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»

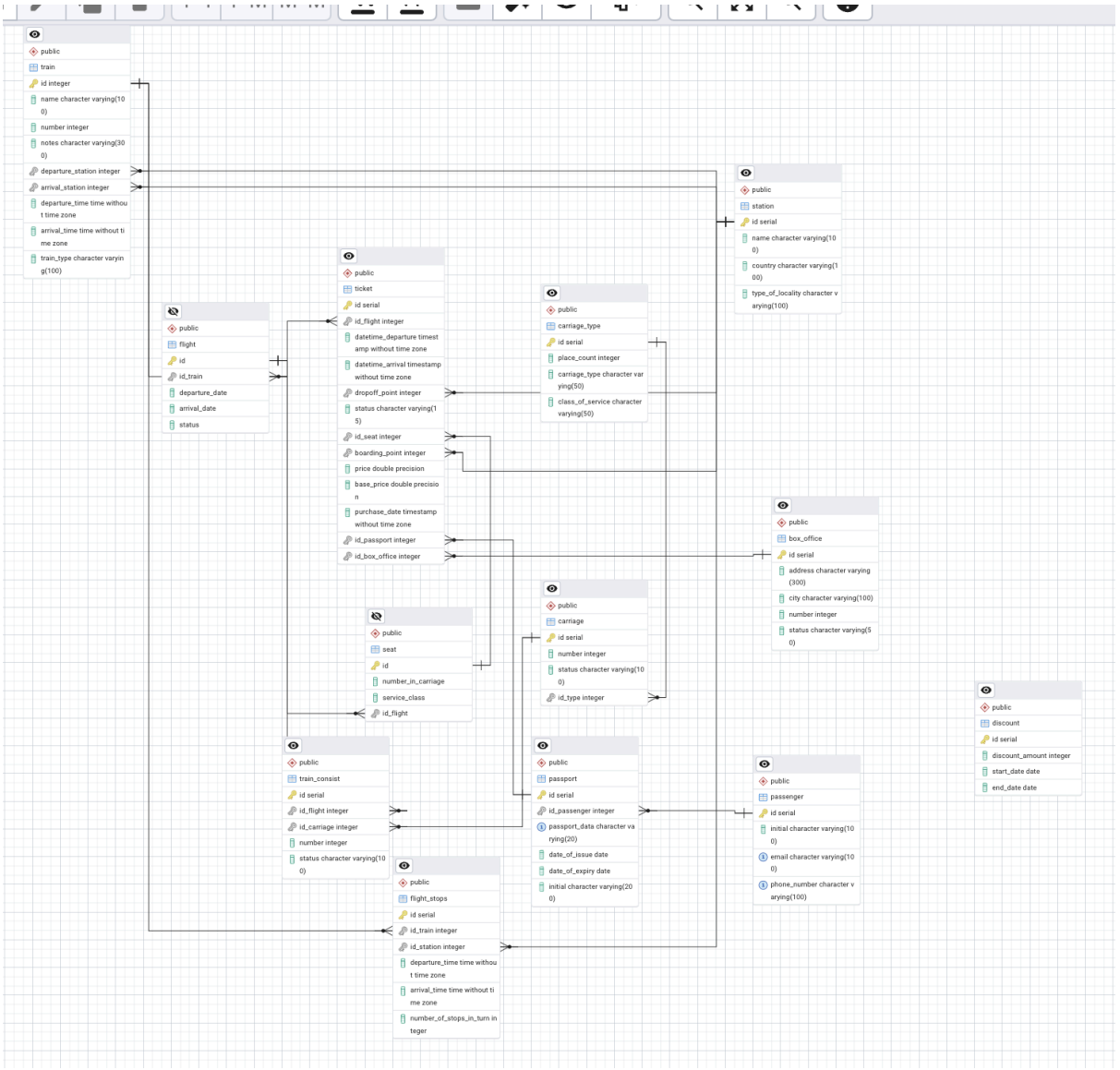
по дисциплине «Проектирование и реализация баз данных»

Обучающийся Архангельская Елизавета Павловна
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

Цель работы: овладеть практическими навыками создания представлений и запросов на выборку данных к базе данных PostgreSQL, использования подзапросов при модификации данных и индексов.

ERD диаграмма



Практическое задание:

1. Создать запросы и представления на выборку данных к базе данных PostgreSQL (согласно индивидуальному заданию лабораторной работы №2, часть 2 и 3).

Создать запросы:

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

```
select flight.id, train.name, seat.number_in_carriage, seat.service_class, station.name, flight.departure_date
from flight
join train on flight.id_train = train.id
```

```
join seat on seat.id_flight = flight.id
join station on train.departure_station = station.id
tefy join ticket on ticket.id_seat = seat.id and ticket.id_flight = flight.id
where flight.departure_date >= now()
        and flight.departure_date < now() + interval '1 day'
        and ticket.id is null;
```

Список поездов за прошедшие сутки с максимальной выручкой от продажи билетов.

```
select  train.id, train.name, flight.id, sum(ticket.price) as total_sum
from flight
join train on flight.id_train = train.id
join ticket on ticket.id_flight = flight.id
where flight.departure_date >= now() - interval '1 day'
   and flight.departure_date < now()
group by train.id, flight.id
order by total_sum desc;
```

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

```
select train.id, flight.id, train.name
from flight
join train on flight.id_train = train.id
join seat on seat.id_flight = flight.id
left join ticket on ticket.id_seat = seat.id and ticket.id_flight = flight.id
where flight.departure_date >= now()
  and flight.departure_date < now() + interval '1 day'
group by train.id, flight.id
having count(seat.id) = count(ticket.id);
```

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

```
select seat.id from seat
join flight on seat.id_flight = flight.id
join train on train.id = flight.id_train
join train_consist on flight.id = train_consist.id_flight
join carriage on train_consist.id_carriage = carriage.id
join carriage_type on carriage_type.id = carriage.id_type
join station on station.id = train.arrival_station
where seat.id not in (select id_seat from ticket) and carriage_type = 'купейный' and
(station.name like '%москва%' or station.name like '%Москва%') and departure_date =
current_date;
```

Выручка от продажи билетов на все поезда за прошедшие сутки.

```
select case when sum(price) is null then 0
   else sum(price)
        end as total_price
        from ticket
where  purchase_date > now() - interval '1 day' AND purchase_date < now() and status =
'оплачен';
```

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

```
select count(*) from ticket
join flight on ticket.id_flight = flight.id
join train_consist on train_consist.id_flight = flight.id
join carriage on carriage.id = train_consist.id_carriage
join carriage_type on carriage.id_type = carriage_type.id
where carriage_type = 'СВ' and ticket.status = 'оплачен';
```

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

```
select train.id, train.name, count(ticket.id), count(seat.id)
from flight
join train on flight.id_train = train.id
join seat on seat.id_flight = flight.id
left join ticket on ticket.id_seat = seat.id and ticket.id_flight = flight.id and ticket.status = 'оплачен'
where flight.departure_date >= now() - interval '1 day'
  and flight.departure_date < now()
group by train.id, flight.id
having count(ticket.id) < count(seat.id) / 2;
```

Создать представление:

для пассажиров о наличии свободных мест на заданный рейс;

```
create view available_seats as
select flight.id as flight_id,
    train.number as train_number,
    carriage.number as carriage_number,
    carriage_type.carriage_type as carriage_type,
    seat.number_in_carriage as seat_number,
    seat.service_class as service_class
from
    flight
join train on flight.id_train = train.id
join train_consist on train_consist.id_flight = flight.id
join carriage on carriage.id = train_consist.id_carriage
join carriage_type  on carriage_type.id = carriage.id_type
join seat on seat.id_flight = flight.id
left join ticket on ticket.id_flight = flight.id and ticket.id_seat = seat.id
where
    ticket.id is null;
```

```
1  select * from available_seats;
```

Data Output | Messages | Notifications

Showing rows: 1 to 1000 | Page N

	flight_id integer	train_number integer	carriage_number integer	carriage_type character varying (50)	seat_number integer	service_class character varying (10)
1	2	32	10	вагон-ресторан	6	комфорт
2	2	32	10	вагон-ресторан	5	комфорт
3	2	32	10	вагон-ресторан	2	комфорт
4	2	32	10	вагон-ресторан	10	комфорт
5	2	32	10	вагон-ресторан	1	комфорт
6	2	32	11	плацкартный	6	комфорт
7	2	32	11	плацкартный	5	комфорт
8	2	32	11	плацкартный	2	комфорт
9	2	32	11	плацкартный	10	комфорт
10	2	32	11	плацкартный	1	комфорт
11	2	32	12	купейный	6	комфорт
12	2	32	12	купейный	5	комфорт
13	2	32	12	купейный	2	комфорт
14	2	32	12	купейный	10	комфорт
15	2	32	12	купейный	1	комфорт
16	3	18	8	плацкартный	21	комфорт
17	3	18	8	плацкартный	25	эконом
18	3	18	8	плацкартный	2	комфорт
19	3	18	8	плацкартный	9	комфорт
20	3	18	8	плацкартный	11	эконом

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

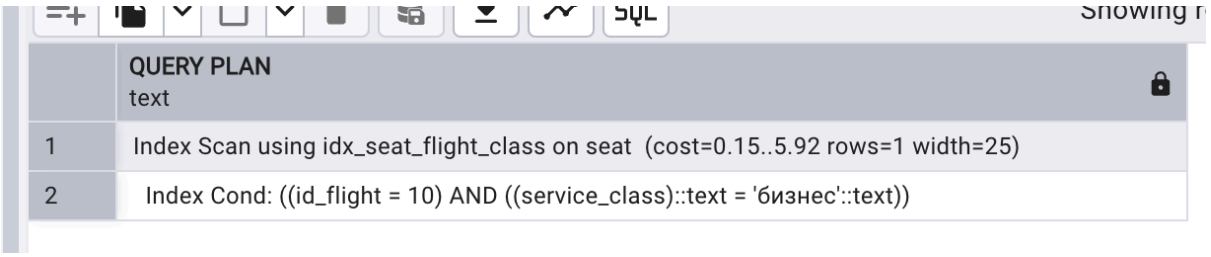
```
create view unsold_tickets_last_day as
select train.number as train_number, carriage_type.carriage_type as carriage_type,
    count(seat.id) as unsold_seats
from train
join flight on flight.id_train = train.id
join train_consist on train_consist.id_flight = flight.id
join carriage on carriage.id = train_consist.id_carriage
join carriage_type on carriage_type.id = carriage.id_type
join seat on seat.id_flight = flight.id
left join ticket on ticket.id_seat = seat.id and ticket.id_flight = flight.id
where
    flight.departure_date >= now() - interval '1 day'
    and flight.departure_date <= now()
    and ticket.id is null
group by
    train.number,
    carriage_type.carriage_type;
```

2. Составить 3 запроса на модификацию данных (INSERT, UPDATE, DELETE) с использованием подзапросов.

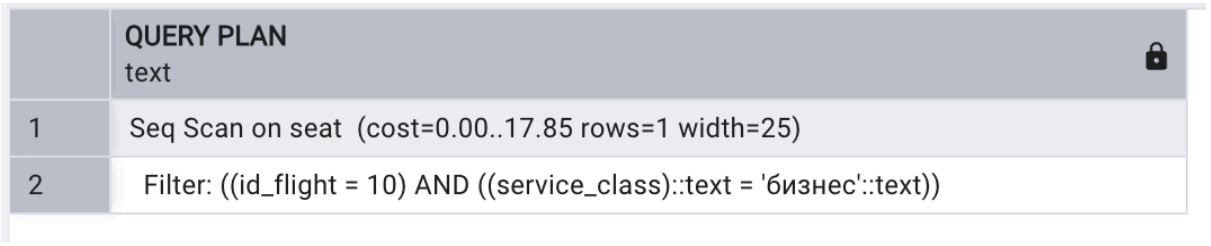
```
update passenger
set email = 'new_email@mail.ru'
where id in (
   select id_passenger
   from passport
   where passport_data = '0878213395'
);

delete from ticket
where id_passport in (
   select id
   from passport
   where id_passenger = (
      select id
      from passenger
      where email = 'sergei.ivanov@gmail.com'
   )
);
insert into passport (id_passenger, passport_data, date_of_issue, date_of_expiry, initial)
values (
   (select id from passenger where phone_number = '+79214567890'),
   '0113449938', '2025-05-29', '2035-05-28',
   (select initial from passenger where phone_number = '+79214567890')
);
```

3. 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.

```
create index idx_seat_flight_class on seat(id_flight, service_class);
explain select * from seat from id_flight = 10 and service_class = 'бизнес';
```

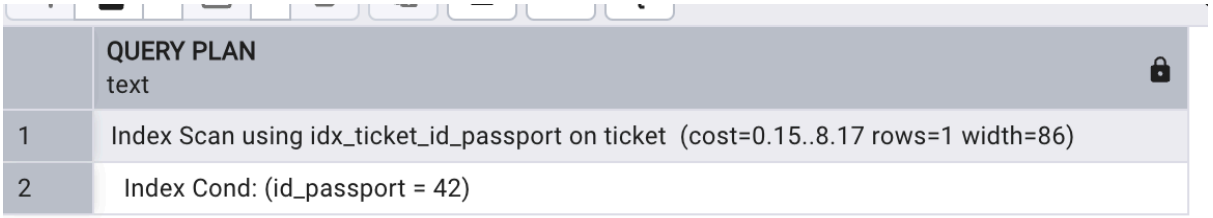


	QUERY PLAN text
1	Index Scan using idx_seat_flight_class on seat (cost=0.15..5.92 rows=1 width=25)
2	Index Cond: ((id_flight = 10) AND ((service_class)::text = 'бизнес'::text))

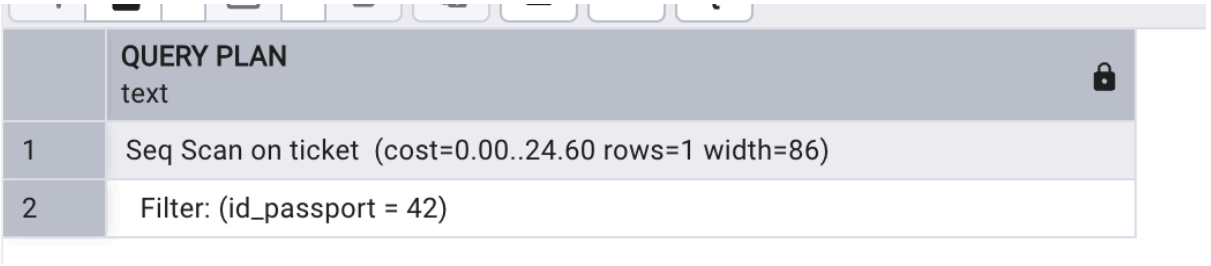


	QUERY PLAN text
1	Seq Scan on seat (cost=0.00..17.85 rows=1 width=25)
2	Filter: ((id_flight = 10) AND ((service_class)::text = 'бизнес'::text))

create index idx_ticket_id_passport on ticket(id_passport);
explain select * from ticket from id_passport = 42;



	QUERY PLAN text
1	Index Scan using idx_ticket_id_passport on ticket (cost=0.15..8.17 rows=1 width=86)
2	Index Cond: (id_passport = 42)



	QUERY PLAN text
1	Seq Scan on ticket (cost=0.00..24.60 rows=1 width=86)
2	Filter: (id_passport = 42)

Выводы:
В ходе выполнения лабораторной работы были закреплены практические навыки проектирования и работы с реляционными базами данных. Освоены инструменты визуализации структуры базы данных с помощью ERD-генератора в pgAdmin, что позволило наглядно представить связи между таблицами. Были успешно реализованы сложные SQL-запросы, включая запросы с подзапросами, а также создание представлений для автоматизации получения аналитической информации. Проведено сравнение производительности запросов до и после создания индексов, что подтвердило значительное ускорение выборок при наличии соответствующих индексов. Полученные результаты демонстрируют важность грамотного проектирования структуры базы данных и оптимизации запросов для повышения эффективности работы информационных систем.