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»

Обучающийся Люсин Дмитрий Витальевич
Факультет прикладной информатики
Группа K3239
Направление подготовки 09.03.03 Прикладная информатика
Образовательная программа Мобильные и сетевые технологии 2023
Преподаватель Говорова Марина Михайловна

Санкт-Петербург
2025/2026

1. **<u>Цель работы:</u>** овладеть практическими создания и использования процедур, функций и триггеров в базе данных PostgreSQL.
2. **<u>Практическое задание:</u>**

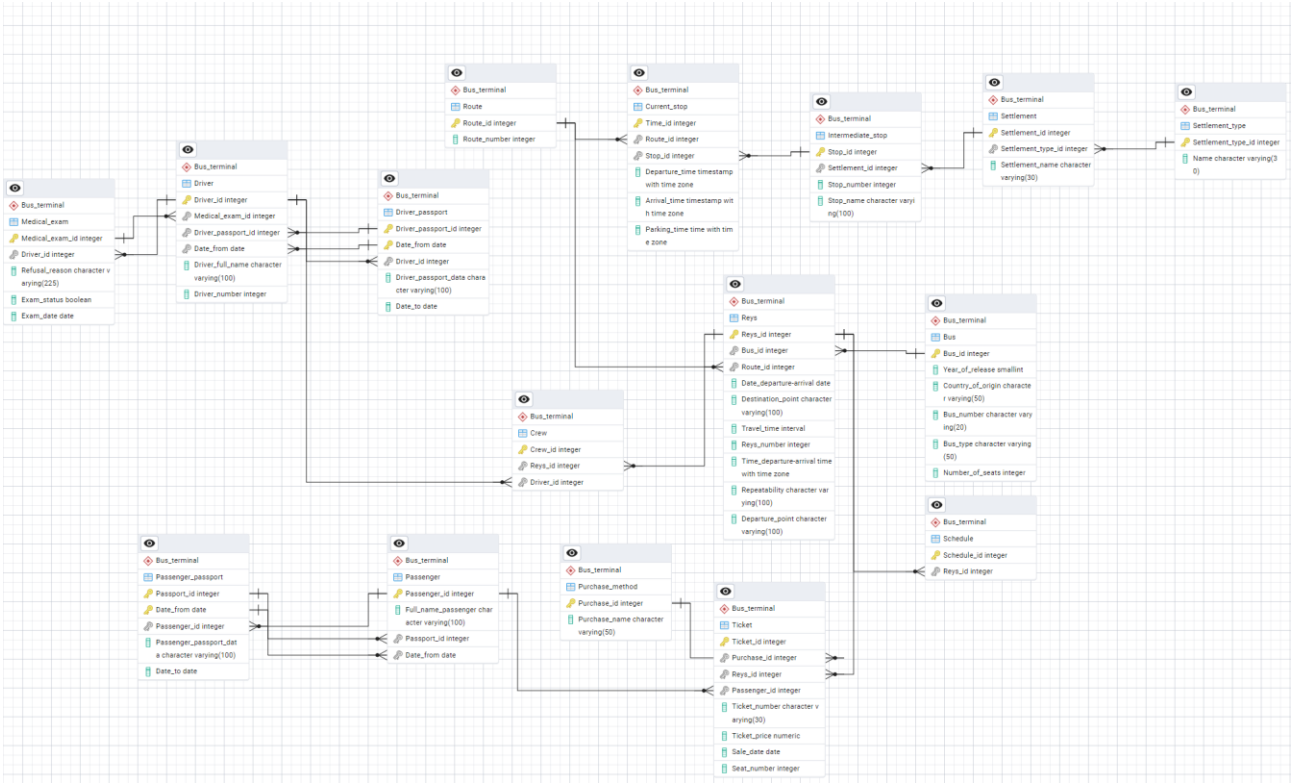
   Создать 3 процедуры для индивидуальной БД согласно варианту (часть 4 ЛР 2). Допустимо использование IN/OUT параметров. Допустимо создать авторские процедуры. (3 балла)

   Создать триггеры для индивидуальной БД согласно варианту:

   Вариант 2.1. 3 триггера - 3 балла (min). Допустимо использовать триггеры логирования из практического занятия по функциям и триггерам.

   Вариант 2.2. 7 оригинальных триггеров - 7 баллов (max).

4. **<u>Выполнение:</u>**
   1. Наименование создаваемой БД: **Bus_terminal**
   2. Схема логической модели базы данных, сгенерированная в pgadmin:



3. Процедуры:

Продажи билета:

```
CREATE OR REPLACE PROCEDURE "Bus_terminal"."Sell_Ticket"(
    IN p_reys_id INT,
    IN p_passenger_id INT,
    IN p_seat_number INT,
    IN p_ticket_price NUMERIC
)
LANGUAGE plpgsql
AS $$
BEGIN
    IF EXISTS (
        SELECT 1 FROM "Bus_terminal"."Ticket"
        WHERE "Reys_id" = p_reys_id AND "Seat_number" = p_seat_number
    ) THEN
        RAISE EXCEPTION 'Место % в рейсе % уже занято', p_seat_number, p_reys_id;
    END IF;
    INSERT INTO "Bus_terminal"."Ticket" (
        "Purchase_id", "Reys_id", "Passenger_id",
        "Ticket_number", "Ticket_price", "Sale_date", "Seat_number"
    )
    VALUES (
        1,
        p_reys_id,
        p_passenger_id,
        'AUTO_' || gen_random_uuid(),
        p_ticket_price,
        CURRENT_DATE,
        p_seat_number
    );
END;
$$;
```

До:

	Ticket_id [PK] integer	Purchase_id integer	Reys_id integer	Passenger_id integer	Ticket_number character varying (30)	Ticket_price numeric	Sale_date date	Seat_number integer
1	1	1	1	1	TICK001	1320.5500	2025-06-10	15
2	2	2	2	2	TICK002	800.00	2025-06-11	10
3	3	3	3	3	TICK003	950.75	2025-06-12	5
4	4	1	1	1	TICK004	1100.0000	2025-06-21	20
5	5	2	1	2	TICK005	1210.0000	2025-06-15	21
6	6	1	4	1	TICK006	990.0000	2025-06-22	12
7	7	2	1	2	TICK007	1210.0000	2025-01-15	25
8	8	1	3	3	TICK008	1000.00	2025-06-23	22

После:

	Ticket_id [PK] integer	Purchase_id integer	Reys_id integer	Passenger_id integer	Ticket_number character varying (50)	Ticket_price numeric	Sale_date date	Seat_number integer
2	2	2	2	2	TICK002	800.00	2025-06-11	10
3	3	3	3	3	TICK003	950.75	2025-06-12	5
4	4	1	1	1	TICK004	1100.0000	2025-06-21	20
5	5	2	1	2	TICK005	1210.0000	2025-06-15	21
6	6	1	4	1	TICK006	990.0000	2025-06-22	12
7	7	2	1	2	TICK007	1210.0000	2025-01-15	25
8	8	1	3	3	TICK008	1000.00	2025-06-23	22
9	11	1	1	2	TICKET_9a4e5937-7be5-4c4f-9f0	1450	2025-06-27	12

Возврат билета:

```
CREATE OR REPLACE PROCEDURE "Bus_terminal"."Return_Ticket"(
    IN p_ticket_id INT
)
LANGUAGE plpgsql
AS $$
BEGIN
    IF NOT EXISTS (
        SELECT 1 FROM "Bus_terminal"."Ticket" WHERE "Ticket_id" = p_ticket_id
    ) THEN
        RAISE EXCEPTION 'Билет с ID % не найден', p_ticket_id;
    END IF;

    DELETE FROM "Bus_terminal"."Ticket"
    WHERE "Ticket_id" = p_ticket_id;
END;
$$;
```

До:

	Ticket_id [PK] integer	Purchase_id integer	Reys_id integer	Passenger_id integer	Ticket_number character varying (50)	Ticket_price numeric	Sale_date date	Seat_number integer
1	1	1	1	1	TICK001	1320.5500	2025-06-10	15
2	2	2	2	2	TICK002	800.00	2025-06-11	10
3	3	3	3	3	TICK003	950.75	2025-06-12	5
4	4	1	1	1	TICK004	1100.0000	2025-06-21	20
5	5	2	1	2	TICK005	1210.0000	2025-06-15	21
6	6	1	4	1	TICK006	990.0000	2025-06-22	12
7	7	2	1	2	TICK007	1210.0000	2025-01-15	25
8	8	1	3	3	TICK008	1000.00	2025-06-23	22
9	11	1	1	2	TICKET_9a4e5937-7be5-4c4f-9f0	1450	2025-06-27	12

После:

	Ticket_id [PK] integer	Purchase_id integer	Reys_id integer	Passenger_id integer	Ticket_number character varying (50)	Ticket_price numeric	Sale_date date	Seat_number integer
1	2	2	2	2	TICK002	800.00	2025-06-11	10
2	3	3	3	3	TICK003	950.75	2025-06-12	5
3	4	1	1	1	TICK004	1100.0000	2025-06-21	20
4	5	2	1	2	TICK005	1210.0000	2025-06-15	21
5	6	1	4	1	TICK006	990.0000	2025-06-22	12
6	7	2	1	2	TICK007	1210.0000	2025-01-15	25
7	8	1	3	3	TICK008	1000.00	2025-06-23	22
8	11	1	1	2	TICKET_9a4e5937-7be5-4c4f-9f0	1450	2025-06-27	12

Добавление нового рейса:

```
CREATE OR REPLACE PROCEDURE "Bus_terminal"."Add_New_Reys"(
    IN p_route_id INT,
    IN p_bus_id INT,
    IN p_date DATE,
    IN p_time TIME,
    IN p_travel_time INTERVAL,
    IN p_departure_point TEXT,
    IN p_destination_point TEXT,
    IN p_reys_number INT,
    IN p_repeatability TEXT
)
LANGUAGE plpgsql
AS $$
BEGIN
    INSERT INTO "Bus_terminal"."Reys" (
        "Route_id", "Bus_id", "Date_departure", "Time_departure",
        "Travel_time", "Departure_point", "Destination_point",
        "Reys_number", "Repeatability", "Time_arrival", "Date_arrival"
    )
    VALUES (
        p_route_id,
        p_bus_id,
        p_date,
        p_time,
        p_travel_time,
        p_departure_point,
        p_destination_point,
        p_reys_number,
        p_repeatability,
        p_time + p_travel_time,
        (p_date + (p_time + p_travel_time)::time)::timestamp::date
    );
END;
$$;
```

До:

	Reys_id [PK] integer	Bus_id integer	Route_id integer	Date_departure date	Destination_point character varying (100)	Travel_time interval	Reys_number integer	Time_departure time with time zone	Repeatability character varying (100)	Departure_point character varying (100)
1	1	1	1	2025-06-20	Moscow	03:30:00	101	08:00:00+03:00	Daily	Saint_Petersburg
2	2	1	2	2025-06-21	Kazan	04:15:00	102	09:00:00+03:00	Weekly	Nizhny_Novgorod
3	3	1	3	2025-06-22	Sochi	05:00:00	103	07:30:00+03:00	Once	Rostov
4	4	1	1	2025-06-23	Kazan	04:00:00	104	10:00:00+03:00	Once	Moscow

После:

	Reys_id [PK] integer	Bus_id integer	Route_id integer	Date_departure date	Destination_point character varying (100)	Travel_time interval	Reys_number integer	Time_departure time with time zone	Repeatability character varying (100)	Departure_point character varying (100)
1	1	1	1	2025-06-20	Moscow	03:30:00	101	08:00:00+03:00	Daily	Saint_Petersburg
2	2	1	2	2025-06-21	Kazan	04:15:00	102	09:00:00+03:00	Weekly	Nizhny_Novgorod
3	3	1	3	2025-06-22	Sochi	05:00:00	103	07:30:00+03:00	Once	Rostov
4	4	1	1	2025-06-23	Kazan	04:00:00	104	10:00:00+03:00	Once	Moscow
5	6	1	1	2025-06-30	Казань	03:00:00	305	08:00:00+03:00	Один раз	Москва

4. Триггеры:

Логирование продажи билета:

```
CREATE OR REPLACE FUNCTION "Bus_terminal"."log_ticket_sale"()
RETURNS TRIGGER AS $$
BEGIN
   INSERT INTO "Bus_terminal"."Ticket_Log"("Ticket_id", "Action")
   VALUES (NEW."Ticket_id", 'Продажа');
   RETURN NEW;
END;
$$ LANGUAGE plpgsql;


CREATE TRIGGER "trigger_log_ticket_sale"
AFTER INSERT ON "Bus_terminal"."Ticket"
FOR EACH ROW
EXECUTE FUNCTION "Bus_terminal"."log_ticket_sale"();
```

Проверка:

9	12	1	1	2	TICKET_ee6532af-16a7-4173-851	1300	2025-06-27	10

	Log_id [PK] integer	Ticket_id integer	Sale_time timestamp without time zone	Action text
1	1	12	2025-06-27 00:54:57.764462	Продажа

Контроль удаления билета:

```
CREATE OR REPLACE FUNCTION "Bus_terminal"."prevent_ticket_delete_if_past"()
RETURNS TRIGGER AS $$
DECLARE
    reys_date DATE;
BEGIN
    SELECT "Date_departure"
    INTO reys_date
    FROM "Bus_terminal"."Reys"
    WHERE "Reys_id" = OLD."Reys_id";

    IF reys_date < CURRENT_DATE THEN
        RAISE EXCEPTION 'Нельзя удалить билет: рейс уже прошёл';
    END IF;

    RETURN OLD;
END;
$$ LANGUAGE plpgsql;



CREATE TRIGGER "trigger_prevent_ticket_delete_if_past"
BEFORE DELETE ON "Bus_terminal"."Ticket"
FOR EACH ROW
EXECUTE FUNCTION "Bus_terminal"."prevent_ticket_delete_if_past"();
```

Проверка:

Прошедшие рейсы:

	Ticket_id integer	Date_departure date
1	2	2025-06-21
2	3	2025-06-22
3	8	2025-06-22
4	4	2025-06-20
5	5	2025-06-20
6	7	2025-06-20
7	6	2025-06-23
8	11	2025-06-20
9	12	2025-06-20

Попытка удалить билет Ticket_id = 12:

```
ERROR:  Нельзя удалить билет: рейс уже прошёл
CONTEXT:  функция PL/pgSQL "Bus_terminal".prevent_ticket_delete_if_past(), строка 11, оператор RAISE

ОШИБКА:  Нельзя удалить билет: рейс уже прошёл
SQL state: P0001
```

Автоматическое обновление количества рейсов у водителя:

```
CREATE OR REPLACE FUNCTION "Bus_terminal"."increment_driver_reys"()
RETURNS TRIGGER AS $$
BEGIN
    UPDATE "Bus_terminal"."Driver"
    SET "Total_reys" = "Total_reys" + 1
    WHERE "Driver_id" = NEW."Driver_id";

    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER "trigger_increment_driver_reys"
AFTER INSERT ON "Bus_terminal"."Crew"
FOR EACH ROW
EXECUTE FUNCTION "Bus_terminal"."increment_driver_reys"();
```

Проверка:

Добавили Crew_id = 8:

	Crew_id [PK] integer	Reys_id integer	Driver_id integer
1	1	1	1
2	2	2	2
3	3	3	3
4	8	1	2

У водителя Driver_id = 2:

	Total_reys integer
1	1

Вывод: В ходе этой лабораторной работы я изучил и научился применять триггеры, процедуры и функции в СУБД PostgreSQL.