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

ОТЧЕТ
О ЛАБОРАТОРНОЙ РАБОТЕ №

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

Специальность:
45.03.04 Интеллектуальные системы в гуманитарной сфере

Проверил:
Говорова М.М.
Дата: «22» мая 2022 г.
Оценка _____

Выполнила:
студентка группы К3243
Нургазизова А.Р.

Санкт-Петербург 2021/2022

Цель работы

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

Практическое задание

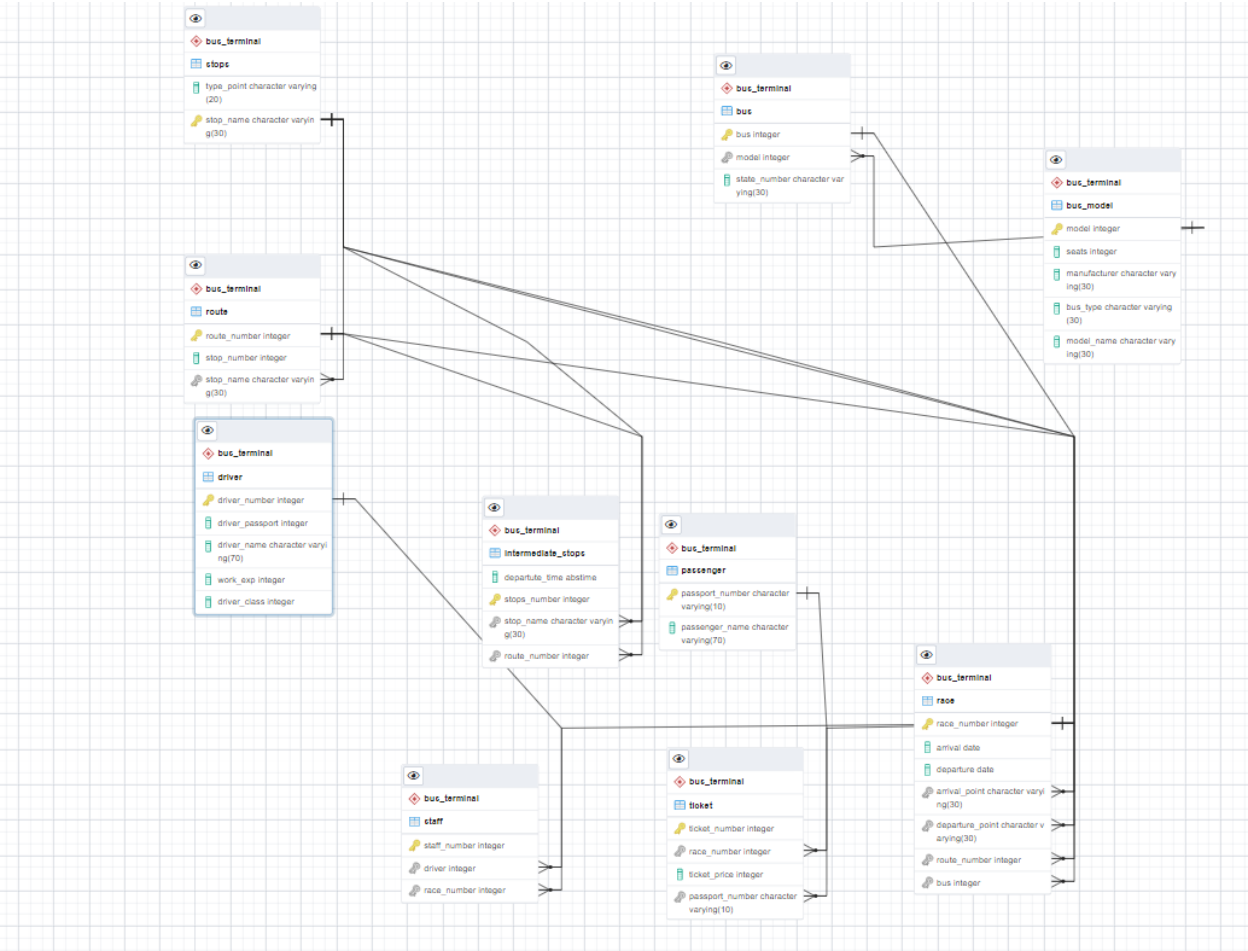
1. Создать процедуры/функции согласно индивидуальному заданию и (согласно индивидуальному заданию, часть 4).

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.

Выполнение

I. Наименование БД

bus_terminal.



II. Схема логической модели БД, сгенерированная в ERD

Рис. 1 — Схема логической модели БД, сгенерированная в ERD

III. Создать хранимые процедуры

1. Продажа билетов

```
SET search_path TO bus_terminal,public;
CREATE FUNCTION s_ticket(price integer)
returns table(passport_number varchar(10), passenger_name varchar(70), ticket_number integer, ticket_price integer)
AS
$$
BEGIN
RETURN QUERY
SELECT bus_terminal.ticket.passport_number,
bus_terminal.passenger.passenger_name,
bus_terminal.ticket.ticket_number,
bus_terminal.ticket.ticket_price
FROM bus_terminal.ticket, bus_terminal.passenger, bus_terminal.race
WHERE
bus_terminal.ticket.ticket_price=price
AND
bus_terminal.ticket.passport_number=bus_terminal.passenger.passport_number
AND date(departure) = '20210503';
END;
$$ LANGUAGE plpgsql;

select * from s_ticket(500)
```

Data Output | Explain | Messages | Notifications

	passport_number character varying	passenger_name character varying	ticket_number integer	ticket_price integer
1	6716529183	Железнова Мария	1	500
2	6715432678	Мин Юнги	2	500
3	2040532681	Зубова Екатерина	3	500
4	4387290146	Нургазизова Айзиля	4	500

2. Возврат билетов

Query Editor | Query History

```
1  SET search_path TO bus_terminal,public;
2  create function ticket_r(departure date)
3  returns table(status boolean, ticket_number integer, race_number integer)
4  as
5  $$
6  begin
7  return query
8  select bus_terminal.ticket.status, bus_terminal.ticket.ticket_number, bus_terminal.race.race_number
9  from bus_terminal.ticket, bus_terminal.race
10 where bus_terminal.ticket.race_number=bus_terminal.race.race_number;
11 end;
12 $$ language plpgsql;
13
14 select * from ticket_r('20210502')
```

Data Output | Explain | Messages | Notifications

	status boolean	ticket_number integer	race_number integer
1	true	1	1
2	true	2	2
3	true	3	3
4	false	4	3

3. Добавления нового рейса

```
1  SET search_path TO bus_terminal,public;
2  create function new_race()
3  returns table(race_number int, arrival date, departure date, arrival_point varchar(30), departure_point varchar(30),
4  as
5  $$
6  begin
7  insert into bus_terminal.race values (7, '20210809', '20210807', 'Екатеринбург', 'Казань', '2', '3');
8  end;
9  $$ language plpgsql;
10
11 select * from new_race()
```

IV. Создать триггерную функцию

```
SET search_path TO bus_terminal,public;
CREATE OR REPLACE FUNCTION add_to_log()
RETURNS TRIGGER AS $$
DECLARE
mstr varchar(30);
astr varchar(100);
retstr varchar(254);
BEGIN
IF TG_op = 'INSERT' THEN
astr = NEW;
mstr := 'Add new';
retstr := mstr || astr;
INSERT INTO
bus_terminal.logs(text, added, table_name) values
(retstr, NOW(), TG_TABLE_NAME);
RETURN NEW;
ELSEIF TG_OP = 'UPDATE' THEN
astr = NEW;
mstr := 'Update';
retstr := mstr || astr;
INSERT INTO
bus_terminal.logs(text, added, table_name) values
(retstr, NOW(), TG_TABLE_NAME);
RETURN NEW;
ELSEIF TG_OP = 'DELETE' THEN
astr = OLD;
mstr := 'Remove';
retstr := mstr || astr;
INSERT INTO
bus_terminal.logs(text, added, table_name) values
(retstr, NOW(), TG_TABLE_NAME);
RETURN OLD;
END IF;
END;
$$ LANGUAGE plpgsql;
```

Data Output Explain Messages Notifications

```
CREATE FUNCTION

Query returned successfully in 79 msec.
```

```
CREATE TRIGGER st_bus AFTER INSERT OR UPDATE OR DELETE
ON bus_terminal.stops
FOR EACH ROW EXECUTE PROCEDURE add_to_log();
```

Data Output Explain Messages Notifications

```
CREATE TRIGGER

Query returned successfully in 61 msec.
```

```sql
INSERT INTO bus_terminal.stops(type_point, stop_name) VALUES ('село', 'Москва');
UPDATE bus_terminal.stops SET stop_name='Архангельск' WHERE type_point='деревня';
DELETE FROM bus_terminal.stops WHERE stop_name='Москва'
```

Выводы

В ходе выполнения лабораторной работы были созданы хранимые процедуры и триггерная функция.