

**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 .....	Ошибка! Закладка не определена.
Практическое задание .....	Ошибка! Закладка не определена.
Вариант 6. БД «Пассажир» .....	Ошибка! Закладка не определена.
Рисунок 1 – Схема логической модели базы данных. ....	Ошибка! Закладка не определена.
Листинг дампа .....	Ошибка! Закладка не определена.
Вывод.....	4

**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 = 'Москва'
    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	



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



	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

<

DELETE

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

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		3	3	1	8000	2023-07-01	Отменён	2	12	203	3
3		4	1	3	5000	2023-11-21	Выкуплен	1	3	1	2
4		1	2	1	5000	2023-06-15	Отменен	1	10	201	1

### 3. Изучить графическое представление запросов и просмотреть историю запросов.

The screenshot shows a PostgreSQL query editor interface. The top bar indicates the connection is 'dbdbdbdb/postgres@PostgreSQL 16'. Below the toolbar, the 'Query' tab is active, displaying a SQL query. The 'Query History' tab is also visible. The 'Data Output' tab is active, showing the 'Explain' view of the query. The 'Graphical' tab is selected, displaying a visual execution plan. The plan shows a sequence of operations: a 'seats' table scan, followed by three 'Nested Loop Inner Join' operations. The status bar at the bottom indicates 'Total rows: 1 of 1' and 'Query complete 00:00:00.034'.

```
1 SELECT
2   t.id AS train_id,
3   t.departure_time,
4   w.id AS wagon_id,
5   s.id AS seat_id,
6   s.seat_type,
7   s.occupancy_status
8 FROM
9   train t
10  JOIN wagon w ON t.id = w.trip
11  JOIN seats s ON w.id = s.wagon
12  JOIN station st ON t.stdepid = st.id
13 WHERE
14   st.station_name = 'Название вокзала'
15   AND t.departure_time BETWEEN NOW() AND NOW() + INTERVAL '1 day'
16   AND s.occupancy_status = 'Свободно';
17
```

Graphical Execution Plan:

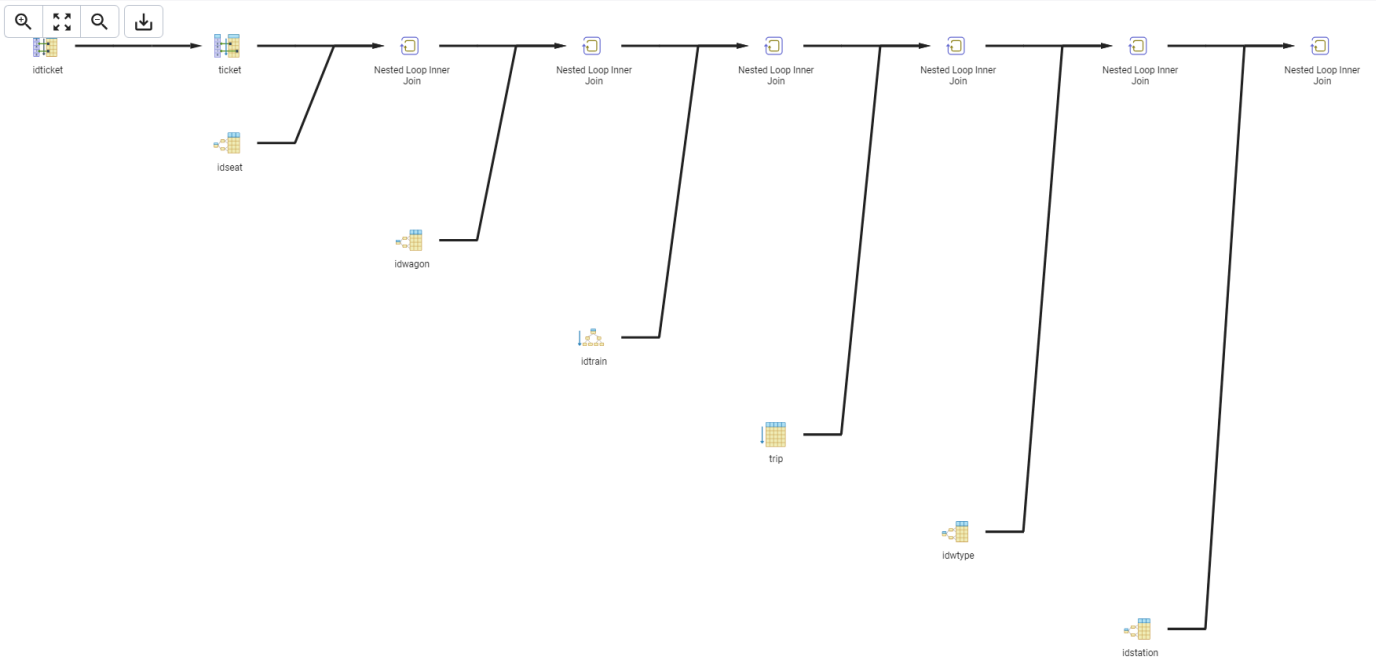
- seats
- idwagon
- Nested Loop Inner Join
- Nested Loop Inner Join
- Nested Loop Inner Join

Total rows: 1 of 1 | Query complete 00:00:00.034 | Ln 17, Col 1

### 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;
```



Total rows: 1 of 1 Query complete 00:00:00.039

Ln 6, Col 5

Индекс:

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

Query Query History

```

1 SELECT
2     tr.id AS train_id,
3     w.wagon_number,
4     s.id AS seat_id,
5     s.seat_type
6 FROM
7     train tr
8     INNER JOIN trip tp ON tr.id = tp.train
9     INNER JOIN wagon w ON tr.id = w.trip
10    INNER JOIN wtype wt ON w.wtype = wt.id
11    INNER JOIN seats s ON w.id = s.wagon
12    LEFT JOIN ticket tk ON s.id = tk.seat_id AND tk.sale_date = CURRENT_DATE
13    INNER JOIN station st ON tk.dest_stationid = st.id
14 WHERE
15     st.station_name = 'Москва'
16     AND wt.wagon_name = 'Kyne'
17     AND tp.departure_date = CURRENT_DATE
18     AND tk.id IS NULL;
19

```

Data Output Messages Explain X Notifications

train\_id

integer

wagon\_number

integer

seat\_id

integer

seat\_type

character varying

Total rows: 0 of 0

Query complete 00:00:00.038

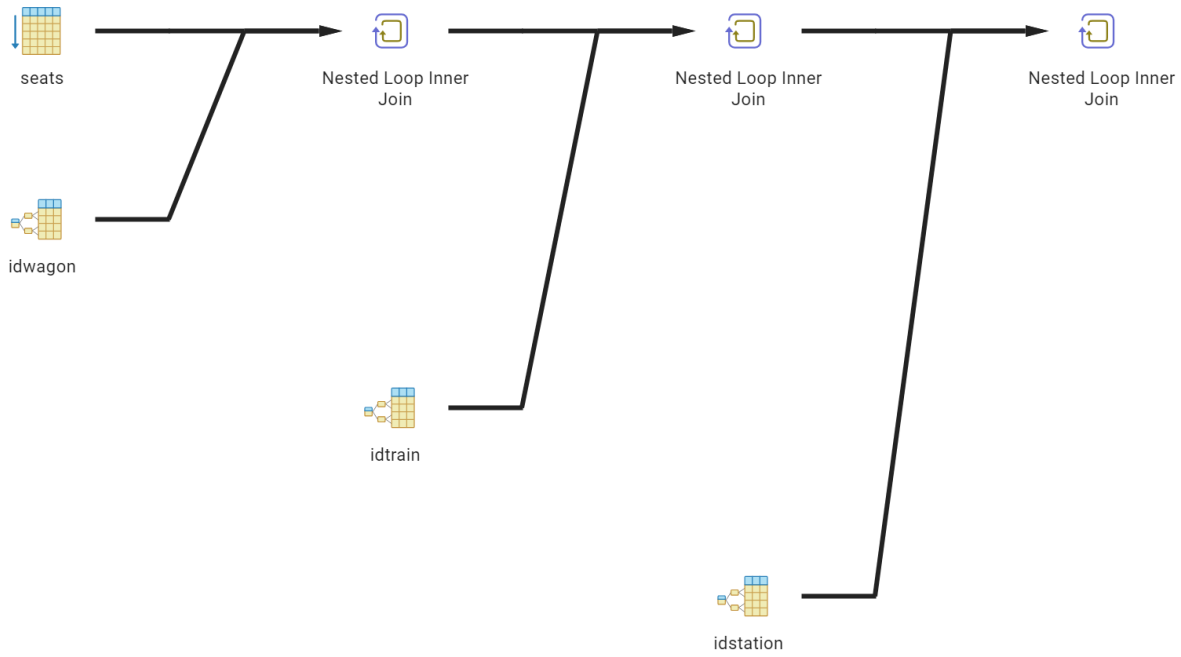
Ln 13, Col 30

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 = 'Москва'
    AND t.departure_time BETWEEN NOW() AND NOW() + INTERVAL '1 day'
    AND s.occupancy_status = 'Свободно';

```



Total rows: 1 of 1 Query complete 00:00:00.039

Ln 1, Col 7

## Индекс:

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

dbdbddb/postgres@PostgreSQL 16

No limit

E

Query

Query History

```
1 SELECT
2   t.id AS train_id,
3   t.departure_time,
4   w.id AS wagon_id,
5   s.id AS seat_id,
6   s.seat_type,
7   s.occupancy_status
8 FROM
9   train t
10 JOIN wagon w ON t.id = w.trip
11 JOIN seats s ON w.id = s.wagon
12 JOIN station st ON t.stdepid = st.id
13 WHERE
14   st.station_name = 'Название вокзала'
15   AND t.departure_time BETWEEN NOW() AND NOW() + INTERVAL '1 day'
16   AND s.occupancy_status = 'Свободно';
17
```

Data Output

Messages

Explain x

Notifications

train_id	departure_time	wagon_id	seat_id	seat_type	occupancy_status
integer	timestamp with time zone	integer	integer	character varying	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).