

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

Отчет

по лабораторной работе №4 «Запросы на выборку и модификацию данных.
Представления. Работа с индексами»

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

Автор: Волжева М. И.

Факультет: ИКТ

Группа: К3141

Преподаватель: Говорова М.М.



Санкт-Петербург 2023

Оглавление

Цель работы.....	3
Практическое задание	3
Вариант 19. БД «Пассажир»	3
Выполнение	4
Вывод	14

Цель работы

Овладеть практическими навыками создания представлений и запросов на выборку данных к базе данных PostgreSQL, использования подзапросов при модификации данных и индексов.

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 19. 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.

Необходимо учитывать, что местом посадки и высадки пассажира могут быть промежуточные пункты по маршруту.

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.

Задание 2. Создать запросы:

- 1) Свободные места на все поезда, отправляющиеся с вокзала в течение следующих суток.
- 2) Список пассажиров, отправившихся в Москву всеми рейсами за прошедшие сутки.
- 3) Номера поездов, на которые проданы все билеты на следующие сутки.
- 4) Свободные места в купейные вагоны всех рейсов до Москвы на текущие сутки.
- 5) Выручка от продажи билетов на все поезда за прошедшие сутки.
- 6) Общее количество билетов, проданных по всем направлениям в вагоны типа "СВ".
- 7) Номера и названия поездов, все вагоны которых были заполнены менее чем наполовину за прошедшие сутки.

Задание 3. Создать представление:

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

Выполнение

Запросы к базе данных:

- 1) Свободные места на все поезда, отправляющиеся с вокзала в течение следующих суток.

```

select
  seats.seat_number,
  scheduled_train_carriages.carriage_order_number,
  timetable.departure_time,
  stations.station_name
from
  railways.seats,
  railways.scheduled_train_carriages,
  railways.timetable,
  railways.scheduled_trains,
  railways.trains,
  railways.train_stations
where
  seats.is_empty
  and seats.scheduled_train_carriage_id = scheduled_train_carriages.scheduled_train_carriage_id
  and scheduled_train_carriages.scheduled_train_id = scheduled_trains.scheduled_train_id
  and scheduled_trains.scheduled_train_id = timetable.scheduled_train_id
  and timetable.train_id = trains.train_id
  and trains.train_id = train_stations.train_id
  and train_stations.station_id = stations.station_id
  and train_stations.order_number = 1
  and timetable.departure_time >= date_trunc('day', current_date + interval '2 days')
  and timetable.departure_time < date_trunc('day', current_date + interval '3 days')

```

seat_number	carriage_order_number	departure_time	station_name
1	1	1 2023-11-27 04:05:06.000000	Москва
2	2	1 2023-11-27 04:05:06.000000	Москва
3	3	1 2023-11-27 04:05:06.000000	Москва
4	4	1 2023-11-27 04:05:06.000000	Москва
5	5	1 2023-11-27 04:05:06.000000	Москва
6	6	1 2023-11-27 04:05:06.000000	Москва
7	7	1 2023-11-27 04:05:06.000000	Москва

```

select
  seats.seat_number,

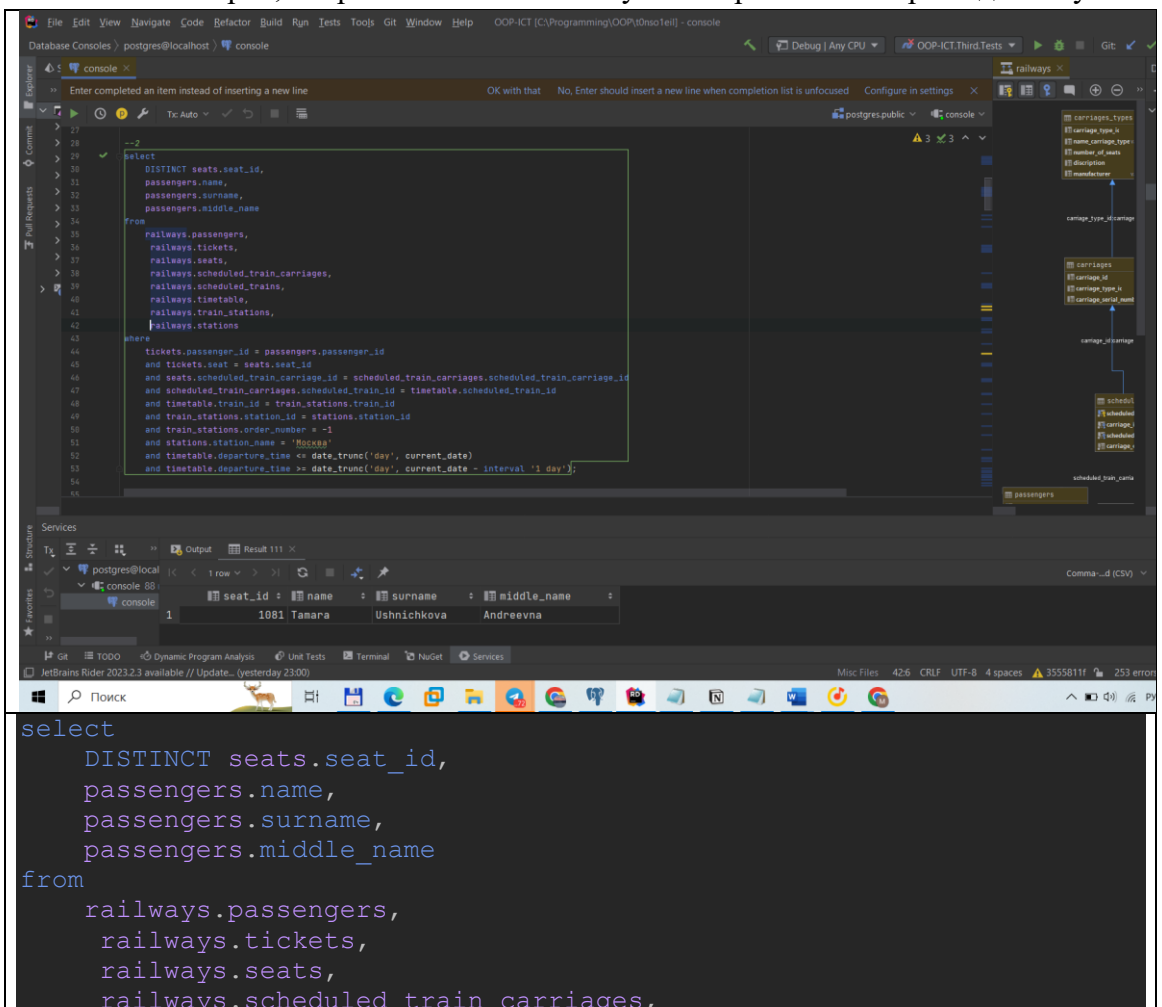
```

```

scheduled_train_carriages.carriage_order_number,
timetable.departure_time,
stations.station_name
from
railways.seats,
railways.scheduled_train_carriages,
railways.timetable,
railways.stations,
railways.scheduled_trains,
railways.trains,
railways.train_stations
WHERE
seats.is_empty
and seats.scheduled_train_carriage_id =
scheduled_train_carriages.scheduled_train_carriage_id
and scheduled_train_carriages.scheduled_train_id =
scheduled_trains.scheduled_train_id
and scheduled_trains.scheduled_train_id =
timetable.scheduled_train_id
and timetable.train_id = trains.train_id
and trains.train_id = train_stations.train_id
and train_stations.station_id = stations.station_id
and train_stations.order_number = '-1'
and timetable.departure_time <= date_trunc('day', current_date +
interval '2 days')
and timetable.departure_time >= date_trunc('day', current_date +
interval '1 day')

```

2) Список пассажиров, отправившихся в Москву всеми рейсами за прошедшие сутки.



The screenshot shows an IDE with a PostgreSQL query editor and a results pane. The query is as follows:

```

select
DISTINCT seats.seat_id,
passengers.name,
passengers.surname,
passengers.middle_name
from
railways.passengers,
railways.tickets,
railways.seats,
railways.scheduled_train_carriages,
railways.scheduled_trains,
railways.timetable,
railways.train_stations,
railways.stations
where
tickets.passenger_id = passengers.passenger_id
and tickets.seat = seats.seat_id
and seats.scheduled_train_carriage_id = scheduled_train_carriages.scheduled_train_carriage_id
and scheduled_train_carriages.scheduled_train_id = timetable.scheduled_train_id
and timetable.train_id = train_stations.train_id
and train_stations.station_id = stations.station_id
and train_stations.order_number = -1
and stations.station_name = 'Москва'
and timetable.departure_time <= date_trunc('day', current_date)
and timetable.departure_time >= date_trunc('day', current_date - interval '1 day');

```

The results pane shows a single row of data:

seat_id	name	surname	middle_name
1081	Tamara	Ushnickova	Andreevna

The bottom part of the image shows the same query text again:

```

select
DISTINCT seats.seat_id,
passengers.name,
passengers.surname,
passengers.middle_name
from
railways.passengers,
railways.tickets,
railways.seats,
railways.scheduled_train_carriages,

```

```

railways.scheduled_trains,
railways.timetable,
railways.train_stations,
railways.stations
where
    tickets.passenger_id = passengers.passenger_id
    and tickets.seat = seats.seat_id
    and seats.scheduled_train_carriage_id =
scheduled_train_carriages.scheduled_train_carriage_id
    and scheduled_train_carriages.scheduled_train_id =
timetable.scheduled_train_id
    and timetable.train_id = train_stations.train_id
    and train_stations.station_id = stations.station_id
    and train_stations.order_number = -1
    and stations.station_name = 'Москва'
    and timetable.departure_time <= date_trunc('day', current_date)
    and timetable.departure_time >= date_trunc('day', current_date -
interval '1 day');
```

3) Номера поездов, на которые проданы все билеты на следующие сутки.

The screenshot shows the PostgreSQL IDE (DataGrip) interface. The main editor displays a SQL query that identifies train numbers without seats. The query uses a Common Table Expression (CTE) named `trains_without_seats` to find train numbers where no seats exist for a given scheduled train carriage ID. The final query selects distinct train numbers from the `trains_without_seats` CTE, joined with the `trains` and `timetable` tables.

```

54 --
55 --
56 --
57 with trains_without_seats
58 as (
59   select distinct a.scheduled_train_carriage_id from railways.seats a
60   where
61     not exists (
62       select * from railways.seats b where b.is_empty and b.scheduled_train_carriage_id = a.scheduled_train_carriage_id
63     )
64 )
65 --
66 select
67   distinct trains.train_number
68 from
69   trains_without_seats, railways.trains, railways.timetable
70 where
71   trains_without_seats.scheduled_train_carriage_id = timetable.scheduled_train_id
72   and timetable.train_id = trains.train_id
73   and timetable.departure_time <= date_trunc('day', current_date + interval '2 days')
74   and timetable.departure_time >= date_trunc('day', current_date + interval '1 day');
75 --

```

The right sidebar shows the database schema for the `railways` database, including tables like `carriages_types`, `carriages`, and `schedules`.

The bottom status bar indicates that 0 rows were retrieved in 38 ms (execution: 8 ms, fetching: 30 ms).

```
with trains_without_seats
as (
    select distinct a.scheduled_train_carriage_id from railways.seats
a
    where
        not exists (
            select * from railways.seats b where b.is_empty and
b.scheduled_train_carriage_id = a.scheduled_train_carriage_id
        )
)

select
    distinct trains.train_number
from
    trains_without_seats, railways.trains, railways.timetable
where
    trains_without_seats.scheduled_train_carriage_id =
timetable.scheduled_train_id
    and timetable.train id = trains.train id
```

```

    and timetable.departure_time <= date_trunc('day', current_date +
interval '2 days')
    and timetable.departure_time >= date_trunc('day', current_date +
interval '1 day');

```

4) Свободные места в купейные вагоны всех рейсов до Москвы на текущие сутки.

The screenshot shows a database console interface with a SQL query and its results. The query is as follows:

```

select distinct train_name, scheduled_train_carriages.carriage_order_number, seat_number
from
    railways.timetable,
    railways.scheduled_train_carriages,
    railways.carriages,
    railways.seats,
    railways.trains
where
    timetable.departure_time <= date_trunc('day', current_date + interval '1 day')
    and timetable.departure_time >= date_trunc('day', current_date)
    and timetable.scheduled_train_id = scheduled_train_carriages.scheduled_train_id
    and scheduled_train_carriages.carriage_id = carriages.carriage_id
    and carriage_type_id = 1
    and seats.scheduled_train_carriage_id = timetable.scheduled_train_id
    and seats.is_empty
    and timetable.train_id = trains.train_id;

```

The results table shows 5 rows of data:

train_name	carriage_order_number	seat_number
Sapsan (Spb - Moscow)	1	12
Sapsan (Spb - Moscow)	1	15
Sapsan (Spb - Moscow)	1	18
Sapsan (Spb - Moscow)	1	4
Sapsan (Spb - Moscow)	1	3

```

select distinct train_name,
scheduled_train_carriages.carriage_order_number, seat_number
from
    railways.timetable,
    railways.scheduled_train_carriages,
    railways.carriages,
    railways.seats,
    railways.trains
where
    timetable.departure_time <= date_trunc('day', current_date +
interval '1 day')
    and timetable.departure_time >= date_trunc('day', current_date)
    and timetable.scheduled_train_id =
scheduled_train_carriages.scheduled_train_id
    and scheduled_train_carriages.carriage_id = carriages.carriage_id
    and carriage_type_id = 1
    and seats.scheduled_train_carriage_id =
timetable.scheduled_train_id
    and seats.is_empty
    and timetable.train_id = trains.train_id;

```

5) Выручка от продажи билетов на все поезда за прошедшие сутки.

```

--5
select sum(seats.price)
from railways.seats, railways.tickets
where
    tickets.buying_time <= date_trunc('day', current_date)
    and tickets.buying_time >= date_trunc('day', current_date - interval '1 day')
    and tickets.seat = seats.seat_id;

```

Services

Output: sum(seats.price):numeric

sum
1185020.00

```

select sum(seats.price)
from railways.seats, railways.tickets
where
    tickets.buying_time <= date_trunc('day', current_date)
    and tickets.buying_time >= date_trunc('day', current_date -
interval '1 day')
    and tickets.seat = seats.seat_id;

```

6) Общее количество билетов, проданных по всем направлениям в вагоны типа “СВ”.

```

--6
select count(DISTINCT tickets.ticket_id)
from railways.tickets,
    railways.seats,
    railways.scheduled_train_carriages,
    railways.carriages,
    railways.carriages_types
where
    tickets.seat = seats.seat_id
    and seats.scheduled_train_carriage_id = scheduled_train_carriages.scheduled_train_carriage_id
    and scheduled_train_carriages.carriage_id = carriages.carriage_id
    and carriages.carriage_type_id = carriages_types.carriage_type_id
    and carriages_types.name_carriage_type = 'business_class';
--7

```

Services

Output: count(DISTINCT tickets.ticket_id):bigint

count
1

```

select count(DISTINCT tickets.ticket_id)
from railways.tickets,
    railways.seats,
    railways.scheduled_train_carriages,
    railways.carriages,
    railways.carriages_types
where

```

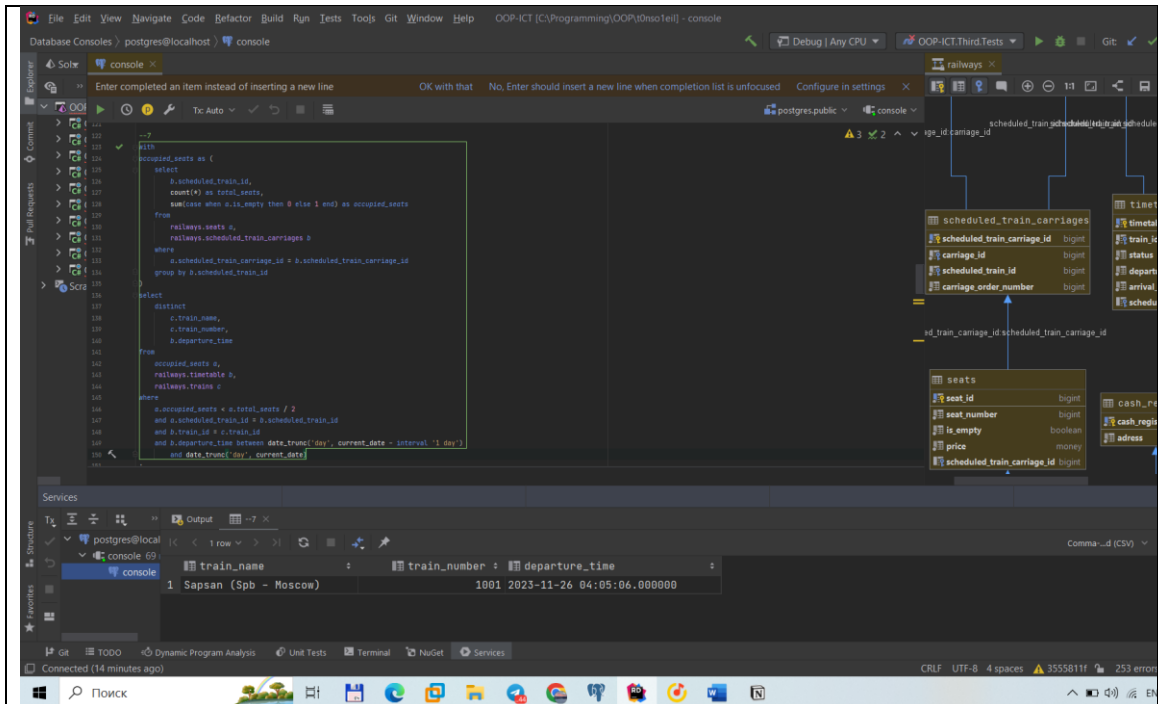


```

tickets.seat = seats.seat_id
and seats.scheduled_train_carriage_id =
scheduled_train_carriages.scheduled_train_carriage_id
and scheduled_train_carriages.carriage_id = carriages.carriage_id
and carriages.carriage_type_id = carriages_types.carriage_type_id
and carriages_types.name_carriage_type = 'business_class';

```

- 7) Номера и названия поездов, все вагоны которых были заполнены менее чем наполовину за прошедшие сутки.



```

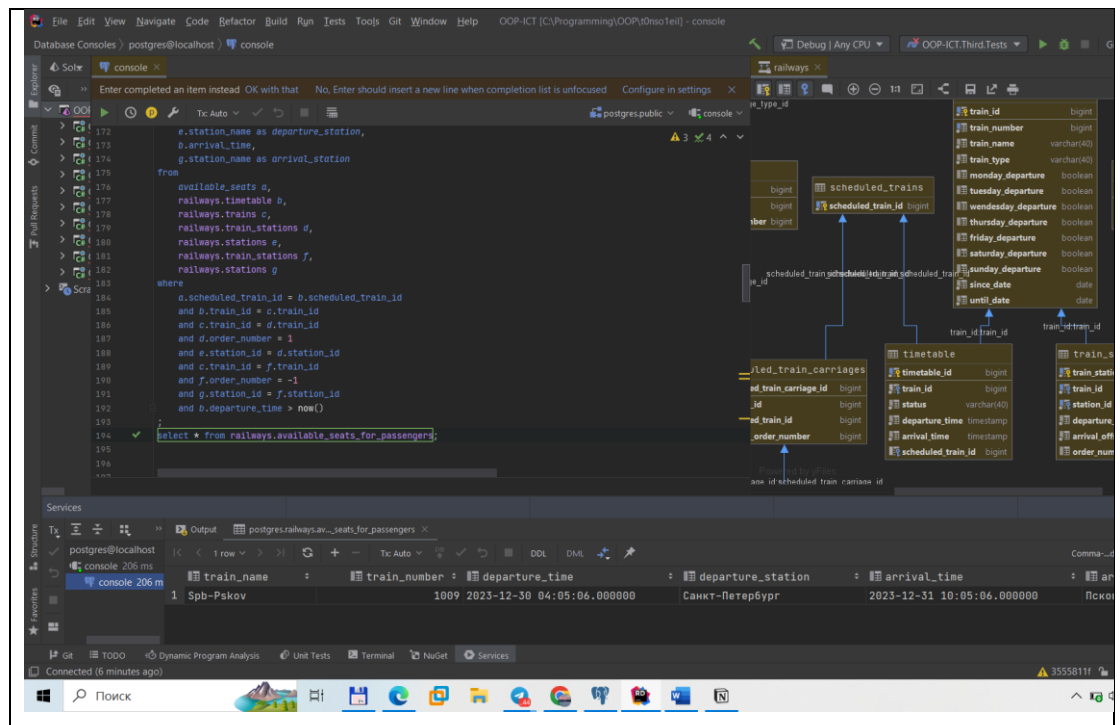
with
occupied_seats as (
    select
        b.scheduled_train_id,
        count(*) as total_seats,
        sum(case when a.is_empty then 0 else 1 end) as occupied_seats
    from
        railway.seats a,
        railway.scheduled_train_carriages b
    where
        a.scheduled_train_carriage_id = b.scheduled_train_carriage_id
    group by b.scheduled_train_id
)
select
    distinct
        c.train_name,
        c.train_number,
        b.departure_time
from
    occupied_seats a,
    railway.timetable b,
    railway.trains c
where
    a.occupied_seats < a.total_seats / 2
    and a.scheduled_train_id = b.scheduled_train_id
    and b.train_id = c.train_id
    and b.departure_time between date_trunc('day', current_date -
interval '1 day')
    and date_trunc('day', current_date)
;

```

Создать представление:

- 1) для пассажиров о наличии свободных мест на заданный рейс (номер поезда, название поезда, время отправления, станция отправления, станция прибытия, количество мест в вагоне);

```
create or replace view railways.available_seats_for_passengers as
with
    available_seats as (
        select
            b.scheduled_train_id,
            count(*) as total_seats,
            sum(case when a.is_empty then 0 else 1 end) as
available_seats
        from
            railways.seats a,
            railways.scheduled_train_carriages b
        where
            a.scheduled_train_carriage_id =
b.scheduled_train_carriage_id
        group by b.scheduled_train_id
    )
select
    c.train_name,
    c.train_number,
    b.departure_time,
    e.station_name as departure_station,
    b.arrival_time,
    g.station_name as arrival_station
from
    available_seats a,
    railways.timetable b,
    railways.trains c,
    railways.train_stations d,
    railways.stations e,
    railways.train_stations f,
    railways.stations g
where
    a.scheduled_train_id = b.scheduled_train_id
and b.train_id = c.train_id
and c.train_id = d.train_id
and d.order_number = 1
and e.station_id = d.station_id
and c.train_id = f.train_id
and f.order_number = -1
and g.station_id = f.station_id
and b.departure_time > now()
;
select * from railways.available_seats_for_passengers;
```



- 2) количество непроданных билетов на все поезда, формирующиеся за прошедшие сутки (номер поезда, тип вагона, количество).

```

create or replace view railways.available_seats_on_yesterday as
with
    available_seats as (
        select
            b.scheduled_train_id,
            b.scheduled_train_carriage_id,
            sum(case when a.is_empty then 1 else 0 end) as
available_seats
        from
            railways.seats a,
            railways.scheduled_train_carriages b
        where
            a.scheduled_train_carriage_id =
b.scheduled_train_carriage_id
        group by b.scheduled_train_id,
b.scheduled_train_carriage_id
    )
select
    c.train_name,
    c.train_number,
    b.departure_time,
    e.name_carriage_type,
    sum(a.available_seats) as available_seats
from
    available_seats a,
    railways.timetable b,
    railways.trains c,
    railways.carriages d,
    railways.carriages_types e,
    railways.scheduled_train_carriages f
where
    a.scheduled_train_id = b.scheduled_train_id
    and b.train_id = c.train_id
    and b.departure_time between date_trunc('day', current_date -

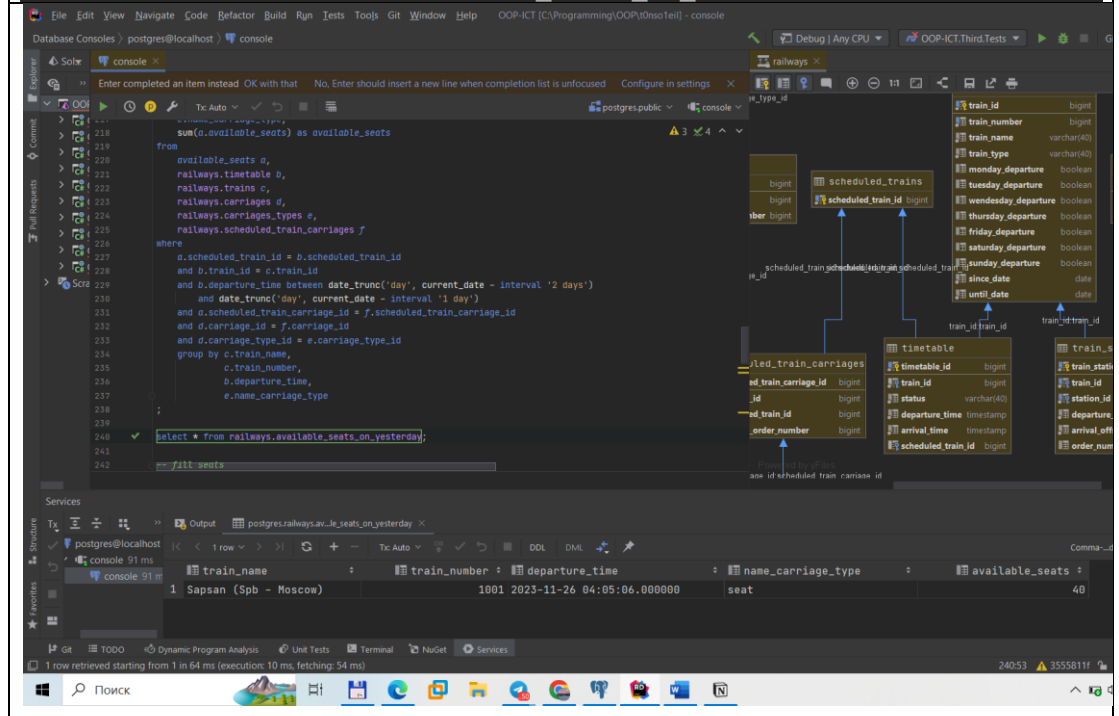
```

```

interval '2 days')
    and date_trunc('day', current_date - interval '1 day')
    and a.scheduled_train_carriage_id =
f.scheduled_train_carriage_id
    and d.carriage_id = f.carriage_id
    and d.carriage_type_id = e.carriage_type_id
group by c.train_name,
         c.train_number,
         b.departure_time,
         e.name_carriage_type
;

select * from railways.available_seats_on_yesterday;

```



Запрос на модификацию данных (INSERT, UPDATE, DELETE) с использованием подзапросов.

```

-- удаляем места, если поезд отменён
delete from railways.seats where seats.scheduled_train_carriage_id in
(
    select
        scheduled_train_carriages.scheduled_train_carriage_id
    from
        railways.timetable,
        railways.scheduled_train_carriages
    where
        timetable.scheduled_train_id =
scheduled_train_carriages.scheduled_train_id
        and timetable.status = 'canceled'
);

--обновляем статус is_empty если билет возвращён
UPDATE railways.seats
SET is_empty = true
WHERE seat_id in(
    select seats.seat_id
    from
        railways.seats,
        railways.tickets
    where

```

```

        seats.seat_id = tickets.seat
        and tickets.status = 'returned'
    );

--добавляем места, если их в вагоне заведено меньше,
delete from railways.seats where scheduled_train_carriage_id = 1;

insert into railways.seats(seat_id, seat_number, is_empty, price,
scheduled_train_carriage_id)
select
    nextval('railways.seat_id_seq'::regclass),
    generate_series(1, a.number_of_seats),
    true,
    1000::money,
    c.scheduled_train_carriage_id
from
    railways.carriages_types a,
    railways.carriages b,
    railways.scheduled_train_carriages c
where
    c.scheduled_train_carriage_id = 1
    and c.carriage_id = b.carriage_id
    and b.carriage_type_id = a.carriage_type_id;

```

Создание простого и составного индексов для двух произвольных запросов и сравнить время выполнения запросов без индексов и с индексами. Для получения плана запроса использовать команду EXPLAIN.

```

-- создание индексов
-- простой
create index if not exists seats_scheduled_train_carriage_id_index on
railways.seats(scheduled train carriage id);

```

```

explain analyse
select distinct train_name, scheduled_train_carriages.carriage_order_number,
seat_number
from
    railways.timetable,
    railways.scheduled_train_carriages,
    railways.carriages,
    railways.seats,
    railways.trains
where
    timetable.departure_time <= date_trunc('day', current_date + interval
'1 day')
    and timetable.departure_time >= date_trunc('day', current_date)
    and timetable.scheduled_train_id =
scheduled_train_carriages.scheduled_train_id
    and scheduled_train_carriages.carriage_id = carriages.carriage_id
    and carriage_type_id = 1
    and seats.scheduled_train_carriage_id = timetable.scheduled_train_id
    and seats.is_empty
    and timetable.train_id = trains.train_id;

```

Без индекса	С индексом
Execution Time: 0.169 ms	Execution Time: 0.118 ms

Время выполнения запроса при использовании простого индекса незначительно отличается в лучшую в сравнение с запросом без использования индекса , поэтому

прогнозируется сильное увеличение производительности запроса при увеличении количества данных.

```
-- составной
create unique index if not exists timetable_uk_1 on
railways.timetable(train_id, departure_time, scheduled_train_id);
drop index railways.timetable_uk_1;
explain analyse select train_id, departure_time, scheduled_train_id from
railways.timetable;
```

Без индекса	С индексом
Execution Time: 0.027 ms	Execution Time: 0.029 ms

Время выполнения запроса при использовании и не использование составного индекса почти не отличается. Это объясняется маленькой выборкой данных, при которой быстрее пройти по всей таблице, а не по составным индексам. При увеличении количества данных предположительно время выполнения запроса с составным индексом будет меньше, чем без составного индекса (исходя из анализа данных в интернете).

Вывод

Были изучены практические навыки создания представлений и запросов на выборку данных к базе данных PostgreSQL, использования подзапросов при модификации данных и индексов.