

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

**Отчет**

По лабораторной работе №2

**«ЗАПРОСЫ НА ВЫБОРКУ И МОДИФИКАЦИЮ ДАННЫХ,  
ПРЕДСТАВЛЕНИЯ И ИНДЕКСЫ В POSTGRESQL»**

Вариант 10. БД «Автовокзал»

Автор: Ле Хоанг Чыонг

Факультет: ИКТ

Группа: К32392

Преподаватель: Говорова М. М.

Санкт-Петербург, 2023

# **1 Описание работы**

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

**Оборудование:** компьютерный класс.

**Программное обеспечение:** СУБД PostgreSQL , 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.

## **2 Описание предметной области**

### **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. В билете указывается номер места в автобусе. 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. На каждый рейс формируется экипаж из двух водителей. 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.

### 3 Выполнение работы

Схема базы данных:

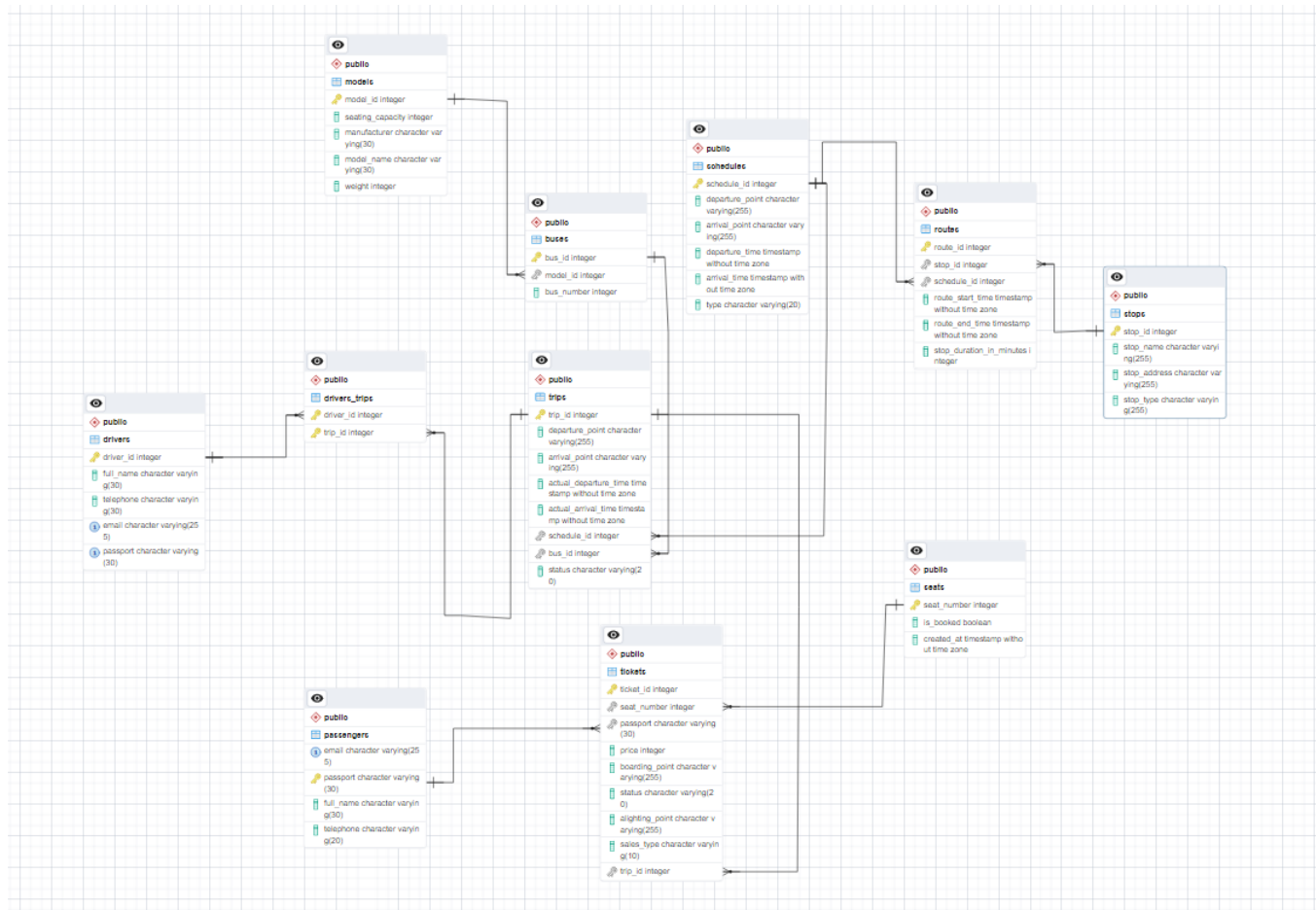


Рисунок 1 – ERD диаграмм

Запросы к базе данных:

1, Вывести фамилии водителей и номера автобусов, отправившиеся в рейсы до 12

часов текущего дня.

```
SELECT d.full_name,  
       b.bus_number  
FROM drivers_trips dt  
JOIN drivers d ON dt.driver_id = d.driver_id  
JOIN trips t ON dt.trip_id = t.trip_id  
JOIN buses b ON t.bus_id = b.bus_id  
WHERE t.actual_departure_time <  
       (SELECT NOW() - INTERVAL '12 hours');
```

	full_name character varying (30)	bus_number integer
1	John Doe	1001
2	Sophia Adams	1006

Рисунок 2 – SELECT №1

2, Рассчитать выручку от продажи билетов за прошедший день.

```
SELECT SUM(ti.price) AS total  
FROM tickets ti  
JOIN trips t ON t.trip_id = ti.trip_id  
WHERE ti.status = 'paid'  
AND CAST(t.actual_arrival_time AS DATE) =  
CAST(NOW() - INTERVAL '1 day' AS DATE);
```

	total bigint
1	600000

Рисунок 3 – SELECT №2

3, Вывести список водителей, которые не выполнили ни одного рейса за прошедший день.

```
SELECT d.driver_id,
```

```

        d.full_name,
        d.telephone,
        d.email,
        d.passport
FROM drivers d
LEFT JOIN drivers_trips dt ON d.driver_id =
dt.driver_id
LEFT JOIN trips t ON dt.trip_id = t.trip_id
AND DATE(t.actual_departure_time) = DATE(NOW() -
INTERVAL '1 DAY')
WHERE t.trip_id IS NULL;

```

	driver_id [PK] integer	full_name character varying (30)	telephone character varying (30)	email character varying (255)	passport character varying (30)
1	1	John Doe	1234567890	john.doe@example.com	AB123456
2	2	Jane Smith	1234567891	jane.smith@example.com	AB123457
3	3	Michael Johnson	1234567892	michael.johnson@example.com	AB123458
4	4	Emily Brown	1234567893	emily.brown@example.com	AB123459
5	5	Sarah Williams	1234567894	sarah.williams@example.com	AB123460
6	6	David Jones	1234567895	david.jones@example.com	AB123461
7	7	Thomas Taylor	1234567896	thomas.taylor@example.com	AB123462

Рисунок 4 – SELECT №3

4, Вывести сумму убытков из-за непроданных мест в автобусе за прошедшую неделю.

```

SELECT SUM((m.seating_capacity - t.num_tickets_sold) *
price_per_seat) AS total_loss
FROM
    (SELECT t.trip_id,
        b.model_id,
        COUNT(ti.ticket_id) AS num_tickets_sold,
        ti.price AS price_per_seat
    FROM trips t
    JOIN buses b ON t.bus_id = b.bus_id
    JOIN tickets ti ON t.trip_id = ti.trip_id
    JOIN models m ON m.model_id = b.model_id
    WHERE t.actual_departure_time >= CURRENT_DATE -

```

```
INTERVAL '1 week'

GROUP BY t.trip_id,
         b.model_id,
         ti.price) AS t
JOIN models m ON t.model_id = m.model_id;
```

	total_loss numeric
1	123750000

Рисунок 5 – SELECT №4

5, Сколько рейсов выполнил каждый водитель за последний месяц.

```
SELECT d.full_name,
       COUNT(t.trip_id) AS trips_count
FROM drivers d
JOIN drivers_trips dt ON dt.driver_id = d.driver_id
JOIN trips t ON dt.trip_id = t.trip_id
WHERE t.actual_departure_time >= DATE_TRUNC('month',
NOW() - INTERVAL '1 month')
GROUP BY d.full_name;
```

	full_name character varying (30)	trips_count bigint
1	Emily Brown	1
2	Jane Smith	1
3	John Doe	1
4	Sarah Williams	1
5	Sophia Adams	1

Рисунок 6 – SELECT №5

6, Вывести тип автобуса, который используется на всех рейсах.

```

SELECT m.manufacturer
FROM trips t
JOIN buses b ON t.bus_id = b.bus_id
JOIN models m ON b.model_id = m.model_id
GROUP BY m.manufacturer
HAVING COUNT(DISTINCT t.trip_id) = COUNT(*);

```


	manufacturer character varying (30) 
1	Iveco
2	MAN
3	Mercedes-Benz
4	Scania
5	Volvo

Рисунок 7 – SELECT №6

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

```

SELECT d.full_name,
       SUM(EXTRACT(EPOCH
                   FROM (t.actual_arrival_time
                        - t.actual_departure_time))) AS
total_travel_time
FROM drivers d
JOIN drivers_trips dt ON dt.driver_id =
d.driver_id
JOIN trips t ON dt.trip_id = t.trip_id
WHERE t.actual_departure_time >=
DATE_TRUNC('week', NOW() - INTERVAL '1 week')
GROUP BY d.full_name
HAVING SUM(EXTRACT(EPOCH
                   FROM (t.actual_arrival_time
                        - t.actual_departure_time))) =
(SELECT MAX(total_travel_time)
 FROM

```



```

        (SELECT SUM(EXTRACT(EPOCH
FROM
(t.actual_arrival_time -
t.actual_departure_time))) AS
total_travel_time
FROM drivers d
JOIN drivers_trips dt ON d.driver_id =
dt.driver_id
JOIN trips t ON dt.trip_id = t.trip_id
GROUP BY d.full_name) AS
total_travel_time);

```

	full_name character varying (30) 🔒	total_travel_time double precision 🔒
1	Jane Smith	108600

Рисунок 8 – SELECT №7

## Создать представление для пассажиров

1, Количество свободных мест на все рейсы на завтра

```

CREATE VIEW available_seats_tomorrow AS
SELECT trips.trip_id,
        buses.model_id,
        trips.departure_point,
        trips.arrival_point,
        (models.seating_capacity) -
COUNT(tickets.ticket_id) AS available_seats
FROM trips
JOIN buses ON trips.bus_id = buses.bus_id
JOIN schedules ON trips.schedule_id =
schedules.schedule_id
JOIN seats ON buses.model_id = seats.seat_number
JOIN tickets ON trips.trip_id = tickets.trip_id

```

```

JOIN models ON models.model_id = buses.model_id
WHERE DATE(schedules.departure_time) > CURRENT_DATE
AND DATE(schedules.departure_time) < (CURRENT_DATE
+ INTERVAL '1 days')::DATE
GROUP BY trips.trip_id,
         buses.model_id,
         trips.departure_point,
         trips.arrival_point,
         models.seating_capacity
HAVING (models.seating_capacity) -
COUNT(tickets.ticket_id) > 0;

```

	trip_id integer	model_id integer	departure_point character varying (255)	arrival_point character varying (255)	available_seats bigint
1	3	1	New York Central	Los Angeles Central	49

Рисунок 9 – VIEW №1

2, Самый популярный маршрут этой зимой.

```

CREATE OR REPLACE VIEW
most_popular_winter_route AS
SELECT r.route_id,
       s.departure_point,
       s.arrival_point,
       COUNT(t.ticket_id) AS total_tickets
FROM public.routes r
JOIN public.schedules s ON r.schedule_id =
s.schedule_id
JOIN public.tickets t ON t.trip_id IN
(SELECT trip_id
 FROM public.trips
 WHERE actual_departure_time >= '2023-12-01'
 AND actual_departure_time < '2024-03-01')
GROUP BY r.route_id,
         s.departure_point,
         s.arrival_point

```

```
ORDER BY total_tickets DESC  
LIMIT 1;
```

## INSERT

1, Добавить новые рейсы для водителей, у которых в настоящее время нет рейсов и у которых наименьшее количество выполненных рейсов.

```
INSERT INTO public.trips (departure_point,  
arrival_point, actual_departure_time,  
actual_arrival_time, schedule_id, bus_id,  
status)  
SELECT 'New York Central',  
       'Los Angeles Central',  
       '2023-06-01 08:00:00',  
       '2023-06-01 16:00:00',  
       sub2.schedule_id,  
       1,  
       'Pending'  
FROM  
  (SELECT d.driver_id  
   FROM public.drivers dell  
   LEFT JOIN public.drivers_trips dt ON  
d.driver_id = dt.driver_id  
   LEFT JOIN public.trips TO Ndt.trip_id =  
t.trip_id  
   WHERE t.trip_id IS NULL  
   GROUP BY d.driver_id  
   ORDER BY COUNT(dt.trip_id)  
   LIMIT 1) AS sub  
JOIN  
  (SELECT s.schedule_id,  
         b.bus_id  
   FROM public.schedules s,
```

```

public.buses b
WHERE s.departure_point = 'New York
Central'
AND s.arrival_point = 'Los Angeles
Central'
ORDER BY s.departure_time ASC
LIMIT 1) AS sub2 ON TRUE RETURNING trip_id;

```

	trip_id [PK] integer
1	8

	trip_id [PK] integer	departure_point character varying (255)	arrival_point character varying (255)	actual_departure_time timestamp without time zone	actual_arrival_time timestamp without time zone
1	3	New York Central	Los Angeles Central	2023-05-05 08:10:00	2023-05-05 16:45:00
2	4	Philadelphia Central	San Antonio Central	2023-05-15 09:10:00	2023-05-15 15:35:00
3	5	Chicago Central	Houston Central	2023-05-07 10:15:00	2023-05-08 16:25:00
4	6	Houston Central	Phoenix Central	2023-05-10 08:25:00	2023-05-10 14:50:00
5	7	Phoenix Central	Philadelphia Central	2023-05-12 08:30:00	2023-05-12 14:15:00
6	8	New York Central	Los Angeles Central	2023-06-01 08:00:00	2023-06-01 16:00:00

Рисунок 10 – INSERT №1

## UPDATE:

1, Замените водителя этого рейса другим водителем, у которого в настоящее время нет рейсов и у которого наименьшее количество выполненных рейсов.

```

UPDATE drivers_trips
SET driver_id = (
SELECT d.driver_id
FROM drivers d
LEFT JOIN drivers_trips dt ON d.driver_id =
dt.driver_id
LEFT JOIN trips t ON dt.trip_id = t.trip_id
WHERE t.trip_id IS NULL
AND d.driver_id <> 3
GROUP BY d.driver_id
ORDER BY COUNT(dt.trip_id)
LIMIT 1

```

```
WHERE drivers_trips.driver_id = 3;
```

До:

	trip_id integer	departure_point character varying (255)	arrival_point character varying (255)	actual_departure_time timestamp without time zone	actual_arrival_time timestamp without time zone	driver_id integer
1	3	New York Central	Los Angeles Central	2023-05-05 08:10:00	2023-05-05 16:45:00	1
2	5	Chicago Central	Houston Central	2023-05-07 10:15:00	2023-05-08 16:25:00	2
3	4	Philadelphia Central	San Antonio Central	2023-05-15 09:10:00	2023-05-15 15:35:00	3
4	7	Phoenix Central	Philadelphia Central	2023-05-12 08:30:00	2023-05-12 14:15:00	4
5	6	Houston Central	Phoenix Central	2023-05-10 08:25:00	2023-05-10 14:50:00	5

После

	trip_id integer	departure_point character varying (255)	arrival_point character varying (255)	actual_departure_time timestamp without time zone	actual_arrival_time timestamp without time zone	driver_id integer
1	3	New York Central	Los Angeles Central	2023-05-05 08:10:00	2023-05-05 16:45:00	1
2	5	Chicago Central	Houston Central	2023-05-07 10:15:00	2023-05-08 16:25:00	2
3	7	Phoenix Central	Philadelphia Central	2023-05-12 08:30:00	2023-05-12 14:15:00	4
4	6	Houston Central	Phoenix Central	2023-05-10 08:25:00	2023-05-10 14:50:00	5
5	4	Philadelphia Central	San Antonio Central	2023-05-15 09:10:00	2023-05-15 15:35:00	10

Рисунок 11 – UPDATE №1

## DELETE:

1, Удалить билет определенного пассажира, когда он хочет отменить рейс

```
DELETE
```

```
FROM tickets t
```

```
WHERE t.passport = 'P123456'
```

```
AND trip_id = 3;
```

До:

	ticket_id [PK] integer	seat_number integer	passport character varying (30)	price integer
1	2	1	P123456	500000
2	3	2	P123457	550000
3	4	3	P123458	600000
4	5	4	P123459	650000
5	6	5	P123460	700000
6	7	6	P123461	750000

После

	ticket_id [PK] integer	seat_number integer	passport character varying (30)	price integer
1	3	2	P123457	550000
2	4	3	P123458	600000
3	5	4	P123459	650000
4	6	5	P123460	700000
5	7	6	P123461	750000
6	8	7	P123462	800000

Рисунок 12 – DELETE №1

## Создание индексов

Без индексов:

```
SELECT t.trip_id,  
       d. full_name,  
       b.bus_number,  
       m. model_name,  
       s.departure_point,  
       s.arrival_point,  
       s.departure_time,  
       s.arrival_time  
FROM trips TO  
JOIN buses b ON t.bus_id = b.bus_id  
JOIN models m ON b.model_id = m.model_id  
JOIN schedules s ON t.schedule_id =  
s.schedule_id  
JOIN drivers_trips dt ON t.trip_id = dt.trip_id  
JOIN drivers d ON dt.driver_id = d.driver_id;
```

	QUERY PLAN text	
1	Hash Join (cost=5.98..21.69 rows=5 width=1212) (actual time=0.111..0.114 rows=5 loops=1)	
2	Hash Cond: (dt.driver_id = d.driver_id)	
3	-> Hash Join (cost=4.76..20.45 rows=5 width=1138) (actual time=0.080..0.083 rows=5 loops=1)	
4	Hash Cond: (m.model_id = b.model_id)	
5	-> Seq Scan on models m (cost=0.00..14.10 rows=410 width=82) (actual time=0.006..0.007 rows=10 loops=1)	
6	-> Hash (cost=4.70..4.70 rows=5 width=1064) (actual time=0.068..0.069 rows=5 loops=1)	
7	Buckets: 1024 Batches: 1 Memory Usage: 9kB	
8	-> Hash Join (cost=3.50..4.70 rows=5 width=1064) (actual time=0.063..0.066 rows=5 loops=1)	
9	Hash Cond: (t.schedule_id = s.schedule_id)	
10	-> Hash Join (cost=2.27..3.46 rows=5 width=20) (actual time=0.046..0.049 rows=5 loops=1)	
11	Hash Cond: (b.bus_id = t.bus_id)	
12	-> Seq Scan on buses b (cost=0.00..1.10 rows=10 width=12) (actual time=0.005..0.006 rows=10 loops=1)	
13	-> Hash (cost=2.21..2.21 rows=5 width=16) (actual time=0.033..0.034 rows=5 loops=1)	
14	Buckets: 1024 Batches: 1 Memory Usage: 9kB	
15	-> Hash Join (cost=1.14..2.21 rows=5 width=16) (actual time=0.030..0.031 rows=5 loops=1)	
16	Hash Cond: (dt.trip_id = t.trip_id)	
17	-> Seq Scan on drivers_trips dt (cost=0.00..1.05 rows=5 width=8) (actual time=0.013..0.013 rows=5 loops=1)	
18	-> Hash (cost=1.06..1.06 rows=6 width=12) (actual time=0.010..0.010 rows=6 loops=1)	
19	Buckets: 1024 Batches: 1 Memory Usage: 9kB	
20	-> Seq Scan on trips t (cost=0.00..1.06 rows=6 width=12) (actual time=0.006..0.007 rows=6 loops=1)	
21	-> Hash (cost=1.10..1.10 rows=10 width=1052) (actual time=0.011..0.011 rows=10 loops=1)	
22	Buckets: 1024 Batches: 1 Memory Usage: 9kB	
23	-> Seq Scan on schedules s (cost=0.00..1.10 rows=10 width=1052) (actual time=0.006..0.008 rows=10 loops=1)	
24	-> Hash (cost=1.10..1.10 rows=10 width=82) (actual time=0.018..0.018 rows=10 loops=1)	
25	Buckets: 1024 Batches: 1 Memory Usage: 9kB	
26	-> Seq Scan on drivers d (cost=0.00..1.10 rows=10 width=82) (actual time=0.013..0.014 rows=10 loops=1)	
27	Planning Time: 0.424 ms	
28	Execution Time: 0.168 ms	
Total rows: 28 of 28    Query complete 00:00:00.084		

Создание индексов:

```
CREATE INDEX trips_bus_id_idx ON trips (bus_id);
CREATE INDEX buses_model_id_idx ON
buses (model_id);
CREATE INDEX schedules_schedule_id_idx ON
schedules (schedule_id);
CREATE INDEX drivers_trips_driver_id_idx ON
drivers_trips (driver_id);
CREATE INDEX drivers_driver_id_idx ON
drivers (driver_id);
```

## С индексами:

	QUERY PLAN	
	text	
1	Hash Join (cost=5.98..21.69 rows=5 width=1212) (actual time=0.108..0.111 rows=5 loops=1)	
2	Hash Cond: (dt.driver_id = d.driver_id)	
3	-> Hash Join (cost=4.76..20.45 rows=5 width=1138) (actual time=0.075..0.078 rows=5 loops=1)	
4	Hash Cond: (m.model_id = b.model_id)	
5	-> Seq Scan on models m (cost=0.00..14.10 rows=410 width=82) (actual time=0.006..0.007 rows=10 loops=1)	
6	-> Hash (cost=4.70..4.70 rows=5 width=1064) (actual time=0.062..0.063 rows=5 loops=1)	
7	Buckets: 1024 Batches: 1 Memory Usage: 9kB	
8	-> Hash Join (cost=3.50..4.70 rows=5 width=1064) (actual time=0.057..0.060 rows=5 loops=1)	
9	Hash Cond: (t.schedule_id = s.schedule_id)	
10	-> Hash Join (cost=2.27..3.46 rows=5 width=20) (actual time=0.040..0.042 rows=5 loops=1)	
11	Hash Cond: (b.bus_id = t.bus_id)	
12	-> Seq Scan on buses b (cost=0.00..1.10 rows=10 width=12) (actual time=0.006..0.006 rows=10 loops=1)	
13	-> Hash (cost=2.21..2.21 rows=5 width=16) (actual time=0.026..0.026 rows=5 loops=1)	
14	Buckets: 1024 Batches: 1 Memory Usage: 9kB	
15	-> Hash Join (cost=1.14..2.21 rows=5 width=16) (actual time=0.022..0.024 rows=5 loops=1)	
16	Hash Cond: (dt.trip_id = t.trip_id)	
17	-> Seq Scan on drivers_trips dt (cost=0.00..1.05 rows=5 width=8) (actual time=0.006..0.006 rows=5 loops=1)	
18	-> Hash (cost=1.06..1.06 rows=6 width=12) (actual time=0.009..0.009 rows=6 loops=1)	
19	Buckets: 1024 Batches: 1 Memory Usage: 9kB	
20	-> Seq Scan on trips t (cost=0.00..1.06 rows=6 width=12) (actual time=0.006..0.007 rows=6 loops=1)	
21	-> Hash (cost=1.10..1.10 rows=10 width=1052) (actual time=0.011..0.011 rows=10 loops=1)	
22	Buckets: 1024 Batches: 1 Memory Usage: 9kB	
23	-> Seq Scan on schedules s (cost=0.00..1.10 rows=10 width=1052) (actual time=0.007..0.008 rows=10 loops=1)	
24	-> Hash (cost=1.10..1.10 rows=10 width=82) (actual time=0.019..0.019 rows=10 loops=1)	
25	Buckets: 1024 Batches: 1 Memory Usage: 9kB	
26	-> Seq Scan on drivers d (cost=0.00..1.10 rows=10 width=82) (actual time=0.013..0.015 rows=10 loops=1)	
27	Planning Time: 0.551 ms	
28	Execution Time: 0.162 ms	
Total rows: 28 of 28    Query complete 00:00:00.040		



После создания индексов запросы выполняются быстрее.

Можно увидеть некоторые преимущества индекса:

Ускорение запросов: индексы ускоряют поиск данных, тем самым ускоряя запросы.

Уменьшите нагрузку на систему: когда запрос выполняется с индексом, система может быстрее извлекать данные, что снижает нагрузку на систему.

Некоторые ограничения индексов в SQL включают:

Увеличьте размер базы данных: создание индекса может увеличить размер базы данных.

Уменьшите частоту обновления: при добавлении, редактировании или удалении записей индекс должен быть обновлен, чтобы отразить изменение. Это может снизить скорость обновления базы данных.

**Вывод:**

Благодаря этому лабораторному занятию я узнал некоторые знания sql об общих запросах и о том, как оптимизировать время запроса с помощью индексов, я обнаружил, что это действительно полезное знание, особенно для программистов.