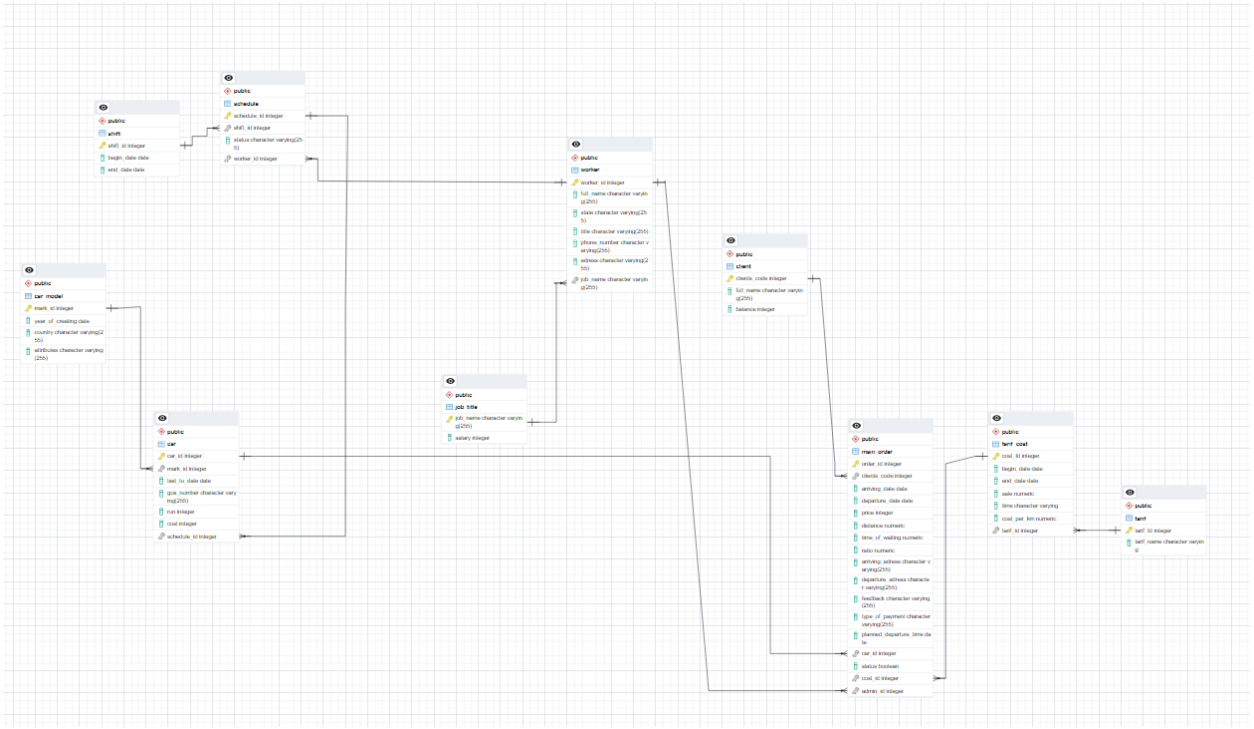
Ежедневно действуют базовые тарифы на тип предоставляемых авто, но в зависимости от времени суток и ситуации на дорогах, цена может корректироваться.

БД должна содержать следующий минимальный набор сведений: Код сотрудника. ФИО сотрудника. Адрес сотрудника. № телефона сотрудника. Паспортные данные сотрудника. Должность сотрудника. Категория сотрудника. Наименование модели и марки автомобиля. Технические характеристики. Стран-производитель. Стоимость. Код тарифа. Наименование тарифа. Цена за километр. Код автомобиля. Госномер автомобиля. Год выпуска. Пробег. Дата последнего ТО. Дата вызова. Время посадки пассажира. Время высадки пассажира. Номер телефона пассажира. Откуда. Куда. Расстояние. Штраф за время ожидания (в минутах). Оплата (онлайн (при заказе) или наличными). Рекламация клиента на вызов.

Задание 4.
Создать хранимые процедуры:
Для вывода данных о пассажирах, которые заказывали такси в заданном, как параметр, временном интервале.
Вывести сведения о том, куда был доставлен пассажир по заданному имени пассажира.
Для вычисления суммарного дохода таксопарка за истекший месяц.

Задание 5.
Создать необходимые триггеры.

Схема логической модели базы данных:



Выполнение

1.1 **Запросы:**

1) Для вывода данных о пассажирах, которые заказывали такси в заданном, как

```
postgres=# CREATE OR REPLACE FUNCTION get_passengers(first_date DATE, second_date DATE)
postgres-# RETURNS TABLE ( clients_code integer, full_name character varying(255), balance integer) as $$ BEGIN
postgres$# RETURN QUERY
postgres$#  SELECT * from client where client.clients_code in (SELECT main_order.clients_code from main_order where departure_date BETWEEN first_date and second_date);
postgres$# END;
postgres$#  $$ LANGUAGE plpgsql
postgres-# ;
CREATE FUNCTION
postgres=# select * from get_passengers('01/01/1990','23/05/2023');
 clients_code | full_name | balance
--------------+-----------+---------
            1 | Boris     |       1
            4 | Vard      |     505
(2 строки)


postgres=#
```

параметр, временном интервале.

2) Вывести сведения о том, куда был доставлен пассажир по заданному имени пассажира.

```
postgres=# create or replace function passanger_adress(name character varying(255))
postgres-# returns table(arriving_adress character varying(255)) as $$ begin
postgres$# return query
postgres$# select main_order.arriving_adress from main_order join client using(clients_code) where client.full_name = name;
postgres$# end;
postgres$# $$ LANGUAGE plpgsql;
CREATE FUNCTION
postgres=#  select * from passanger_adress('Boris')
postgres-# ;
 arriving_adress
-----------------
 Visnyovaya
(1 строка)
```

3) Для вычисления суммарного дохода таксопарка за истекший месяц.

```
postgres=# create or replace function money() returns integer as $$ begin
postgres$# return
postgres$# (select sum(price) as sum from main_order where extract(month from arriving_date) =  extract(month from now())-1 and  extract(year from arriving_date) =  extract(year from now()));
postgres$# end;
postgres$# $$ LANGUAGE plpgsql;
CREATE FUNCTION
postgres=# select * from money()
postgres-# ;
 money
-------
   500
(1 строка)
```

1.2 Триггер

1)

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

```
postgres=# CREATE TRIGGER t_client AFTER INSERT OR UPDATE OR DELETE ON
postgres-# client FOR EACH ROW EXECUTE PROCEDURE
postgres-# add_to_log ();
CREATE TRIGGER
postgres=# insert into client (clients_code,full_name,balance) values(15,'Гой',53535);
INSERT 0 1
postgres=# update client set full_name= 'Йор' where clients_code = 15;
UPDATE 1
postgres=# delete from client where clients_code = 15;
DELETE 1
postgres=#
```

ЗАКЛЮЧЕНИЕ

SQL запросы позволяют изменять, добавлять или удалять данные, а также составлять различные выборки, подсчитывать числовые характеристики. Сравнив время выполнения запросов с индексами и без, можно сделать вывод, что с индексами запросы выполнялись примерно столько же. Это связано с небольшим количеством данных в таблице.