

**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 .....	16

## **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.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')

```

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,

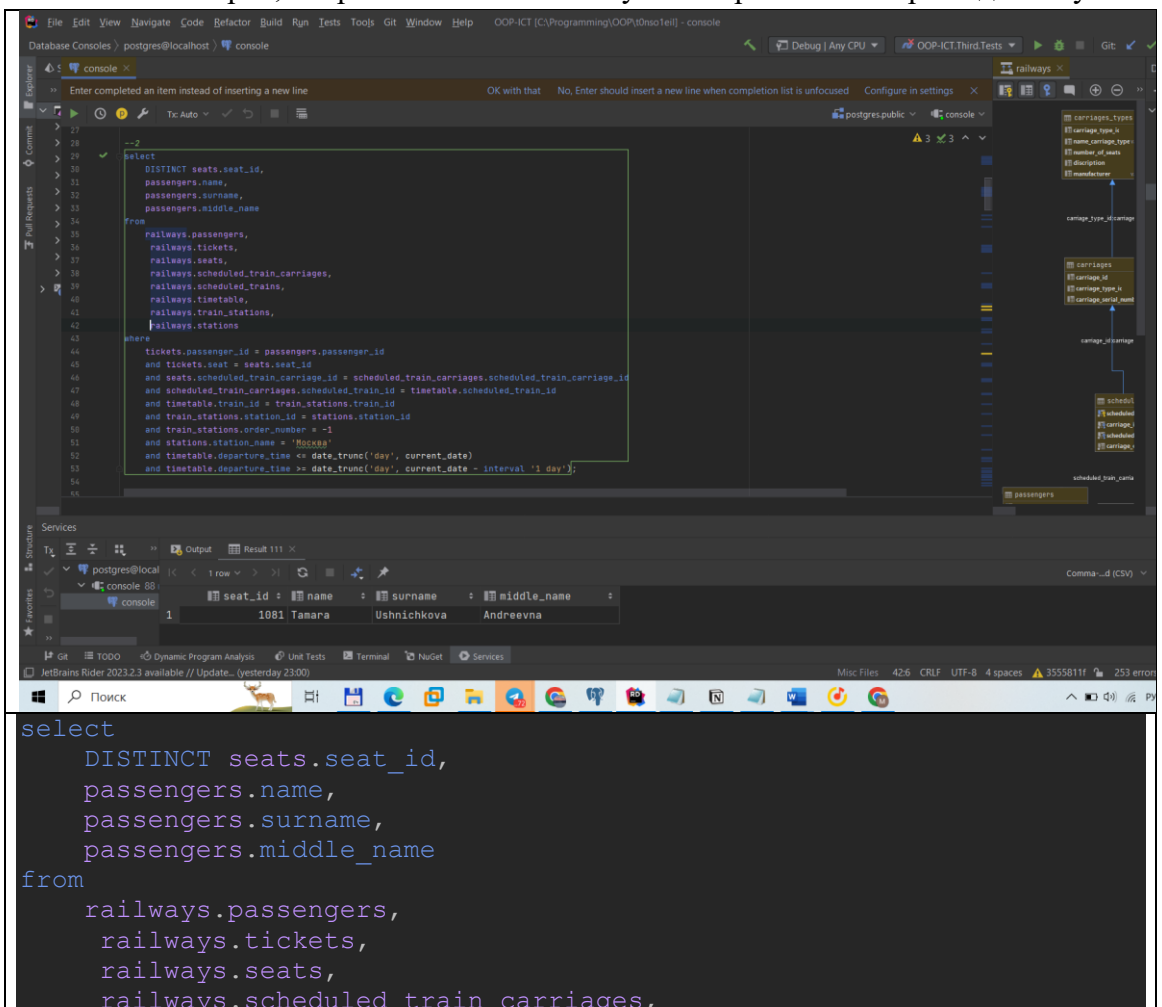
```

```

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,

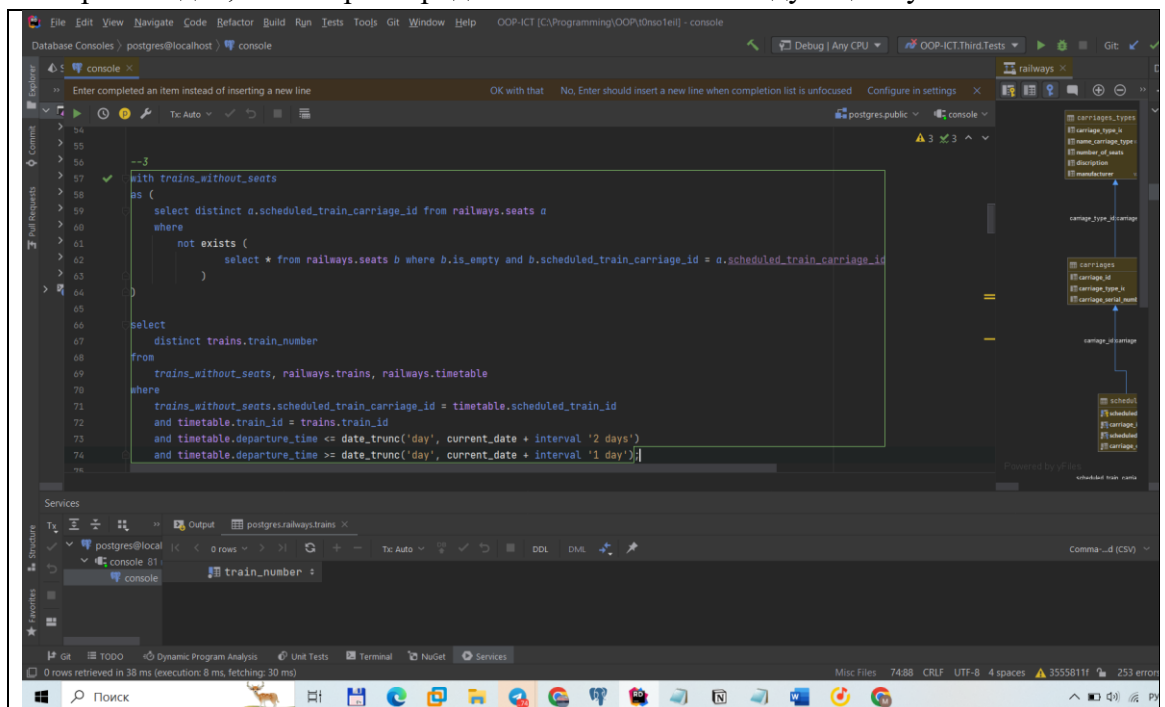
```

```

        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.



```

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 are displayed in a table with the following columns: train\_name, carriage\_order\_number, and seat\_number. The results show 5 rows of data for the train 'Sapsan (Spb - Moscow)'.

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.

The screenshot shows an IDE with a PostgreSQL console. The query being executed is:

```
--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;
```

The result shown in the console is:

sum
1185020,00

Below the screenshot, the query is repeated in a text box:

```
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”.

The screenshot shows an IDE with a PostgreSQL console. The query being executed is:

```
--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';
```

The result shown in the console is:

count
1

Below the screenshot, the query is repeated in a text box:

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

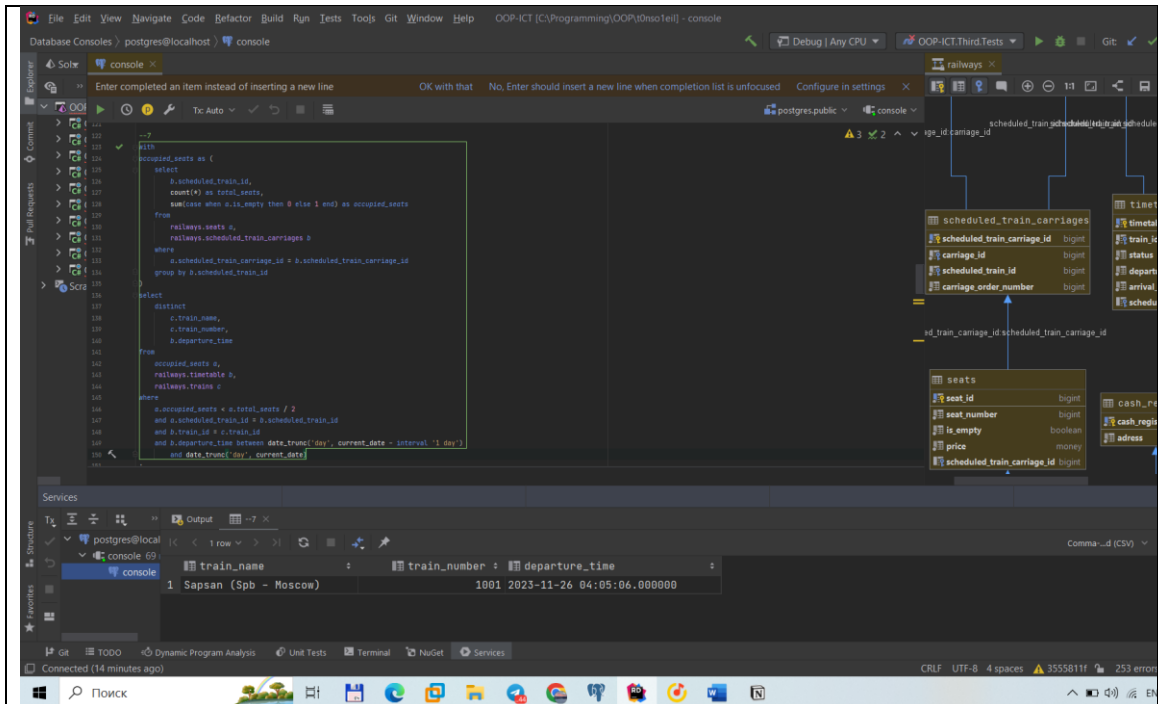


```

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.



```

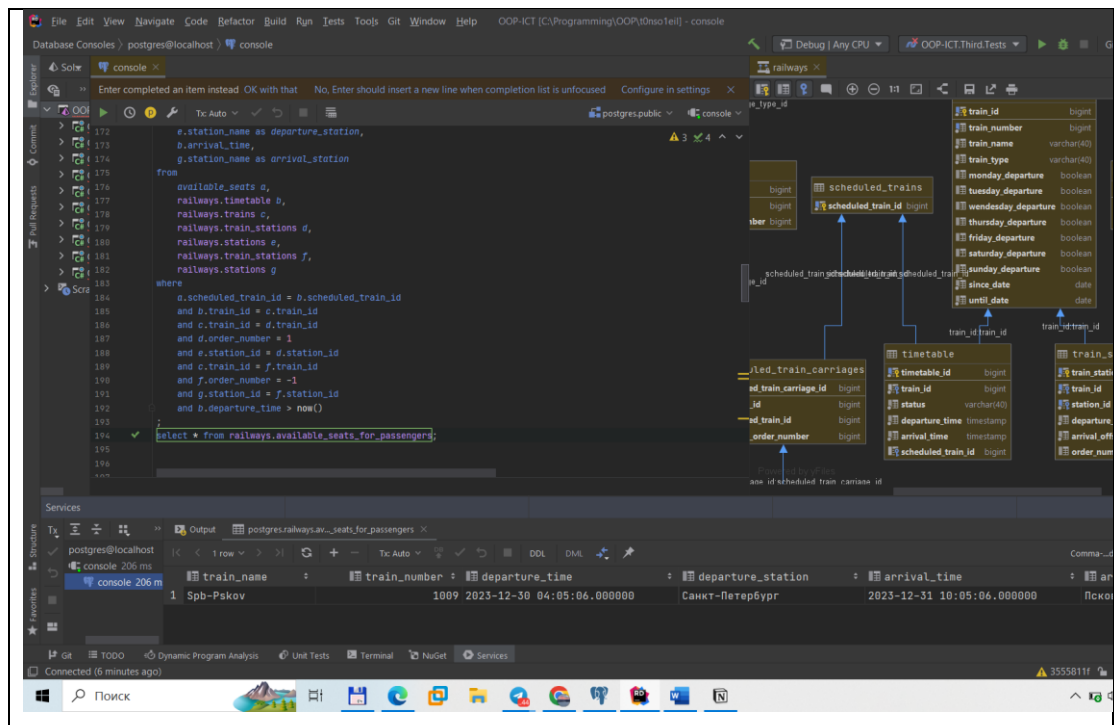
with
occupied_seats as (
    select
        b.scheduled_train_id,
        count(*) as total_seats,
        sum(case when a.is_empty then 0 else 1 end) as occupied_seats
    from
        railways.seats a,
        railways.scheduled_train_carriages b
    where
        a.scheduled_train_carriage_id = b.scheduled_train_carriage_id
    group by b.scheduled_train_id
)
select
    distinct
        c.train_name,
        c.train_number,
        b.departure_time
from
    occupied_seats a,
    railways.timetable b,
    railways.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).

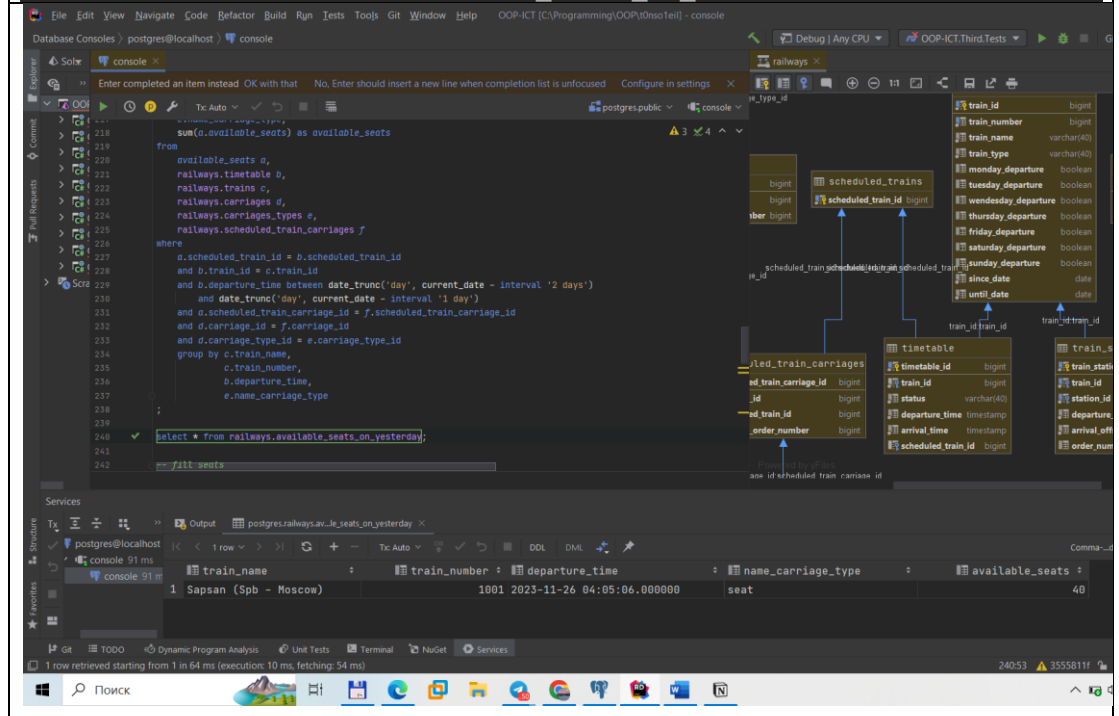
```
create or replace view railways.available_seats_on_yesterday as
with
    available_seats as (
        select
            b.scheduled_train_id,
            b.scheduled_train_carriage_id,
            sum(case when a.is_empty then 1 else 0 end) as
available_seats
        from
            railways.seats 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.

```

create or replace function railways.create_seats(
    id_scheduled_train_carriage    int,
    price                          money
) returns integer as $create_seats$
declare
    counter          integer := 0;
    number_of_seats integer;
begin
    number_of_seats = (
        select
            carriages_types.number_of_seats
        from
            railways.carriages_types,
            railways.carriages,
            railways.scheduled_train_carriages
        where

scheduled_train_carriages.scheduled_train_carriage_id =
id_scheduled_train_carriage
        and scheduled_train_carriages.carriage_id =

```

```

carriages.carriage_id
                                and carriages.carriage_type_id =
carriages_types.carriage_type_id
                                );
    if number_of_seats is null then number_of_seats = 0;
    end if;
    delete from railways.seats where seats.scheduled_train_carriage_id =
id_scheduled_train_carriage;
    for i in 1..number_of_seats loop
        insert into railways.seats(seat_id, seat_number, is_empty, price,
scheduled_train_carriage_id)
            values (nextval('railways.seat_id_seq'::regclass), i, true, price,
id_scheduled_train_carriage);
    end loop;
    return number_of_seats;
end;
$create_seats$ language plpgsql;

```

```

create or replace function railways.create_ticket(
    id_scheduled_train_carriage    int,
    id_cash_register               int,
    id_passenger                   int,
    way_of_paying                   character varying,
    ticket_buying_time              timestamp without time zone
) returns integer as $create_ticket$
declare
    counter                integer := 0;
    id_departure_station    integer;
    id_arrival_station      integer;
    id_seat                 integer;
begin
    id_departure_station = (
        select
            train_stations.station_id
        from
            railways.timetable,
            railways.train_stations,
            railways.scheduled_train_carriages
        where

scheduled_train_carriages.scheduled_train_carriage_id =
id_scheduled_train_carriage
                                and timetable.scheduled_train_id =
scheduled_train_carriages.scheduled_train_id
                                and timetable.train_id =
train_stations.train_id
                                and train_stations.order_number = 1
        );
    id_arrival_station = (
        select
            train_stations.station_id
        from
            railways.timetable,
            railways.train_stations,
            railways.scheduled_train_carriages
        where
            scheduled_train_carriages.scheduled_train_carriage_id =
id_scheduled_train_carriage
                                and timetable.scheduled_train_id =
scheduled_train_carriages.scheduled_train_id
                                and timetable.train_id = train_stations.train_id
                                and train_stations.order_number = - 1
    );

```

```

select
    seats.seat_id
into id_seat
from
    railways.seats,
    railways.scheduled_train_carriages
where
    scheduled_train_carriages.scheduled_train_id = 1
    and scheduled_train_carriages.scheduled_train_id =
seats.scheduled_train_carriage_id
    and seats.is_empty
LIMIT 1;

if id_seat is not null then
    insert into railways.tickets(ticket_id,
                                passenger_id,
                                cash_register_id,
                                departure_station_id,
                                arrival_station_id,
                                status,
                                buying_time,
                                seat,
                                way_of_paying)
    values (nextval('railways.ticket_id_seq'::regclass),
            id_passenger,
            id_cash_register,
            id_departure_station,
            id_arrival_station,
            'sold'::character_varying,
            ticket_buying_time,
            id_seat,
            way_of_paying_);
end if;
UPDATE railways.seats
SET is_empty = false
WHERE seat_id = id_seat;
return id_seat;
end;
$create_ticket$ language plpgsql;

```

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);
-- составной
create unique index if not exists timetable_uk_1 on
railways.timetable(train_id, departure_time, scheduled_train_id);
drop index railways.seats_scheduled_train_carriage_id_index

```

С индексом:

```

HashAggregate (cost=41.41..41.73 rows=32 width=114) (actual
time=0.042..0.044 rows=0 loops=1)
" Group Key: trains.train_name,
scheduled_train_carriages.carriage_order_number, seats.seat_number"
  Batches: 1  Memory Usage: 24kB
  -> Nested Loop (cost=1.97..41.17 rows=32 width=114) (actual
time=0.040..0.041 rows=0 loops=1)
    Join Filter: (timetable.scheduled_train_id =
seats.scheduled_train_carriage_id)

```

```

-> Nested Loop (cost=1.82..39.97 rows=1 width=122) (actual
time=0.039..0.041 rows=0 loops=1)
-> Nested Loop (cost=1.68..31.76 rows=1 width=32) (actual
time=0.039..0.041 rows=0 loops=1)
-> Hash Join (cost=1.52..30.29 rows=7 width=40) (actual
time=0.039..0.040 rows=0 loops=1)
    Hash Cond:
    (scheduled_train_carriages.scheduled_train_id = timetable.scheduled_train_id)
-> Seq Scan on scheduled_train_carriages
(cost=0.00..23.60 rows=1360 width=24) (actual time=0.015..0.015 rows=1
loops=1)
-> Hash (cost=1.51..1.51 rows=1 width=16) (actual
time=0.018..0.018 rows=0 loops=1)
    Buckets: 1024 Batches: 1 Memory Usage: 8kB
-> Seq Scan on timetable (cost=0.00..1.51
rows=1 width=16) (actual time=0.018..0.018 rows=0 loops=1)
"
    Filter: ((departure_time <=
date_trunc('day'::text, (CURRENT_DATE + '1 day'::interval))) AND
(departure_time >= date_trunc('day'::text, (CURRENT_DATE)::timestamp with
time zone)))"
    Rows Removed by Filter: 17
-> Index Scan using carriages_pkey on carriages
(cost=0.15..0.21 rows=1 width=8) (never executed)
    Index Cond: (carriage_id =
scheduled_train_carriages.carriage_id)
    Filter: (carriage_type_id = 1)
-> Index Scan using trains_pkey on trains (cost=0.15..8.17
rows=1 width=106) (never executed)
    Index Cond: (train_id = timetable.train_id)
-> Index Scan using seats_scheduled_train_carriage_id_index on seats
(cost=0.15..0.77 rows=34 width=16) (never executed)
    Index Cond: (scheduled_train_carriage_id =
scheduled_train_carriages.scheduled_train_id)
    Filter: is_empty
Planning Time: 0.490 ms
Execution Time: 0.095 ms

```

#### Без индекса:

```

HashAggregate (cost=54.11..54.43 rows=32 width=114) (actual
time=0.049..0.051 rows=0 loops=1)
" Group Key: trains.train_name,
scheduled_train_carriages.carriage_order_number, seats.seat_number"
    Batches: 1 Memory Usage: 24kB
-> Hash Join (cost=39.99..53.87 rows=32 width=114) (actual
time=0.048..0.049 rows=0 loops=1)
    Hash Cond: (seats.scheduled_train_carriage_id =
timetable.scheduled_train_id)
-> Seq Scan on seats (cost=0.00..11.50 rows=550 width=16) (actual
time=0.013..0.013 rows=1 loops=1)
    Filter: is_empty
    Rows Removed by Filter: 4
-> Hash (cost=39.97..39.97 rows=1 width=122) (actual
time=0.031..0.032 rows=0 loops=1)
    Buckets: 1024 Batches: 1 Memory Usage: 8kB
-> Nested Loop (cost=1.82..39.97 rows=1 width=122) (actual
time=0.031..0.032 rows=0 loops=1)
    -> Nested Loop (cost=1.68..31.76 rows=1 width=32)
(actual time=0.031..0.031 rows=0 loops=1)
    -> Hash Join (cost=1.52..30.29 rows=7 width=40)
(actual time=0.030..0.031 rows=0 loops=1)
    Hash Cond:
    (scheduled_train_carriages.scheduled_train_id =
timetable.scheduled_train_id)

```

```

-> Seq Scan on scheduled_train_carriages
(cost=0.00..23.60 rows=1360 width=24) (actual time=0.006..0.006 rows=1
loops=1)
-> Hash (cost=1.51..1.51 rows=1 width=16)
(actual time=0.017..0.017 rows=0 loops=1)
      Buckets: 1024 Batches: 1 Memory
Usage: 8kB
-> Seq Scan on timetable
(cost=0.00..1.51 rows=1 width=16) (actual time=0.017..0.017 rows=0 loops=1)
"
      Filter: ((departure_time <=
date_trunc('day'::text, (CURRENT_DATE + '1 day'::interval))) AND
(departure_time >= date_trunc('day'::text, (CURRENT_DATE)::timestamp with
time zone)))"
      Rows Removed by Filter: 17
-> Index Scan using carriages_pkey on carriages
(cost=0.15..0.21 rows=1 width=8) (never executed)
      Index Cond: (carriage_id =
scheduled_train_carriages.carriage_id)
      Filter: (carriage_type_id = 1)
-> Index Scan using trains_pkey on trains
(cost=0.15..8.17 rows=1 width=106) (never executed)
      Index Cond: (train_id = timetable.train_id)
Planning Time: 0.605 ms
Execution Time: 0.096 ms

```

Время выполнения запроса при использовании индексов незначительно отличается в лучшую, поэтому прогнозируется сравнительное улучшение эффективности работы при увеличении количество данных в выборке.

## Вывод

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