

**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 (Синюков Лев Владимирович)

Факультет прикладной информатики

Группа __3240__

Направление подготовки 09.03.03 Прикладная информатика

Образовательная программа Мобильные и сетевые технологии 2023

Преподаватель Говорова Марина Михайловна

Санкт-Петербург
2024/2025

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 лабораторной работы №2, 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.

Индивидуальное задание:

Вариант 10. БД «Автовокзал»

Описание предметной области: С автовокзала ежедневно отправляется несколько междугородных/международных автобусных рейсов. Номер рейса определяется маршрутом и временем отправления. По всем промежуточным остановкам на маршруте известны название, тип населенного пункта, время прибытия, отправления, время стоянки.

Автобусы курсируют по расписанию, но могут назначаться дополнительные рейсы на заданный период или определенные даты.

Билеты могут продаваться предварительно, но не ранее чем за 10 суток. Билет может быть приобретен в кассе автовокзала или онлайн. В билете указывается номер места в автобусе (необязательно). На каждый рейс может продаваться не более 10 билетов без места, цена на которые снижается на 10%. Пунктами отправления и назначения, согласно билету, могут быть промежуточные остановки.

Билеты могут продаваться в кассе автовокзала или онлайн.

Необходимо учитывать, что местом посадки и высадки пассажира могут быть промежуточные остановки согласно купленному билету.

На каждый рейс формируется экипаж из двух водителей. Необходимо хранить данные о прохождении медосмотра перед рейсом (дата, статус, причина недопуска).

БД должна содержать следующий минимальный набор сведений: Номер рейса. Номер водителя. Номер автобуса. Паспортные данные водителя. Пункт отправления. Пункт назначения. Промежуточные остановки. Дата отправления. Время отправления. Время в пути. Тип автобуса. Количество мест в автобусе. Страна. Производитель. Год выпуска. Номер билета. Номер места в автобусе (при наличии). Цена билета. ФИО пассажира. Паспортные данные пассажира.

Дополните состав атрибутов на основе анализа предметной области.

Задание 2. Создать запросы:

- Вывести фамилии водителей и номера автобусов, отправившиеся в рейсы до 12 часов текущего дня.
- Рассчитать выручку от продажи билетов за прошедший день.
- Вывести список водителей, которые не выполнили ни одного рейса за прошедшую неделю.
- Вывести сумму убытков из-за непроданных мест в автобусе за прошедшую неделю.
- Найти самый популярный маршрут за прошедший месяц.
- Вывести тип автобуса, который используется на всех рейсах.

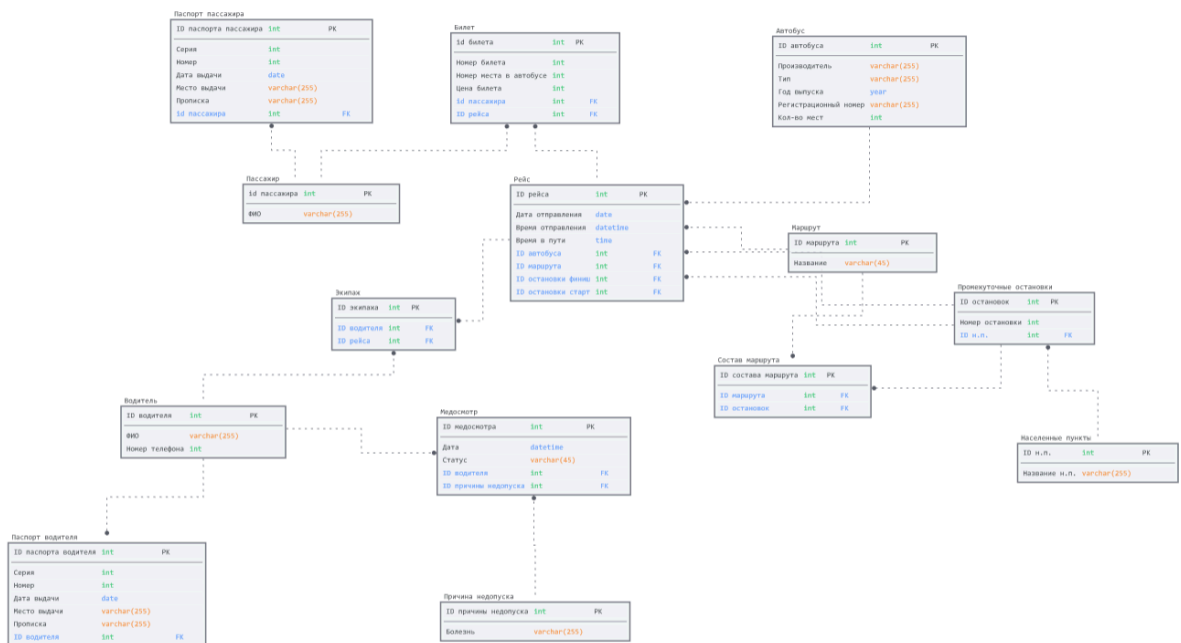
- Вывести данные водителя, который провел максимальное время в пути за прошедшую неделю.

Задание 3. Создать представление для пассажиров:

- количество свободных мест на все рейсы на завтра;
- самый популярный маршрут этой зимой.

Выполнение

Схема инфологической модели данных БД в нотации IDEF1X



Дамп БД из предыдущей работы

Создание БД:

```
-- Создание базы данных
CREATE DATABASE bus_system;
\c bus_system;
```

Добавление таблиц:

```
-- Населенные пункты
CREATE TABLE settlement (
    settlement_id SERIAL PRIMARY KEY,
    name VARCHAR(255) NOT NULL
);

-- Модель автобуса, для ссылки в таблицу автобус
```

```

CREATE TABLE bus_model (
    bus_model_id SERIAL PRIMARY KEY,
    manufacturer VARCHAR(255),
    model_name VARCHAR(255),
    seats_number INT NOT NULL
);

-- Автобус, добавлена связь с моделью
CREATE TABLE bus (
    bus_id SERIAL PRIMARY KEY,
    bus_model_id INT REFERENCES bus_model(bus_model_id),
    production_year INT,
    registration_number VARCHAR(255)
);

-- Маршрут. Добавлены время отправления, прибытия, в пути
CREATE TABLE route (
    route_id SERIAL PRIMARY KEY,
    name VARCHAR(255),
    start_settlement_id INT REFERENCES settlement(settlement_id),
    final_settlement_id INT REFERENCES settlement(settlement_id),
    departure_time TIME,
    arrival_time TIME,
    travel_time INTERVAL
);

-- Маршрут
CREATE TABLE route_composition (
    route_composition_id SERIAL PRIMARY KEY,
    route_id INT REFERENCES route(route_id),
    settlement_id INT REFERENCES settlement(settlement_id),
    stop_number INT,
    arrival_time TIME,
    stop_duration INTERVAL
);

-- Рейс. Добавлен статус, стоимость
CREATE TABLE trip (
    trip_id SERIAL PRIMARY KEY,
    route_id INT REFERENCES route(route_id),
    bus_id INT REFERENCES bus(bus_id),
    departure_date DATE,
    status VARCHAR(50),
    price NUMERIC(10, 2)
);

```

```

-- Пассажир
CREATE TABLE passenger (
    passenger_id SERIAL PRIMARY KEY,
    full_name VARCHAR(255) NOT NULL
);

-- Паспорт пассажира
CREATE TABLE passport_passenger (
    passport_id SERIAL PRIMARY KEY,
    series INT,
    number INT,
    issue_date DATE,
    issue_place VARCHAR(255),
    registration_address VARCHAR(255),
    passenger_id INT UNIQUE REFERENCES passenger(passenger_id)
);

-- Билет. Связь с паспортом пассажира
CREATE TABLE ticket (
    ticket_id SERIAL PRIMARY KEY,
    trip_id INT REFERENCES trip(trip_id),
    passport_passenger_id INT REFERENCES passport_passenger(passport_id),
    seat_number INT,
    status VARCHAR(50)
);

-- Водитель
CREATE TABLE driver (
    driver_id SERIAL PRIMARY KEY,
    full_name VARCHAR(255),
    phone_number VARCHAR(20)
);

-- Паспорт водителя
CREATE TABLE passport_driver (
    passport_id SERIAL PRIMARY KEY,
    series INT,
    number INT,
    issue_date DATE,
    issue_place VARCHAR(255),
    registration_address VARCHAR(255),
    driver_id INT UNIQUE REFERENCES driver(driver_id)
);

-- Причины недопуска по медосмотру
CREATE TABLE reason_for_disqualification (

```

```

        reason_id SERIAL PRIMARY KEY,
        disease VARCHAR(255)
    );

-- Экипаж. Связь с паспортом водителя. Добавлен статус медосмотра, дата
CREATE TABLE crew (
    crew_id SERIAL PRIMARY KEY,
    trip_id INT REFERENCES trip(trip_id),
    passport_driver_id INT REFERENCES passport_driver(passport_id),
    medical_status VARCHAR(50),
    medical_checkup_date DATE,
    reason_id INT REFERENCES reason_for_disqualification(reason_id)
);

```

Далее проверим запросы на вставку данных:

```

INSERT INTO settlement (name) VALUES ('Moscow'), ('St. Petersburg'),
('Pskov');

INSERT INTO bus_model (manufacturer, model_name, seats_number)
VALUES ('GAZ', 'Gazelle', 12);

INSERT INTO bus (bus_model_id, production_year, registration_number)
VALUES (1, 2020, 'A777AA77');

INSERT INTO route (name, start_settlement_id, final_settlement_id,
departure_time, arrival_time, travel_time)
VALUES ('Moscow - St. Petersburg', 1, 2, '08:00', '15:00', '07:00'),
('Pskov - Moscow', 3, 1, '09:00', '17:00', '08:00');

INSERT INTO route_composition (route_id, settlement_id, stop_number,
arrival_time, stop_duration)
VALUES (1, 1, 1, '08:00', '00:10'),
(1, 2, 2, '15:00', '00:00');

INSERT INTO trip (route_id, bus_id, departure_date, status, price)
VALUES (1, 1, '2025-01-01', 'Boarding', 1500.00);

INSERT INTO passenger (full_name) VALUES ('Andrey Sahur');

INSERT INTO passport_passenger (series, number, issue_date, issue_place,
registration_address, passenger_id)
VALUES (1111, 777111, '2002-02-01', 'Moscow', 'Moscow, Lesnaya st. 13',
1);

INSERT INTO ticket (trip_id, passport_passenger_id, seat_number, status)

```

```
VALUES (1, 1, 5, 'Booked');

INSERT INTO driver (full_name, phone_number)
VALUES ('Ivan Ivanich', '79992220222');

INSERT INTO passport_driver (series, number, issue_date, issue_place,
registration_address, driver_id)
VALUES (1234, 123456, '1999-05-01', 'Pskov', 'Pskov, Krasniy ave. 1', 1);

INSERT INTO reason_for_disqualification (disease)
VALUES ('Heart disease');

INSERT INTO crew (trip_id, passport_driver_id, medical_status,
medical_checkup_date, reason_id)
VALUES (1, 1, 'Passed', '2025-01-01', NULL);
```

Выполним запросы по заданию:

```
-- Вывести фамилии водителей и номера автобусов, отправившиеся в рейсы до
12 часов текущего дня
SELECT driver.full_name, bus.registration_number FROM trip
JOIN crew ON crew.trip_id = trip.trip_id
JOIN passport_driver ON crew.passport_driver_id =
passport_driver.passport_id
JOIN driver ON driver.driver_id = passport_driver.driver_id
JOIN bus ON bus.bus_id = trip.bus_id
JOIN route ON route.route_id = trip.route_id
WHERE trip.departure_date = CURRENT_DATE AND route.departure_time <
'12:00';

-- Рассчитать выручку от продажи билетов за прошедший день
SELECT SUM(trip.price) AS total_revenue FROM trip
JOIN ticket ON ticket.trip_id = trip.trip_id
WHERE trip.departure_date = CURRENT_DATE - 1 AND ticket.status =
'Purchased';

-- Вывести список водителей, которые не выполнили ни одного рейса за
прошедшую неделю
SELECT driver.full_name FROM driver
LEFT JOIN passport_driver ON passport_driver.driver_id = driver.driver_id
LEFT JOIN crew ON crew.passport_driver_id = passport_driver.passport_id
LEFT JOIN trip ON trip.trip_id = crew.trip_id AND trip.departure_date >=
CURRENT_DATE - 7
WHERE crew.trip_id IS NULL OR trip.trip_id IS NULL;
```

```

-- Вывести сумму убытков из-за непроданных мест в автобусе за прошедшую
неделю
SELECT SUM((bus_model.seats_number - COALESCE(sold_tickets.count, 0)) *
trip.price) AS lost_revenue FROM trip
JOIN bus ON bus.bus_id = trip.bus_id
JOIN bus_model ON bus_model.bus_model_id = bus.bus_model_id
LEFT JOIN (
    SELECT ticket.trip_id, COUNT(*) AS count
    FROM ticket
    WHERE ticket.status = 'Purchased'
    GROUP BY ticket.trip_id
) sold_tickets ON sold_tickets.trip_id = trip.trip_id
WHERE trip.departure_date BETWEEN CURRENT_DATE - 7 AND CURRENT_DATE;

-- Найти самый популярный маршрут за прошедший месяц
SELECT route.name, COUNT(trip.trip_id) AS trip_count FROM route
JOIN trip ON trip.route_id = route.route_id
WHERE trip.departure_date BETWEEN CURRENT_DATE - INTERVAL '1 month' AND
CURRENT_DATE
GROUP BY route.route_id, route.name
HAVING COUNT(trip.trip_id) = (
    SELECT MAX(trip_count)
    FROM (
        SELECT COUNT(trip.trip_id) AS trip_count FROM route
        JOIN trip ON trip.route_id = route.route_id
        WHERE trip.departure_date BETWEEN CURRENT_DATE - INTERVAL '1
month' AND CURRENT_DATE
        GROUP BY route.route_id
    ) AS counts
);

-- Вывести тип автобуса, который используется на всех рейсах
SELECT bus_model.manufacturer, bus_model.model_name FROM bus_model
JOIN bus ON bus.bus_model_id = bus_model.bus_model_id
JOIN trip ON trip.bus_id = bus.bus_id
GROUP BY bus_model.bus_model_id
HAVING COUNT(DISTINCT trip.route_id) = (SELECT COUNT(DISTINCT route_id)
FROM trip);

-- Вывести данные водителя, который провел максимальное время в пути за
прошедшую неделю
SELECT driver.full_name, SUM(route.travel_time) AS total_travel_time
FROM crew

```



```

JOIN      passport_driver      ON      crew.passport_driver_id      =
passport_driver.passport_id
JOIN driver ON driver.driver_id = passport_driver.driver_id
JOIN trip ON trip.trip_id = crew.trip_id
JOIN route ON route.route_id = trip.route_id
WHERE trip.departure_date BETWEEN CURRENT_DATE - INTERVAL '7 days' AND
CURRENT_DATE
GROUP BY driver.driver_id, driver.full_name
HAVING SUM(route.travel_time) = (
    SELECT MAX(total_travel_time)
    FROM (
        SELECT SUM(route.travel_time) AS total_travel_time
        FROM crew
            JOIN      passport_driver      ON      crew.passport_driver_id      =
passport_driver.passport_id
            JOIN driver ON driver.driver_id = passport_driver.driver_id
            JOIN trip ON trip.trip_id = crew.trip_id
            JOIN route ON route.route_id = trip.route_id
            WHERE trip.departure_date BETWEEN CURRENT_DATE - INTERVAL '7 days'
AND CURRENT_DATE
            GROUP BY driver.driver_id
        ) AS max_times
    );

```

И представления:

```

-- ПРЕДСТАВЛЕНИЯ

-- Количество свободных мест на все рейсы на завтра
CREATE VIEW view_free_seats_tomorrow AS
SELECT
    trip.trip_id,
    route.name AS route_name,
    trip.departure_date,
    bus.registration_number,
    bus_model.seats_number,
    COALESCE(sold_tickets.sold_count, 0) AS sold_seats,
    (bus_model.seats_number - COALESCE(sold_tickets.sold_count, 0)) AS
free_seats
FROM trip
JOIN route ON route.route_id = trip.route_id
JOIN bus ON bus.bus_id = trip.bus_id
JOIN bus_model ON bus_model.bus_model_id = bus.bus_model_id
LEFT JOIN (
    SELECT ticket.trip_id, COUNT(*) AS sold_count FROM ticket
    WHERE ticket.status = 'Purchased'

```

```

        GROUP BY ticket.trip_id
    ) sold_tickets ON sold_tickets.trip_id = trip.trip_id
WHERE trip.departure_date = CURRENT_DATE + 1;

-- Самый популярный маршрут этой зимой
CREATE VIEW view_popular_route_winter AS
SELECT route.name, COUNT(trip.trip_id) AS trip_count FROM route
JOIN trip ON trip.route_id = route.route_id
WHERE trip.departure_date BETWEEN '2025-12-01' AND '2026-02-28'
GROUP BY route.route_id
ORDER BY trip_count DESC
LIMIT 1;

```

Теперь выполним три подзапроса:

```

-- 1. INSERT. Определим пассажира с наименьшим количеством билетов и
выдадим ему билет на завтра
INSERT INTO ticket (trip_id, passport_passenger_id, seat_number, status)
SELECT
    trip.trip_id,
    passport_passenger.passport_id,
    10,
    'Purchased'
FROM passport_passenger
JOIN passenger ON passenger.passenger_id = passport_passenger.passenger_id
JOIN trip ON trip.departure_date = CURRENT_DATE + 1
WHERE passport_passenger.passport_id = (
    SELECT passport_passenger_id
    FROM ticket
    GROUP BY passport_passenger_id
    ORDER BY COUNT(*) ASC
    LIMIT 1
);

```

До выполнения подзапроса:

QueryQuery History

1

SELECT passport_passenger_id, COUNT(*) AS ticket_count FROM ticket

2

GROUP BY passport_passenger_id;

Data OutputMessagesNotifications

≡+

▼

▼

SQL

	passport_passenger_id integer	ticket_count bigint
1	1	11

После:

Query Query History

1

SELECT passport_passenger_id, COUNT(*) AS ticket_count FROM ticket

2

GROUP BY passport_passenger_id;

Data Output Messages Notifications

SQL

	passport_passenger_id integer	ticket_count bigint
1	1	12

```
-- 2. UPDATE. Найдем рейс с наибольшей ценой билета и обновим статус для
каждого
UPDATE ticket
SET status = 'Cancelled'
WHERE trip_id = (
    SELECT trip_id
    FROM trip
    ORDER BY price DESC
    LIMIT 1
);
```

До выполнения подзапроса:

1 **SELECT** * **FROM** trip **ORDER BY** price **DESC LIMIT** 1;

Data Output Messages Notifications

	trip_id [PK] integer	route_id integer	bus_id integer	departure_date date	status character varying (50)	price numeric (10,2)
1	11	2	1	2026-01-15	Completed	2000.00

После:

Query

Query History

1

2

3

4

5

SELECT * FROM ticket

WHERE trip_id = (

SELECT trip_id FROM trip ORDER BY price DESC LIMIT 1

);

Data Output

Messages

Notifications

≡+

📄

▼

📋

▼

🗑️

🗄️

⬇️

📈

SQL

	ticket_id [PK] integer	trip_id integer	passport_passenger_id integer	seat_number integer	status character varying (50)
1	10	11	1	9	Cancelled

-- 3. DELETE. Определим самый непопулярный маршрут и удалим все билеты для этого маршрута

```
DELETE FROM ticket WHERE trip_id IN (
    SELECT trip_id FROM trip
    WHERE route_id IN (
        SELECT route_id FROM (
            SELECT trip.route_id, COUNT(ticket.ticket_id) AS ticket_count
            FROM trip
            JOIN ticket ON ticket.trip_id = trip.trip_id
            WHERE trip.departure_date > CURRENT_DATE - INTERVAL '1 month'
            GROUP BY trip.route_id
        ) AS route_ticket_counts
        WHERE ticket_count IN (
            SELECT MIN(ticket_count)
            FROM (
                SELECT trip.route_id, COUNT(ticket.ticket_id) AS
ticket_count
```


Query

Query History

1

2

SELECT * FROM ticket WHERE trip_id = 5;

Data Output

Messages

Notifications

≡+

▼

▼

SQL

	<div>ticket_id</div> <div>[PK] integer</div>	<div>trip_id</div> <div>integer</div>	<div>passport_passenger_id</div> <div>integer</div>	<div>seat_number</div> <div>integer</div>	<div>status</div> <div>character varying (50)</div>
1	18	5	1	77	Purchased

После:

Query Query History

1

SELECT * FROM ticket WHERE trip_id = 5;

2

Data Output Messages Notifications

≡+

▼

▼

SQL

ticket_id	trip_id	passport_passenger_id	seat_number	status
[PK] integer	integer	integer	integer	character varying (50)

Индексы

Выполним запрос без индекса:

Query

Query History

1

▼

EXPLAIN ANALYZE

2

SELECT * FROM ticket WHERE status = 'Purchased';

Data Output

Messages

Notifications

≡+

▼

▼

SQL

QUERY PLAN

text

🔒

1

Seq Scan on ticket (cost=0.00..16.25 rows=2 width=134) (actual time=0.021..0.026 rows=15 loops=...

2

Filter: ((status)::text = 'Purchased'::text)

3

Rows Removed by Filter: 2

4

Planning Time: 0.274 ms

5

Execution Time: 0.133 ms

Теперь выполним составной запрос:

Query

Query History

1

EXPLAIN ANALYZE

2

SELECT * FROM ticket

3

WHERE passport_passenger_id = 1 AND status = 'Purchased';

Data Output

Messages

Notifications


SQL

	QUERY PLAN	
	text	
1	Seq Scan on ticket (cost=0.00..17.50 rows=1 width=134) (actual time=0.014..0.017 rows=15 loops=...	
2	Filter: ((passport_passenger_id = 1) AND ((status)::text = 'Purchased'::text))	
3	Rows Removed by Filter: 2	
4	Planning Time: 0.143 ms	
5	Execution Time: 0.031 ms	

Теперь создадим индекс и выполним первый запрос:

```
1 CREATE INDEX idx_ticket_status ON ticket(status);
2 EXPLAIN ANALYZE
3 SELECT * FROM ticket WHERE status = 'Purchased';
4
```



	QUERY PLAN	
	text	
1	Seq Scan on ticket (cost=0.00..1.21 rows=1 width=134) (actual time=0.016..0.019 rows=15 loops=...	
2	Filter: ((status)::text = 'Purchased'::text)	
3	Rows Removed by Filter: 2	
4	Planning Time: 0.637 ms	
5	Execution Time: 0.038 ms	


Теперь создадим составной индекс и выполним составной запрос:

Query Query History

```
1 CREATE INDEX idx_ticket_passenger_status ON ticket(passport_passenger_id, status);
2 EXPLAIN ANALYZE
3 SELECT * FROM ticket
4 WHERE passport_passenger_id = 1 AND status = 'Purchased';
```

Data Output Messages Notifications



	QUERY PLAN	
	text	
1	Seq Scan on ticket (cost=0.00..1.25 rows=1 width=134) (actual time=0.009..0.011 rows=15 loops=...	
2	Filter: ((passport_passenger_id = 1) AND ((status)::text = 'Purchased'::text))	
3	Rows Removed by Filter: 2	
4	Planning Time: 0.442 ms	
5	Execution Time: 0.024 ms	

Можно заметить, что время выполнения с индексами значительно меньше, чем без них.

Теперь удалим индексы:

```
1 DROP INDEX IF EXISTS idx_ticket_status;  
2 DROP INDEX IF EXISTS idx_ticket_passenger_status;
```

Data Output Messages Notifications

DROP INDEX

Query returned successfully in 53 msec.

Выводы

При работе над лабораторной работой 4 я узнал, как выполнять запросы к базам данных и выполнил запросы на создание представлений. С использованием подзапросов я создал 3 запроса на модификацию данных (INSERT, UPDATE, DELETE). Также в ходе выполнения лабораторной работы я создал индексы и сравнил время выполнения запросов.