Министерство науки и высшего образования Российской Федерации
федеральное государственное автономное образовательное учреждение высшего образования
«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»

Отчет

по лабораторной работе «Создание таблиц базы данных PostgreSQL. Заполнение таблиц рабочими данными»

по дисциплине «**Базы данных**»

Автор: Акулов Алексей

Факультет: ФИКТ

Группа: К32391

Преподаватель: Говорова М.М.

Санкт-Петербург 2022

Цель работы: овладеть практическими навыками создания таблиц базы данных PostgreSQL 1X, заполнения их рабочими данными, резервного копирования и восстановления БД.

Оборудование: компьютерный класс.

Программное обеспечение: СУБД PostgreSQL 1X, pgAdmin 4.

Практическое задание:

1. Создать запросы и представления на выборку данных к базе данных PostgreSQL (согласно индивидуальному заданию, часть 2 и 3).
2. Составить 3 запроса на модификацию данных (INSERT, UPDATE, DELETE) с использованием подзапросов.
3. Изучить графическое представление запросов и просмотреть историю запросов.
4. Создать простой и составной индексы для двух произвольных запросов и сравнить время выполнения запросов без индексов и с индексами. Для получения плана запроса использовать команду EXPLAIN.

Вариант 6. БД «Пассажир»

Описание предметной области: Информационная система служит для продажи железнодорожных билетов. Билеты могут продаваться на текущие сутки или предварительно (не более чем за 45 суток). Цена билета при предварительной продаже снижается на 5%. Билет может быть приобретен в кассе или онлайн. Если билет приобретен в кассе, необходимо знать, в какой. Для каждой кассы известны номер и адрес. Кассы могут располагаться в различных населенных пунктах.

Поезда курсируют по расписанию, но могут назначаться дополнительные поезда на заданный период или определенные даты.

По всем промежуточным остановкам на маршруте известны название, тип населенного пункта, время прибытия, отправления, время стоянки.

БД должна содержать следующий минимальный набор сведений: Номер поезда. Название поезда. Тип поезда. Пункт назначения. Пункт назначения для проданного билета. Номер вагона. Тип вагона. Количество мест в вагоне. Цена билета. Дата отправления. Дата прибытия. Дата прибытия для пункта назначения проданного билета. Время отправления. Номер вагона в поезде. Номер билета. Место. Тип места. Фамилия пассажира. Имя пассажира. Отчество пассажира. Паспортные данные.

I. Название модели – «База пассажиров»
II. Состав реквизитов сущностей
    a. Пассажир (Паспортные данные, Фамилия, Имя, Отчество, Телефон, Электронная почта)

b. Билет (Номер, Пассажир, Пункт назначения, Пункт отправления, Место, Цена, Статус оплаты, Статус возврата, В кассе или онлайн)
c. Место (Номер места, Вагон, Статус занятости)
d. Вагон (Номер вагона, Номер в поезде, Поезд, Количество мест, Тип)
e. Поезд (Номер поезда, Расписание, Дата отправления, Дата прибытия, Выполнение, Название)
f. Расписание поездов (Номер маршрута, Время отправления, Время прибытия, Тип, Пункт отправления, Пункт прибытия, Периодичность)
g. Остановка (Номер, Название, Тип, Маршрут)
h. Остановка поезда (Время прибытия, Время стоянки, Время отправления)
i. Касса (Номер кассы, Адрес, Населенный пункт)

Задание 2. Создать запросы:

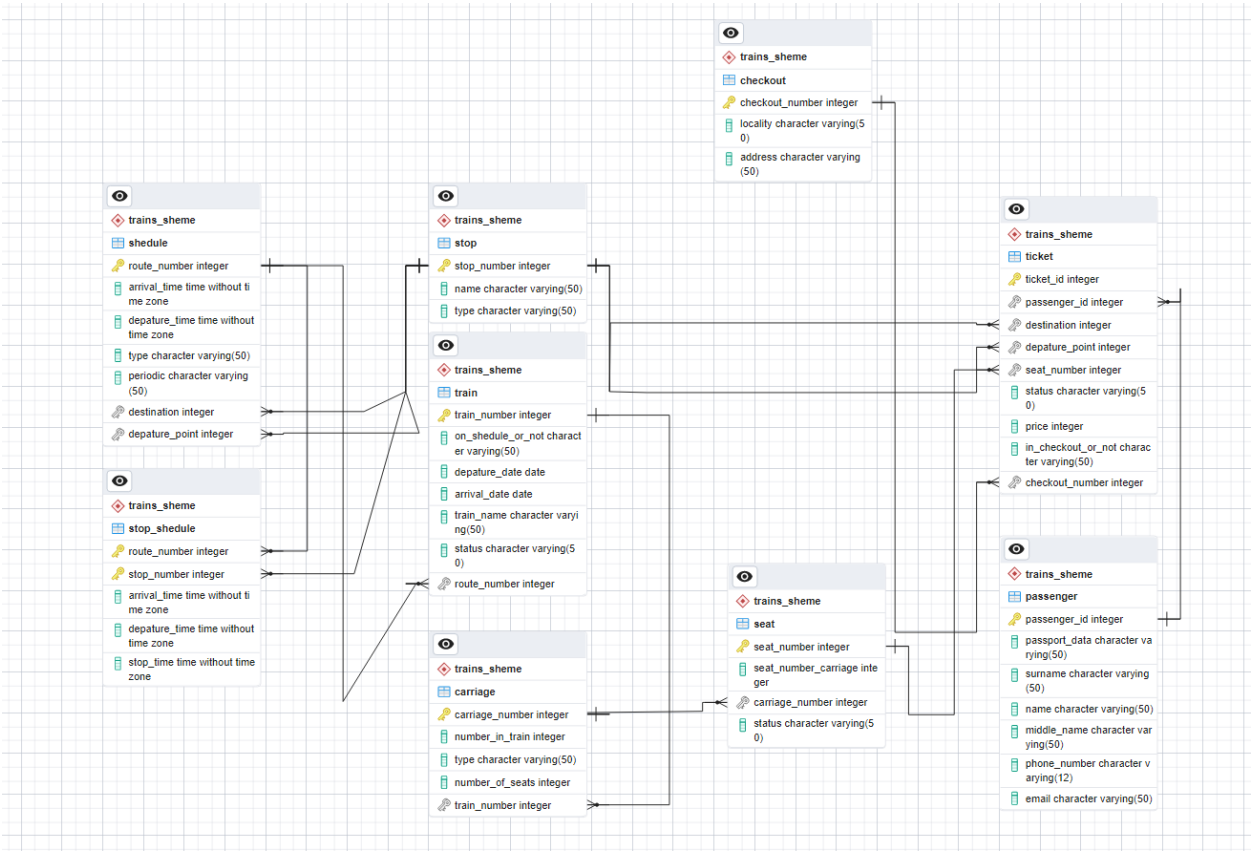
a) Свободные места на все поезда, отправляющиеся с вокзала в течение следующих суток.
b) Список пассажиров, отправившихся в Москву всеми рейсами за прошедшие сутки.
c) Номера поездов, на которые проданы все билеты на следующие сутки.
d) Свободные места в купейные вагоны всех рейсов до Москвы на текущие сутки.
e) Выручка от продажи билетов на все поезда за прошедшие сутки.
f) Общее количество билетов, проданных по всем направлениям в вагоны типа “СВ”.
g) Номера и названия поездов, все вагоны которых были заполнены менее чем наполовину за прошедшие сутки.

Задание 3. Создать представление:

- для пассажиров о наличии свободных мест на заданный рейс;
- количество непроданных билетов на все поезда, формирующиеся за прошедшие сутки (номер поезда, тип вагона, количество).

Выполнение:

1. После перенесения всех таблиц из прошлого задания с помощью скрипта в SQL получается схема базы данных



2. Дальше делаем задания 2 и 3. Скрипт для запросов будет приложен отдельно. Здесь прикрепляю примеры ответов на последние два запроса

```sql
WITH trains_next_24h AS (
    SELECT t.train_number
    FROM "trains_sheme"."train" t
    JOIN "trains_sheme"."shedule" s ON t.route_number = s.route_number
    WHERE t.depature_date >= CURRENT_DATE
        AND t.depature_date <= CURRENT_DATE + INTERVAL '1 day'
        AND s.depature_point = (SELECT stop_number FROM "trains_sheme"."stop" WHERE name = 'S
)

SELECT
    t.train_number,
    c.carriage_number,
    s.seat_number,
    s.seat_number_carriage
FROM "trains_sheme"."seat" s
JOIN "trains_sheme"."carriage" c ON s.carriage_number = c.carriage_number
JOIN trains_next_24h t ON c.train_number = t.train_number
WHERE s.status = 'free'
ORDER BY t.train_number, c.carriage_number, s.seat_number;
```

Data Output | Messages | Notifications

	train_number integer	carriage_number integer	seat_number integer	seat_number_carriage integer
1	101	103	11	1
2	101	103	12	2
3	104	104	13	3

Total rows: 3 of 3 | Query complete 00:00:00.101

a.

```sql
WITH moscow_trains AS (
    SELECT t.train_number
    FROM "trains_sheme"."train" t
    JOIN "trains_sheme"."shedule" s ON t.route_number = s.route_number
    WHERE t.depature_date >= CURRENT_DATE - INTERVAL '1 day'
        AND t.depature_date < CURRENT_DATE
        AND s.destination = (SELECT stop_number FROM "trains_sheme"."stop" WHERE name = 'MSC')
)

SELECT DISTINCT
    p.passenger_id,
    p.surname,
    p.name,
    p.middle_name
FROM "trains_sheme"."ticket" tk
JOIN "trains_sheme"."passenger" p ON tk.passenger_id = p.passenger_id
JOIN moscow_trains t ON tk.destination = (SELECT stop_number FROM "trains_sheme"."stop" WHERE name = 'MSC'
WHERE tk.status = 'paid'
ORDER BY p.surname, p.name, p.middle_name;
```

Data Output | Messages | Notifications

	passenger_id [PK] integer	surname character varying (50)	name character varying (50)	middle_name character varying (50)
1	1	Ivanov	Ivan	Ivanovich
2	4	Smirnov	Alexey	Alexeevich
3	7	Vasiliev	Vasiliy	Vasilievich

b.

c. **F**

```sql
WITH moscow_trains_today AS (
    SELECT t.train_number
    FROM "trains_sheme"."train" t
    JOIN "trains_sheme"."shedule" s ON t.route_number = s.route_number
    WHERE t.depature_date = CURRENT_DATE
        AND s.destination = (SELECT stop_number FROM "trains_sheme"."stop" WHERE name = 'M
)

SELECT
    t.train_number,
    c.carriage_number,
    s.seat_number,
    s.seat_number_carriage
FROM "trains_sheme"."seat" s
JOIN "trains_sheme"."carriage" c ON s.carriage_number = c.carriage_number
JOIN moscow_trains_today t ON c.train_number = t.train_number
WHERE s.status = 'free' AND c.type = 'coupe'
ORDER BY t.train_number, c.carriage_number, s.seat_number;
```

Data Output | Messages | Notifications

	train_number integer	carriage_number integer	seat_number integer	seat_number_carriage integer
1	107	107	104	1
2	107	107	105	5
3	108	108	107	6

Total rows: 3 of 3 | Query complete 00:00:00.073

d.

```sql
SELECT
    SUM(ti.price) AS total_revenue
FROM
    trains_sheme.ticket ti
    JOIN trains_sheme.train t ON ti.train_number = t.train_number
WHERE
    ti.status = 'paid'
    AND t.depature_date >= CURRENT_DATE - INTERVAL '1 day'
    AND t.depature_date < CURRENT_DATE;
```

Data Output | Messages | Notifications

	total_revenue bigint
1	2500

e.

```sql
SELECT COUNT(tk.ticket_id) AS total_sv_tickets_sold
FROM "trains_sheme"."ticket" tk
JOIN "trains_sheme"."seat" s ON tk.seat_number = s.seat_number
JOIN "trains_sheme"."carriage" c ON s.carriage_number = c.carriage_nu
WHERE c.type = 'SV' AND tk.status = 'paid';
```

f.

	total_sv_tickets_sold bigint
1	1

g.

```sql
WITH trains_past_24h AS (
    SELECT t.train_number, t.train_name
    FROM "trains_sheme"."train" t
    WHERE t.depature_date >= CURRENT_DATE - INTERVAL '1 day'
        AND t.depature_date < CURRENT_DATE
),
carriage_capacity AS (
    SELECT
        c.train_number,
        c.carriage_number,
        COUNT(s.seat_number) AS total_seats
    FROM "trains_sheme"."carriage" c
    JOIN "trains_sheme"."seat" s ON c.carriage_number = s.carriage_number
    GROUP BY c.train_number, c.carriage_number
),
carriage_sold_tickets AS (
    SELECT
        t.train_number,
        c.carriage_number,
        COUNT(tk.ticket_id) AS sold_tickets_count
    FROM "trains_sheme"."ticket" tk
    JOIN "trains_sheme"."seat" s ON tk.seat_number = s.seat_number
    JOIN "trains_sheme"."carriage" c ON s.carriage_number = c.carriage_number
    JOIN trains_past_24h t ON c.train_number = t.train_number
    WHERE tk.status = 'paid'
    GROUP BY t.train_number, c.carriage_number
),
carriage_stats AS (
    SELECT
        cc.train_number,
        cc.carriage_number,
        cc.total_seats,
        COALESCE(cst.sold_tickets_count, 0) AS sold_tickets_count
    FROM carriage_capacity cc
    LEFT JOIN carriage_sold_tickets cst ON cc.train_number = cst.train_number AND cc.carriage_number = cst.carriage
)

SELECT
    tp.train_number,
    tp.train_name
FROM trains_past_24h tp
WHERE NOT EXISTS (
    SELECT 1
    FROM carriage_stats cs
    WHERE cs.train_number = tp.train_number AND cs.sold_tickets_count >= (cs.total_seats / 2)
)
ORDER BY tp.train_number;
```

	train_number [PK] integer	train_name character varying (50)
1	7	Train 7
2	9	Train 2

3. Создаем виды

```sql
CREATE VIEW trains_sheme.free_seats_on_train AS
    SELECT
        t.train_number,
        c.number_in_train AS carriage_number,
        s.seat_number_carriage AS seat_number_in_carriage,
        c.type AS carriage_type,
        s.status AS seat_status
    FROM
        trains_sheme.train t
        JOIN trains_sheme.carriage c ON t.train_number = c.train_number
        JOIN trains_sheme.seat s ON c.carriage_number = s.carriage_number
    WHERE
        s.status = 'free';
```

```sql
CREATE VIEW trains_sheme.unsold_tickets_per_train_and_carriage_type AS
    SELECT
        t.train_number,
        c.type AS carriage_type,
        COUNT(s.seat_number) AS unsold_tickets_count
    FROM
        trains_sheme.train t
        JOIN trains_sheme.carriage c ON t.train_number = c.train_number
        JOIN trains_sheme.seat s ON c.carriage_number = s.carriage_number
    WHERE
        s.status != 'occupied'
        AND t.depature_date >= CURRENT_DATE - INTERVAL '1 day'
        AND t.depature_date < CURRENT_DATE
    GROUP BY
        t.train_number, c.type;
```

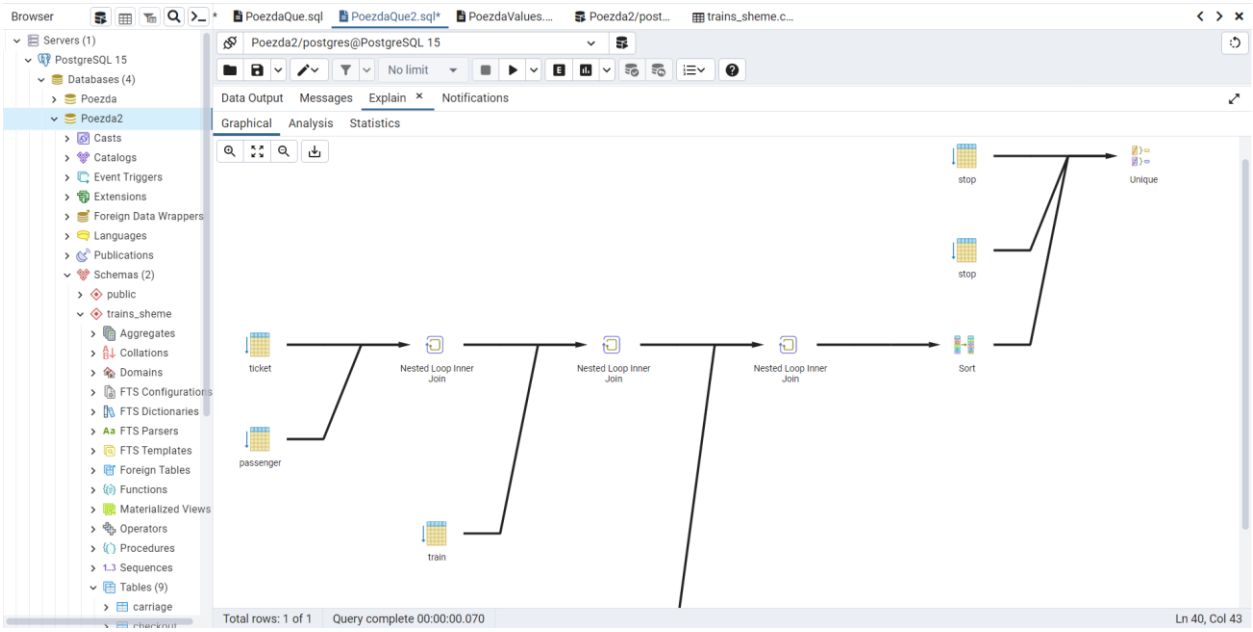
4. Также делаем три запроса с подзапросами, их скрипт также будет приложен к отчету

```sql
INSERT INTO "trains_sheme"."ticket"("passenger_id", "train_number", "destination", "depature_point", "seat_number", "s
SELECT 1, 4, "stop_number", 1, 7, 'paid', 2000, 'yes', 1
FROM "trains_sheme"."stop"
WHERE "name" = 'MSC';

UPDATE "trains_sheme"."ticket"
SET "status" = 'changed', "destination" = (
    SELECT "stop_number"
    FROM "trains_sheme"."stop"
    WHERE "name" = 'SPB'
)
WHERE "ticket_id" = (
    SELECT MIN("ticket_id")
    FROM "trains_sheme"."ticket"
    WHERE "status" = 'paid' AND "destination" = (
        SELECT "stop_number"
        FROM "trains_sheme"."stop"
        WHERE "name" = 'MSC'
    )
);
```

```
21
22 INSERT INTO "trains_sheme"."checkout"("checkout_number", "locality", "address")
23 VALUES (9, 'City1', 'Address1');
24
25 DELETE FROM "trains_sheme"."checkout"
26 WHERE "checkout_number" = (
27     SELECT MAX("checkout_number")
28     FROM "trains_sheme"."checkout"
29     WHERE "locality" = 'City1'
30 );
31
32
```

5. Изучаем графическое представление запроса и смотрим историю



6. После этого проводим измерения с индексом и без. Файлы с результатами также прикрепляю отдельным файлом

Вывод:

PGadmin позволяет хорошо работать с запросами всех типов. Также при формировании индекса ускоряется время работы