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

Отчет

по лабораторной работе №4 «Создание таблиц базы данных PostgreSQL. Заполнение таблиц рабочими данными»

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

Автор: Тюленев А.С.

Факультет: ИКТ

Группа: К3240

Преподаватель: Говорова М.М.

Санкт-Петербург 2023

Оглавление

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.

1. Создать запросы:

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

```
SELECT
  t.id AS train_id,
  t.departure_time,
  w.id AS wagon_id,
  s.id AS seat_id,
  s.seat_type,
  s.occupancy_status
FROM
  train t
  JOIN wagon w ON t.id = w.trip
  JOIN seats s ON w.id = s.wagon
  JOIN station st ON t.stdepid = st.id
WHERE
  st.station_name = 'Название вокзала'
  AND t.departure_time BETWEEN NOW() AND NOW() + INTERVAL '1 day'
  AND s.occupancy_status = 'Свободно';
```

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

```
SELECT
  p.full_name,
  tr.departure_time,
  tr.arrival_time
FROM
  passenger p
  JOIN ticket tk ON p.id = tk.passenger_id
  JOIN trip tr ON tk.id = tr.ticket_id
  JOIN station s ON tr.dest_stationid = s.id
WHERE
  s.station_name = 'Москва'
  AND tr.departure_time BETWEEN NOW() - INTERVAL '1 day' AND NOW();
```

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

```
SELECT
  tr.id AS train_id
FROM
  train tr
  JOIN trip tp ON tr.id = tp.train
  JOIN wagon w ON tr.id = w.trip
  JOIN seats s ON w.id = s.wagon
  JOIN ticket tk ON s.id = tk.seat_id
WHERE
  tp.departure_date BETWEEN CURRENT_DATE + INTERVAL '1 day' AND CURRENT_DATE +
INTERVAL '2 days'
GROUP BY
  tr.id
HAVING
  COUNT(DISTINCT s.id) = COUNT(tk.id);
```

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

```
SELECT
  tr.id AS train_id,
```

```
  w.wagon_number,
  s.id AS seat_id,
  s.seat_type
FROM
  train tr
  INNER JOIN trip tp ON tr.id = tp.train
  INNER JOIN wagon w ON tr.id = w.trip
  INNER JOIN wtype wt ON w.wtype = wt.id
  INNER JOIN seats s ON w.id = s.wagon
  LEFT JOIN ticket tk ON s.id = tk.seat_id AND tk.sale_date = CURRENT_DATE
  INNER JOIN station st ON tp.dest_stationid = st.id
WHERE
  st.station_name = 'Москва'
  AND wt.wagon_name = 'Купе'
  AND tp.departure_date = CURRENT_DATE
  AND tk.id IS NULL;
```

Выручка от продажи билетов на все поезда за прошедшие сутки.

```
SELECT
  SUM(tk.ticket_price) AS total_revenue
FROM
  ticket tk
WHERE
  tk.sale_date BETWEEN CURRENT_DATE - INTERVAL '1 day' AND CURRENT_DATE;
```

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

```
SELECT
  COUNT(*) AS total_sv_tickets
FROM
  ticket tk
  JOIN seats s ON tk.seat_id = s.id
  JOIN wagon w ON s.wagon = w.id
  JOIN wtype wt ON w.wtype = wt.id
WHERE
  wt.wagon_name = 'СВ';
```

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

```
SELECT
  tr.id AS train_id,
  tr.train_name
FROM
  train tr
  JOIN wagon w ON tr.id = w.trip
  JOIN wtype wt ON w.wtype = wt.id
  LEFT JOIN (
    SELECT
      s.wagon,
      COUNT(tk.id) AS tickets_sold
    FROM
      seats s
      LEFT JOIN ticket tk ON s.id = tk.seat_id AND tk.sale_date BETWEEN
CURRENT_DATE - INTERVAL '1 day' AND CURRENT_DATE
    GROUP BY s.wagon
  ) AS ts ON w.id = ts.wagon
WHERE
  ts.tickets_sold <= (wt.seats_total / 2)
GROUP BY
```

```
  tr.id,
  tr.train_name
HAVING
  COUNT(w.id) = COUNT(ts.wagon);

Представления:

Для пассажиров о наличии свободных мест на заданный рейс.
CREATE VIEW available_seats_view AS
SELECT
  tr.id AS train_id,
  tr.train_name,
  w.wagon_number,
  wt.wagon_name,
  s.id AS seat_id,
  s.seat_type,
  CASE
    WHEN tk.id IS NULL THEN 'Свободно'
    ELSE 'Занято'
  END AS occupancy_status
FROM
  train tr
  JOIN wagon w ON tr.id = w.trip
  JOIN wtype wt ON w.wtype = wt.id
  JOIN seats s ON w.id = s.wagon
  LEFT JOIN ticket tk ON s.id = tk.seat_id;
```

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

```
CREATE VIEW unsold_tickets_view AS
SELECT
  tr.id AS train_id,
  wt.wagon_name AS wagon_type,
  COUNT(*) - COUNT(tk.id) AS unsold_tickets_count
FROM
  train tr
  JOIN trip tp ON tr.id = tp.train
  JOIN wagon w ON tr.id = w.trip
  JOIN wtype wt ON w.wtype = wt.id
  JOIN seats s ON w.id = s.wagon
  LEFT JOIN ticket tk ON s.id = tk.seat_id AND tk.sale_date BETWEEN CURRENT_DATE -
INTERVAL '1 day' AND CURRENT_DATE
WHERE
  tp.departure_date BETWEEN CURRENT_DATE - INTERVAL '1 day' AND CURRENT_DATE
GROUP BY
  tr.id, wt.wagon_name;
```

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

```
INSERT INTO ticket (passenger_id, seat_id, ticket_price, sale_date, ticket_status,
dep_stationid, dest_stationid)
SELECT
  1,
  (SELECT s.id FROM seats s
   INNER JOIN wagon w ON s.wagon = w.id
   INNER JOIN wtype wt ON w.wtype = wt.id
```

```
    LEFT JOIN ticket tk ON s.id = tk.seat_id AND tk.sale_date = CURRENT_DATE
    WHERE wt.wagon_name = 'Купе'
      AND tk.id IS NULL
    LIMIT 1),
  5000,
  CURRENT_DATE,
  'Выкуплен',
  1,
  2
WHERE
  EXISTS (
    SELECT s.id FROM seats s
    INNER JOIN wagon w ON s.wagon = w.id
    INNER JOIN wtype wt ON w.wtype = wt.id
    LEFT JOIN ticket tk ON s.id = tk.seat_id AND tk.sale_date = CURRENT_DATE
    WHERE wt.wagon_name = 'Купе'
      AND tk.id IS NULL
    LIMIT 1
  );
```

Query | Query History

```
1  select * from ticket;
```

Data Output | Messages | Notifications

	id [PK] integer	passenger integer	seats integer	ticket_price integer	sale_date date	ticket_status character varying	tickettype integer	seat_id integer	passenger_id integer	dep_station integer
1	1	2	1	5000	2023-06-15	Выкуплен	1	10	201	
2	2	1	2	3000	2023-06-20	Забронирован	2	11	202	
3	3	3	1	8000	2023-07-01	Отменён	2	12	203	
4	4	1	3	5000	2023-11-21	Выкуплен	1	3	1	

Query | Query History

```
1  select * from ticket;
```

Data Output | Messages | Notifications

	id [PK] integer	passenger integer	seats integer	ticket_price integer	sale_date date	ticket_status character varying	tickettype integer	seat_id integer	passenger_id integer	dep_station integer
1	1	2	1	5000	2023-06-15	Выкуплен	1	10	201	
2	2	1	2	3000	2023-06-20	Забронирован	2	11	202	
3	3	3	1	8000	2023-07-01	Отменён	2	12	203	
4	4	1	3	5000	2023-11-21	Выкуплен	1	3	1	

UPDATE

```
UPDATE ticket
SET ticket_status = 'Отменен'
WHERE id IN (
  SELECT tk.id FROM ticket tk
  INNER JOIN stnumber sn ON sn.id = tk.dest_stationid
  JOIN trip tr ON sn.trip_id = tr.id
  WHERE tr.status = 'Отменен'
);
```

Query | Query History

```
1  select * from ticket;
```

Data Output | Messages | Notifications

	id [PK] integer	passenger integer	seats integer	ticket_price integer	sale_date date	ticket_status character varying	tickettype integer	seat_id integer	passenger_id integer	dep_stationid integer	dest_stationid integer
1	2	1	2	3000	2023-06-20	Забронирован	2	11	202	2	2
2	3	3	1	8000	2023-07-01	Отменён	2	12	203	3	3
3	4	1	3	5000	2023-11-21	Выкуплен	1	3	1	1	2
4	1	2	1	5000	2023-06-15	Отменен	1	10	201	1	1

DELETE

```sql
DELETE FROM ticket
WHERE dep_stationid = (
  SELECT id FROM station
  WHERE station_name = 'Станция закрыта на ремонт'
);
```

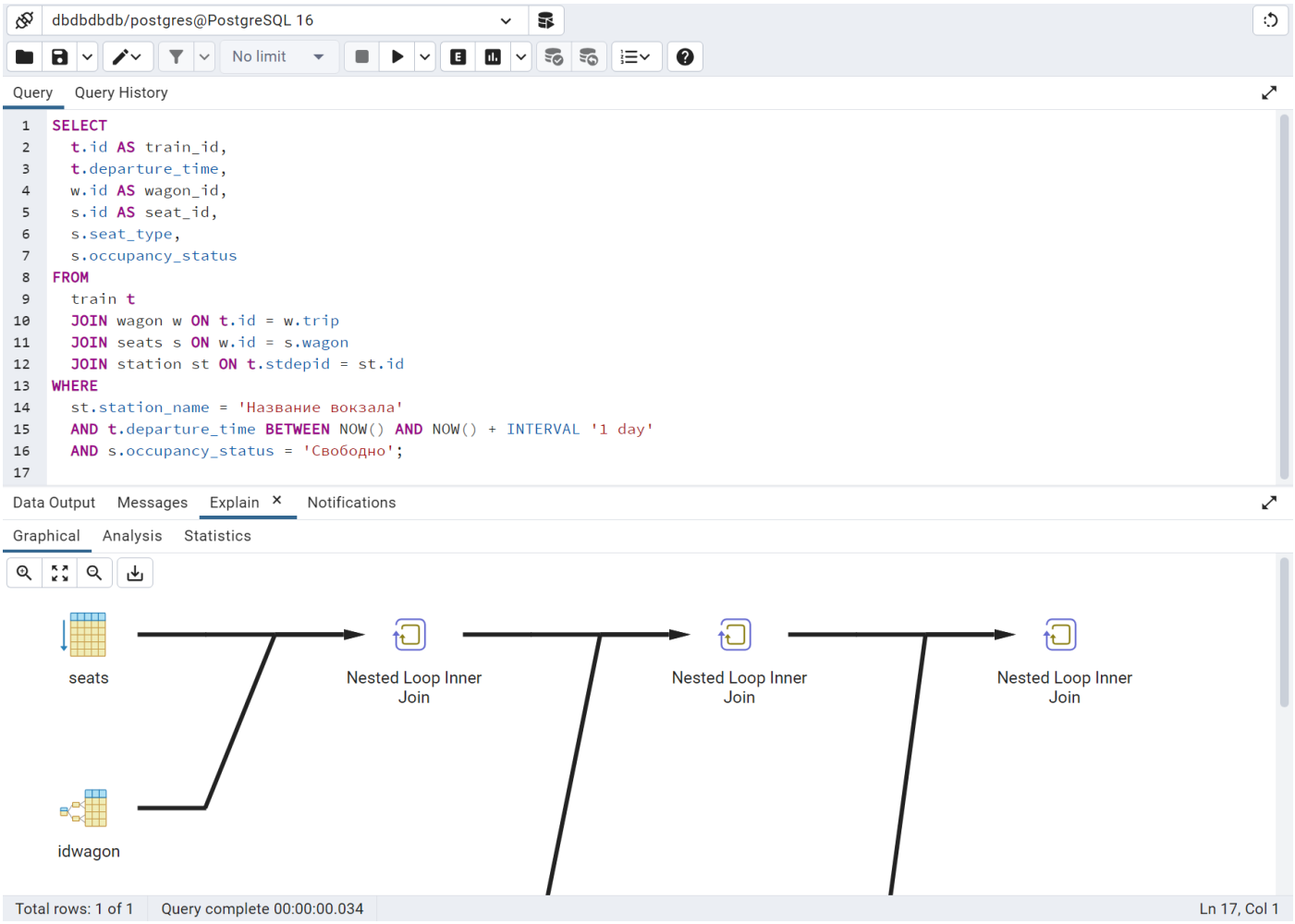
Query | Query History

```sql
1  select * from ticket;
```

Data Output | Messages | Notifications

	id [PK] integer	passenger integer	seats integer	ticket_price integer	sale_date date	ticket_status character varying	tickettype integer	seat_id integer	passenger_id integer	dep_stationid integer	dest_stationid integer
1	2	1	2	3000	2023-06-20	Забронирован	2	11	202	2	2
2	3	3	1	8000	2023-07-01	Отменён	2	12	203	3	3
3	4	1	3	5000	2023-11-21	Выкуплен	1	3	1	1	2
4	1	2	1	5000	2023-06-15	Отменен	1	10	201	1	1

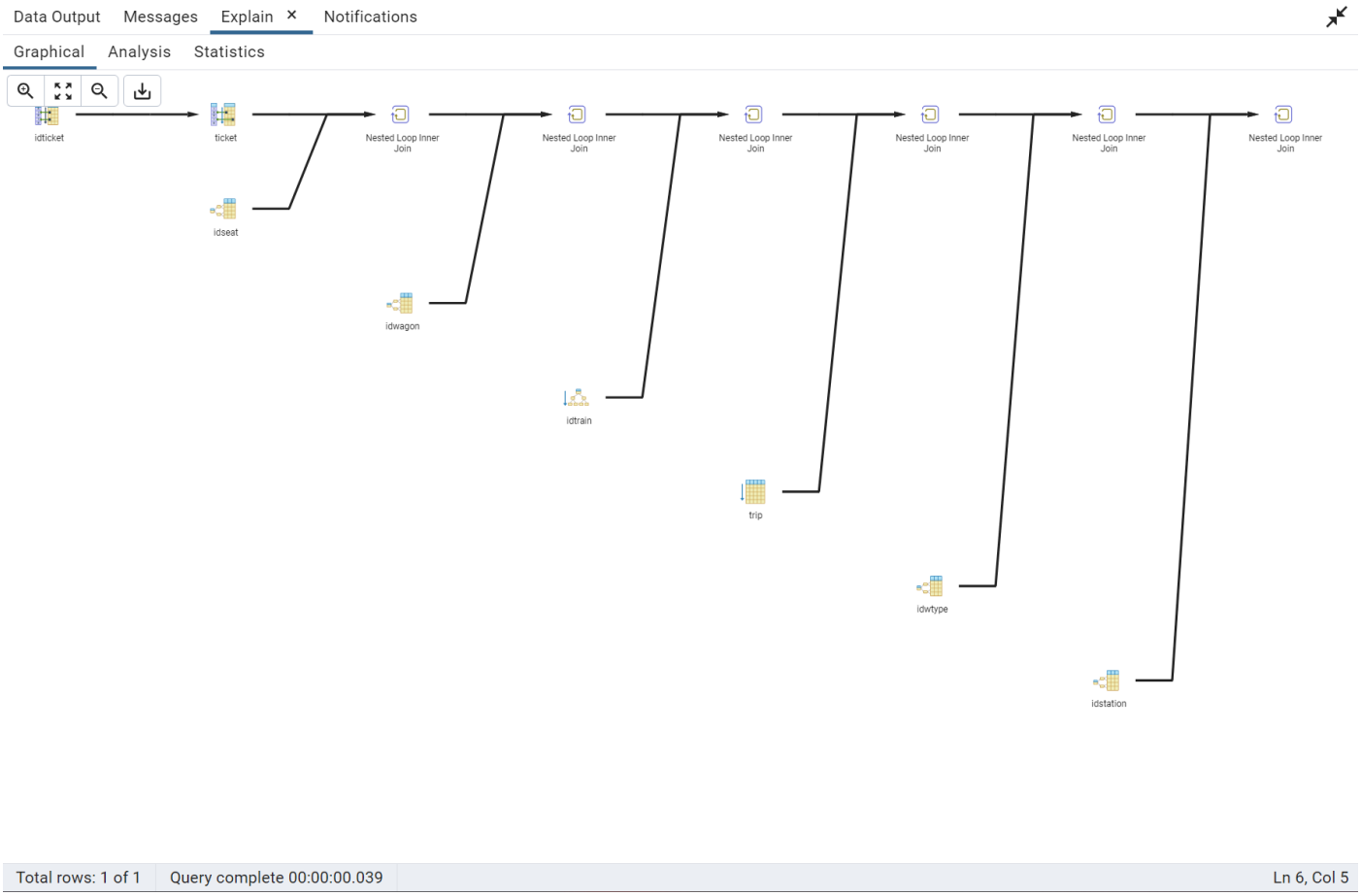
3. Изучить графическое представление запросов и просмотреть историю запросов.



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.

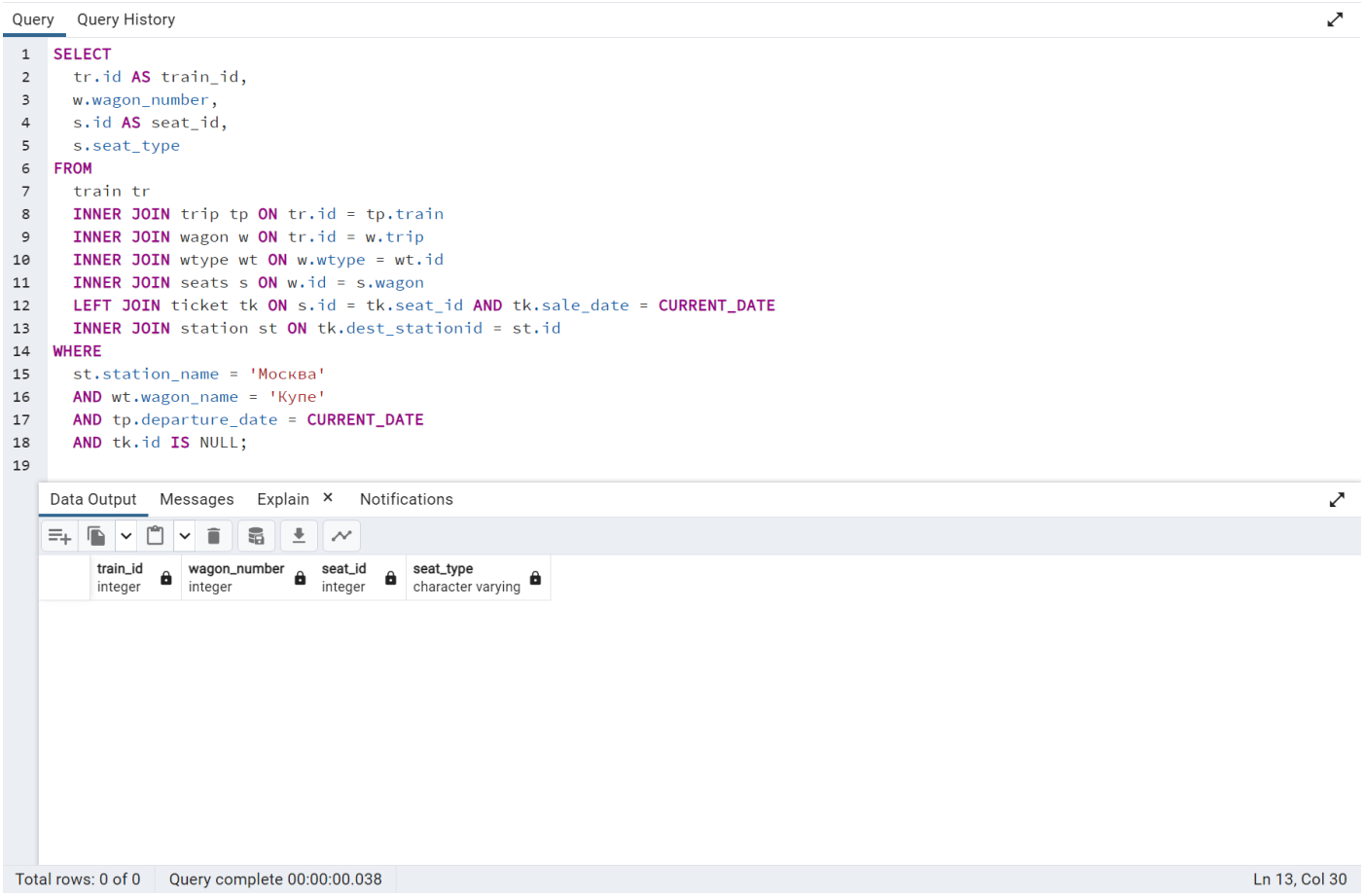
Запрос №1. Свободные места в купейные вагоны всех рейсов до Москвы на текущие сутки.

```
SELECT
  tr.id AS train_id,
  w.wagon_number,
  s.id AS seat_id,
  s.seat_type
FROM
  train tr
  INNER JOIN trip tp ON tr.id = tp.train
  INNER JOIN wagon w ON tr.id = w.trip
  INNER JOIN wtype wt ON w.wtype = wt.id
  INNER JOIN seats s ON w.id = s.wagon
  LEFT JOIN ticket tk ON s.id = tk.seat_id AND tk.sale_date = CURRENT_DATE
  INNER JOIN station st ON tp.dest_stationid = st.id
WHERE
  st.station_name = 'Москва'
  AND wt.wagon_name = 'Купе'
  AND tp.departure_date = CURRENT_DATE
 AND tk.id IS NULL;
```



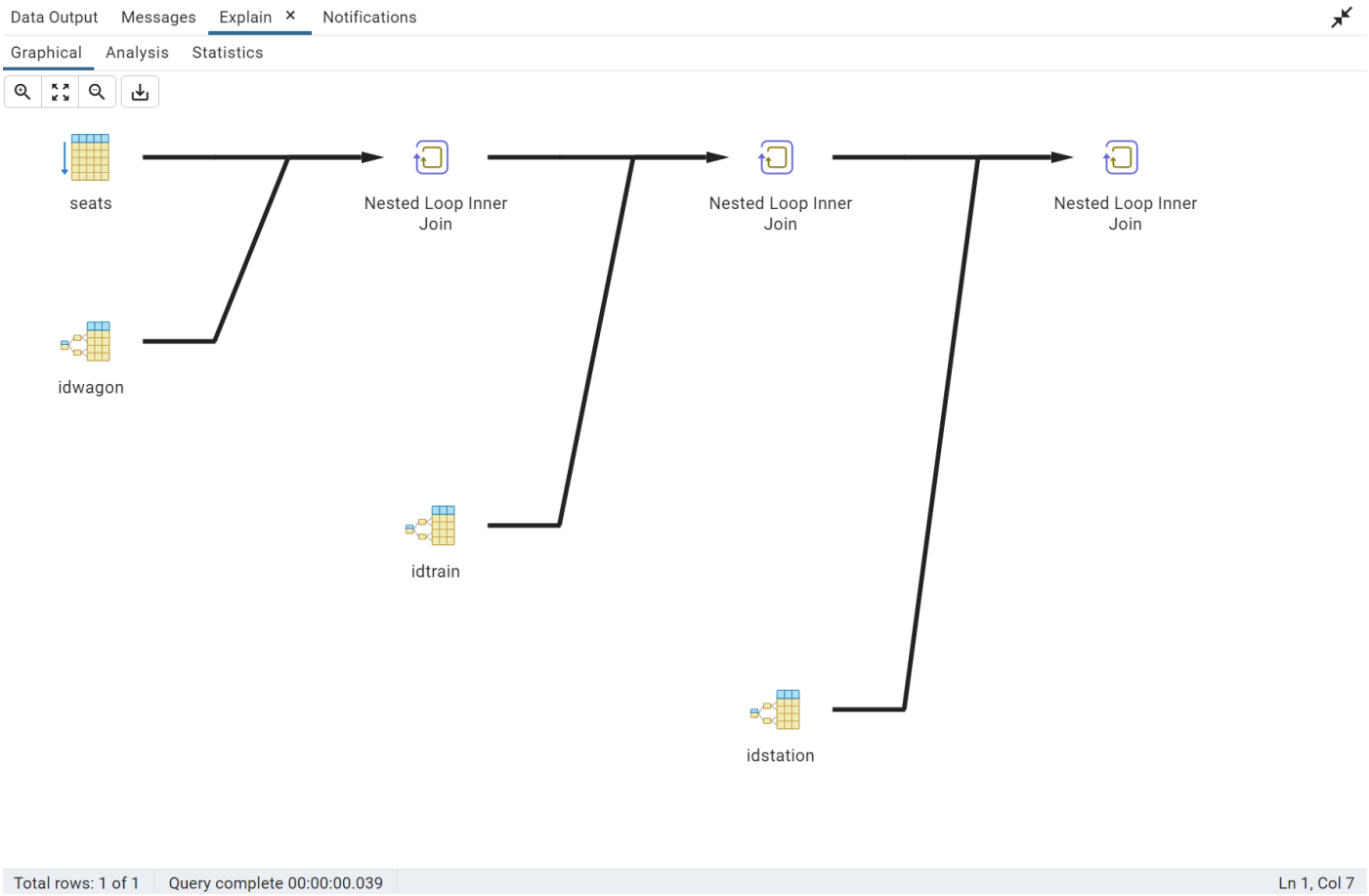
Индекс:

```
CREATE INDEX idx_station_name ON station(station_name);
```



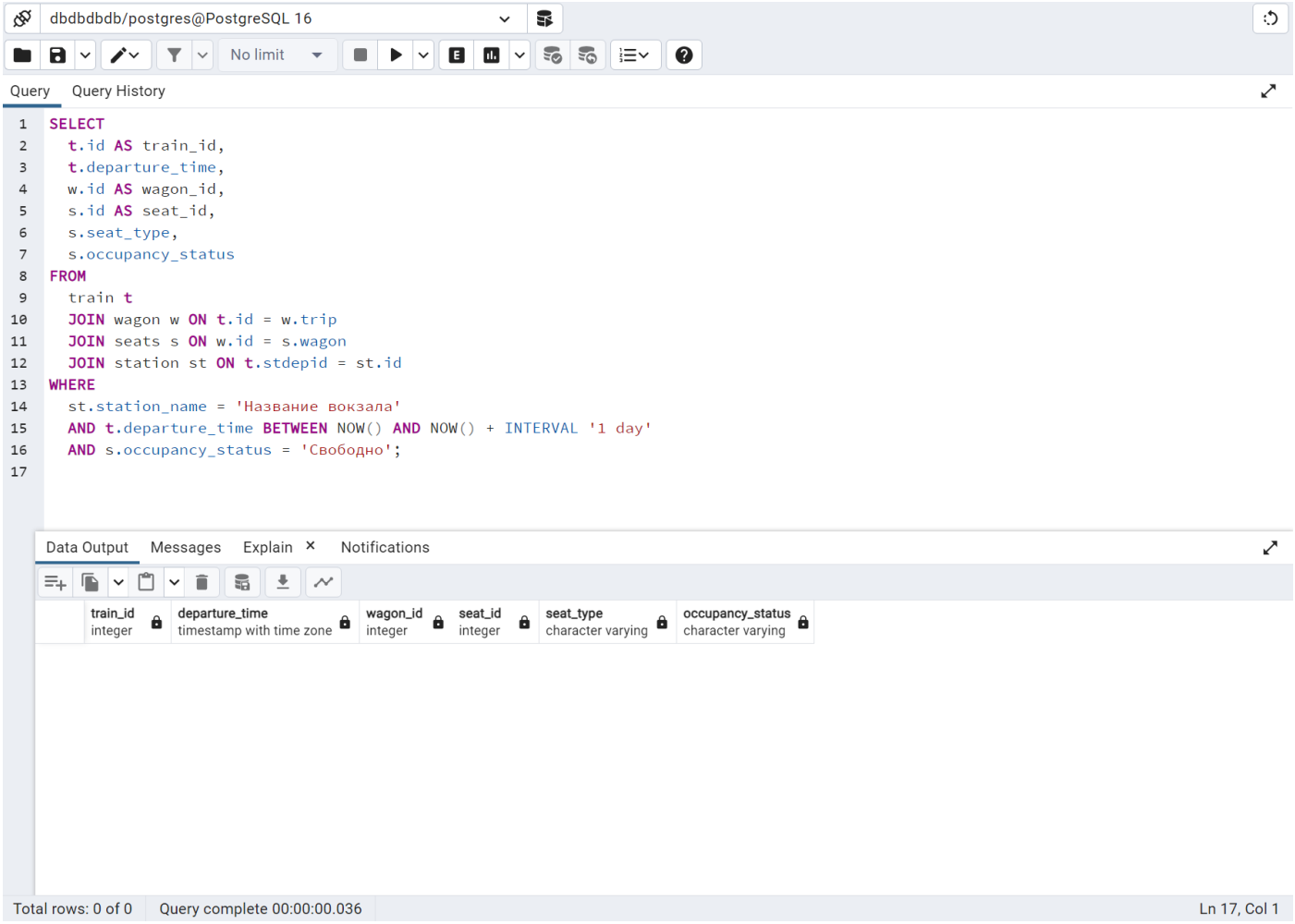
Запрос №2. Свободные места на все поезда, отправляющиеся с вокзала в течение следующих суток.

```
SELECT
  t.id AS train_id,
  t.departure_time,
  w.id AS wagon_id,
  s.id AS seat_id,
  s.seat_type,
  s.occupancy_status
FROM
  train t
  JOIN wagon w ON t.id = w.trip
  JOIN seats s ON w.id = s.wagon
  JOIN station st ON t.stdepid = st.id
WHERE
  st.station_name = 'Название вокзала'
  AND t.departure_time BETWEEN NOW() AND NOW() + INTERVAL '1 day'
  AND s.occupancy_status = 'Свободно';
```



Индекс:

```
CREATE INDEX idx_train_station_seats ON train(departure_time) INCLUDE (id);
CREATE INDEX idx_seats_occupancy ON seats(occupancy_status) INCLUDE (wagon);
```


dbdbdbdb/postgres@PostgreSQL 16
No limit
Query
Query History
SELECT
t.id AS train_id,
t.departure_time,
w.id AS wagon_id,
s.id AS seat_id,
s.seat_type,
s.occupancy_status
FROM
train t
JOIN wagon w ON t.id = w.trip
JOIN seats s ON w.id = s.wagon
JOIN station st ON t.stdepid = st.id
WHERE
st.station_name = 'Название вокзала'
AND t.departure_time BETWEEN NOW() AND NOW() + INTERVAL '1 day'
AND s.occupancy_status = 'Свободно';
Data Output
Messages
Explain
Notifications
train_id integer
departure_time timestamp with time zone
wagon_id integer
seat_id integer
seat_type character varying
occupancy_status character varying
Total rows: 0 of 0
Query complete 00:00:00.036
Ln 17, Col 1

Вывод

Во время лабораторной работы я научился выполнять разнообразные SQL-запросы к базе данных, а также создавать представления и индексы. Кроме того, я сравнил скорость выполнения SELECT-запросов до и после создания индексов. Безусловно, использование индексов привело к сокращению времени выполнения запросов (однако стоит отметить, что не к существенному, так как данных достаточно мало).