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 и триггеры в PostgreSQL»

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

Автор: Ананьев Н. В.

Факультет: ИКТ

Группа: К3240

Преподаватель: Говорова М.М.

Санкт-Петербург 2023

Цель работы

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

Практическое задание

Вариант 2 (max - 8 баллов)

1. Создать процедуры 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).

Схема базы данных

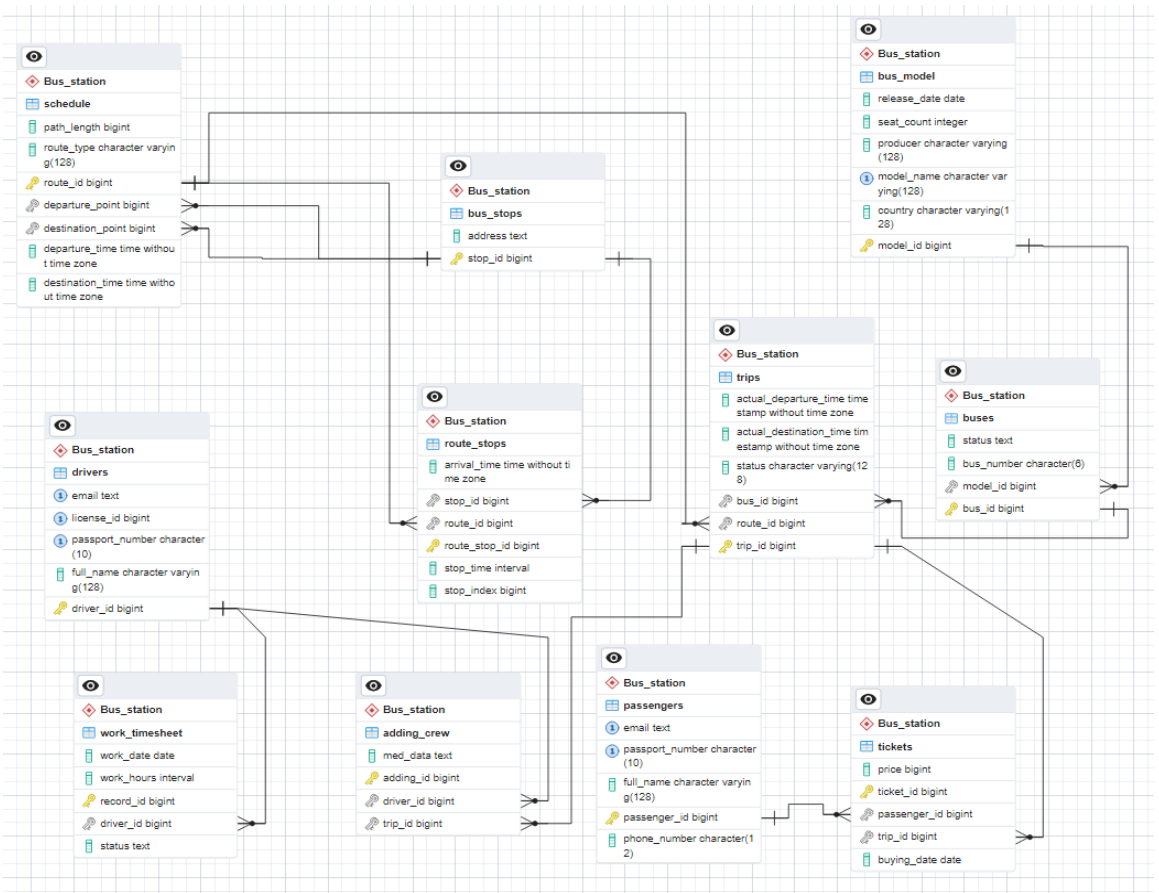

Рисунок 1 — ERD-схема базы данных

Выполнение

Задание 4. Создать хранимые процедуры:
Продажи билета:

```
CREATE PROCEDURE Ticket_buying
(
	id_passenger INT,
	id_trip INT
)
LANGUAGE plpgsql
AS $$
DECLARE
	reserved INT;
	id_bus INT;
	id_model INT;
	count INT;
BEGIN
	reserved := (SELECT COUNT(*) FROM "Bus_station".tickets WHERE trip_id = id_trip);
	id_bus := (SELECT bus_id FROM "Bus_station".trips WHERE trip_id=id_trip);
	id_model := (SELECT model_id FROM "Bus_station".buses WHERE bus_id=id_bus);
	count := (SELECT seat_count FROM "Bus_station".bus_model WHERE model_id=id_model);
	DECLARE remain INT := count - reserved;
	BEGIN
		IF remain > 0 THEN
			INSERT INTO "Bus_station".tickets (passenger_id, trip_id, buying_date)
			VALUES (id_passenger, id_trip, CURRENT_DATE);
		END IF;
	END;
END
$$;
```

Возврата билета:

```
CREATE PROCEDURE Ticket_return
(
	id_ticket INT
)
LANGUAGE plpgsql
AS $$
BEGIN
	IF EXISTS (SELECT * FROM "Bus_station".tickets WHERE ticket_id=id_ticket) THEN
		DELETE FROM "Bus_station".tickets WHERE ticket_id=id_ticket;
	END IF;
END
$$;
```

Добавления нового рейса:

```
CREATE PROCEDURE Add_trip
(
	id_route INT,
	id_bus INT,
	trip_price INT,
	planned_date DATE
)
LANGUAGE plpgsql
AS $$
DECLARE
	bus_trips_count INT;
BEGIN
	bus_trips_count := (SELECT COUNT(*) FROM "Bus_station".trips WHERE bus_id=id_bus AND DATE(actual_departure_time)=planned_date);
	BEGIN
		IF bus_trips_count = 0 THEN
			INSERT INTO "Bus_station".trips (actual_departure_time, actual_destination_time, status, bus_id, route_id, price)
			VALUES (NULL, NULL, 'planned', id_bus, id_route, trip_price);
		END IF;
	END;
END
$$;
```

Задание 5. Создать необходимые триггеры (вариант 2.2)

Триггер, проверяющий добавляемого в поездку водителя на то, что в этот день у него нет другого рейса

```
CREATE OR REPLACE FUNCTION fn_check_adding_crew() returns
trigger as $psql$
BEGIN
	DECLARE trip_date DATE := (SELECT DATE(actual_departure_time) FROM "Bus_station".trips WHERE trip_id=new.trip_id);
	BEGIN
	IF new.driver_id IN (SELECT trip_crew.driver_id FROM (SELECT * FROM "Bus_station".adding_crew
			JOIN "Bus_station".trips ON "Bus_station".adding_crew.trip_id="Bus_station".trips.trip_id) AS trip_crew
			WHERE DATE(trip_crew.actual_departure_time) = trip_date) THEN
		return NULL;
	END IF;
	return new;
	END;

END;
$psql$ LANGUAGE plpgsql;


CREATE TRIGGER check_adding_crew before insert ON "Bus_station".adding_crew
for each row execute procedure fn_check_adding_crew();
```

Триггер, срабатывающий на изменение времени окончания поездки - после чего отработанное водителем время вносится в табель рабочего времени.

```
CREATE OR REPLACE FUNCTION fn_update_timesheet() returns
trigger as $psql$
BEGIN
	BEGIN
		IF NEW.actual_departure_time IS NULL THEN
			return NULL;
		END IF;
	END;
	DECLARE id_driver INT := (SELECT driver_id FROM "Bus_station".adding_crew WHERE trip_id=NEW.trip_id);
	BEGIN
		IF id_driver IS NOT NULL THEN
			INSERT INTO "Bus_station".work_timesheet (work_date, work_hours, driver_id, status)
			VALUES (DATE(NEW.actual_departure_time), NEW.actual_destination_time - NEW.actual_departure_time, id_driver, 'ok');
			return NEW;
		END IF;
		return NULL;
	END;
END;
$psql$ LANGUAGE plpgsql;

CREATE TRIGGER update_timesheet after update ON "Bus_station".trips
for each row execute procedure fn_check_adding_crew();
```

Вывод

В ходе выполнения этой лабораторной работы были изучены такие инструменты работы с базами данных как функции, процедуры и триггеры. Они оказались крайне полезны в своем применении, так как с их помощью можно предотвращать множество ошибок несоответствия данных, а также автоматизировать некоторые процессы работы с данными.