

**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»

Автор: Русинов В.А.

Факультет: ИКТ

Группа: К3240

Преподаватель: Говорова М.М.



Санкт-Петербург 2023

## Оглавление

Цель работы и практическое задание .....	3
1. Запросы к БД .....	4
2. Создание представлений .....	9
3. Запросы на модификацию данных .....	12
Вывод .....	16

## **Цель работы и практическое задание**

**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. Запросы к БД

- Определить расчетное время полета по всем маршрутам. Листинг:

```
SELECT
```

```
    id_route,
```

```
    ROUND(((EXTRACT(EPOCH FROM s.time_arrival) - EXTRACT (EPOCH FROM  
s.time_departure))/3600),2) AS flight_time
```

```
FROM schema.route r
```

```
JOIN schema.schedule s ON r.id_schedule = s.id_schedule;
```

The screenshot shows a database query tool interface. At the top, there are tabs for 'Query' and 'Query History'. The 'Query' tab is active, displaying the following SQL query:

```
1 SELECT
2     id_route,
3     ROUND(((EXTRACT(EPOCH FROM s.time_arrival) - EXTRACT (EPOCH FROM s.time
4 FROM schema.route r
5 JOIN schema.schedule s ON r.id_schedule = s.id_schedule;
6
7
```

Below the query editor, there are tabs for 'Data Output', 'Messages', and 'Notifications'. The 'Data Output' tab is active, showing a table with the results of the query. The table has two columns: 'id\_route' (integer, primary key) and 'flight\_time' (numeric). The results are as follows:

	id_route [PK] integer	flight_time numeric
1	5	1.50
2	1	1.50
3	4	3.50
4	2	3.50
5	6	2.00
6	3	2.00

- Определить расход топлива по всем маршрутам. Листинг:

```
SELECT r.id_route,
```

```
    p.fuel_rate AS fuel_rate_per_hour,
```

```
    ROUND((EXTRACT(EPOCH FROM s.time_arrival) - EXTRACT (EPOCH FROM  
s.time_departure))/3600 * p.fuel_rate,0) AS fuel_consumption
```

```
FROM schema.route r
```

```
JOIN schema.schedule s ON r.id_schedule = s.id_schedule
```

```
JOIN schema.plane pl ON r.id_plane = pl.id_plane
```

```
JOIN schema.model p ON pl.id_model = p.id_model;
```

Query Query History

```

1 SELECT r.id_route,
2       p.fuel_rate AS fuel_rate_per_hour,
3       ROUND((EXTRACT(EPOCH FROM s.time_arrival) - EXTRACT (EPOCH FRC
4 FROM schema.route r
5 JOIN schema.schedule s ON r.id_schedule = s.id_schedule
6 JOIN schema.plane pl ON r.id_plane = pl.id_plane
7 JOIN schema.model p ON pl.id_model = p.id_model;
8
9

```

Data Output Messages Notifications

	id_route integer	fuel_rate_per_hour integer	fuel_consumption numeric
1	5	500	750
2	1	500	750
3	4	500	1750
4	2	600	2100
5	6	500	1000
6	3	600	1200

- Вывести данные о том, сколько свободных мест оставалось в самолетах, совершавших полет по заданному маршруту за вчерашний день. Листинг:

```
SELECT r.id_route,
```

```
       s.id_seat,
```

```
       s.number,
```

```
       s.row,
```

```
       s.status
```

```
FROM schema.route r
```

```
JOIN schema.seat s ON r.id_route = s.id_route
```

```
WHERE DATE(r.date_departure) = CURRENT_DATE - INTERVAL '1 day' and s.status =
'Available' AND r.id_schedule = 2;
```

Query

Query History

1

2

3

4

5

6

7

8

9

10

```
SELECT r.id_route,
       s.id_seat,
       s.number,
       s.row,
       s.status
FROM schema.route r
JOIN schema.seat s ON r.id_route = s.id_route
WHERE DATE(r.date_departure) = CURRENT_DATE - INTERVAL '1 day' and s.st
```

Data Output

Messages

Notifications

	id_route integer	id_seat integer	number integer	row integer	status character varying (20)
1	4	7	1	1	Available
2	4	5	1	2	Available

- Рассчитать убытки компании за счет непроданных билетов за вчерашний день.

SELECT COUNT(t.id\_ticket) AS total\_not\_sold\_tickets,

SUM(t.price) AS losses\_due\_to\_unsold\_tickets

FROM schema.ticket t

JOIN schema.seat s ON t.id\_seat = s.id\_seat

JOIN schema.route r ON t.id\_route = r.id\_route

WHERE DATE(r.date\_departure) = CURRENT\_DATE - INTERVAL '1 day' and t.status = 'Available';

Query	Query History
1	<b>SELECT</b> COUNT(t.id_ticket) <b>AS</b> total_not_sold_tickets,
2	SUM(t.price) <b>AS</b> losses_due_to_unsold_tickets
3	<b>FROM</b> schema.ticket t
4	<b>JOIN</b> schema.seat s <b>ON</b> t.id_seat = s.id_seat
5	<b>JOIN</b> schema.route r <b>ON</b> t.id_route = r.id_route
6	<b>WHERE</b> DATE(r.date_departure) = <b>CURRENT_DATE</b> - INTERVAL '1 day

  

Data Output	Messages	Notifications						
<div> <div>+</div> <div>📄</div> <div>▼</div> <div>📋</div> <div>▼</div> <div>🗑️</div> <div>🔍</div> <div>⬇️</div> <div>📈</div> </div> <table> <tr> <th></th><th>total_not_sold_tickets bigint</th><th>losses_due_to_unsold_tickets real</th></tr> <tr> <td>1</td><td>2</td><td>250</td></tr> </table>		total_not_sold_tickets bigint	losses_due_to_unsold_tickets real	1	2	250		
	total_not_sold_tickets bigint	losses_due_to_unsold_tickets real						
1	2	250						

- Определить, какой тип самолетов чаще всего летал в заданный аэропорт назначения.  
Листинг:

```
SELECT * FROM (  
  
  SELECT m.type_of_plane as type_of_plane,  
  
         COUNT(*) AS flight_count  
  
  FROM schema.route r  
  
  JOIN schema.schedule sc ON r.id_schedule = sc.id_schedule  
  
  JOIN schema.plane p ON r.id_plane = p.id_plane  
  
  JOIN schema.model m ON p.id_model = m.id_model  
  
  WHERE sc.id_airport_arrival = 1  
  
  GROUP BY type_of_plane  
  
  ORDER BY flight_count DESC  
  
)  
  
WHERE flight_count in (SELECT MAX(flight_count) FROM (  
  
  SELECT m.type_of_plane as type_of_plane,  
  
         COUNT(*) AS flight_count  
  
  FROM schema.route r  
  
  JOIN schema.schedule sc ON r.id_schedule = sc.id_schedule  
  
  JOIN schema.plane p ON r.id_plane = p.id_plane  
  
  JOIN schema.model m ON p.id_model = m.id_model  
  
  WHERE sc.id_airport_arrival = 1  
  
  GROUP BY type_of_plane  
  
  ORDER BY flight_count DESC  
  
))
```

Query
Query History

```

1  SELECT * FROM (
2  SELECT m.type_of_plane as type_of_plane,
3         COUNT(*) AS flight_count
4  FROM schema.route r
5  JOIN schema.schedule sc ON r.id_schedule = sc.id_schedule
6  JOIN schema.plane p ON r.id_plane = p.id_plane
7  JOIN schema.model m ON p.id_model = m.id_model
8  WHERE sc.id_airport_arrival = 1
9  GROUP BY type_of_plane
10 ORDER BY flight_count DESC
11 )
12 WHERE flight_count in (SELECT MAX(flight_count) FROM (
13 SELECT m.type_of_plane as type_of_plane,
14        COUNT(*) AS flight_count
15 FROM schema.route r
16 JOIN schema.schedule sc ON r.id_schedule = sc.id_schedule
17 JOIN schema.plane p ON r.id_plane = p.id_plane
18 JOIN schema.model m ON p.id_model = m.id_model
19 WHERE sc.id_airport_arrival = 1
20 GROUP BY type_of_plane
21 ORDER BY flight_count DESC
22 ))

```

Data Output
Messages
Notifications

+

📄

▼

📋

▼

🗑️

🔄

⬇️

📈

type\_of\_plane

character varying (20)

flight\_count

bigint

1	Narrow-body	3
---	-------------	---

- Вывести список самолетов, “возраст” которых превышает средний “возраст” самолетов этого типа. Листинг:

```

SELECT *
FROM (
SELECT p.id_plane,
       p.tail_number,
       AGE(CURRENT_DATE, p.date_last_repair) AS plane_age,
       AVG(AGE(CURRENT_DATE, p.date_last_repair)) OVER (PARTITION BY
m.type_of_plane) AS avg_age
FROM schema.plane p
JOIN schema.model m ON p.id_model = m.id_model
)

```



WHERE plane\_age > avg\_age

Query Query History

```
1 SELECT *
2 FROM (
3 SELECT p.id_plane,
4        p.tail_number,
5        AGE(CURRENT_DATE, p.date_last_repair) AS plane_age,
6        AVG(AGE(CURRENT_DATE, p.date_last_repair)) OVER (PARTITION BY m.
7
8 FROM schema.plane p
9 JOIN schema.model m ON p.id_model = m.id_model
10 )
```

Data Output Messages Notifications

	id_plane [PK] integer	tail_number character varying (20)	plane_age interval	avg_age interval
1	3	CC003	2 years 1 mon 11 days	1 year 5 mons 25 days

## 2. Создание представлений

- для пассажиров авиакомпании о рейсах в Москву на ближайшую неделю; Листинг:

```
CREATE VIEW schema.passenger_flights_to_moscow AS SELECT
```

```
DISTINCT
```

```
    r.id_route,
```

```
    r.date_departure,
```

```
    r.date_arrival
```

```
FROM
```

```
    schema.route r JOIN schema.schedule s ON
```

```
    r.id_schedule = s.id_schedule JOIN
```

```
    schema.airport a ON s.id_airport_arrival = a.id_airport
```

```
WHERE
```

```
    a.city = 'Moscow' AND
```

```
    DATE(r.date_departure) BETWEEN CURRENT_DATE AND CURRENT_DATE +  
    INTERVAL '7 days';
```

Query Query History

```

1 CREATE VIEW schema.passenger_flights_to_moscow AS
2 SELECT DISTINCT
3     r.id_route,
4     r.date_departure,
5     r.date_arrival
6 FROM
7     schema.route r
8 JOIN
9     schema.schedule s ON r.id_schedule = s.id_schedule
10 JOIN
11     schema.airport a ON s.id_airport_arrival = a.id_airport
12 WHERE
13     a.city = 'Moscow' AND
14     DATE(r.date_departure) BETWEEN CURRENT_DATE AND CURRENT_DATE + INTERVAL '7 days'

```

Data Output Messages Notifications

CREATE VIEW

Query returned successfully in 76 msec.

Query Query History

```

1 SELECT * FROM schema.passenger_flights_to_moscow
2

```

Data Output Messages Notifications

	id_route integer	date_departure date	date_arrival date
1	5	2023-11-13	2023-11-13
2	6	2023-11-16	2023-11-16

- количество самолетов каждого типа, летавшими за последний месяц. Листинг:

```
CREATE VIEW schema.aircraft_count_by_type AS
```

```
SELECT
```

```
    m.type_of_plane,
```

```
    COUNT(r.id_plane) AS airplane_count
```

```
FROM
```

```
    schema.route r
```

JOIN schema.plane p ON p.id\_plane =

r.id\_plane JOIN schema.model m ON

p.id\_model = m.id\_model

WHERE

DATE(r.date\_departure) BETWEEN CURRENT\_DATE - INTERVAL '1 month' AND  
CURRENT\_DATE

GROUP BY

m.type\_of\_plane;

Query Query History

```
1 CREATE VIEW schema.aircraft_count_by_type AS
2 SELECT
3     m.type_of_plane,
4     COUNT(r.id_plane) AS airplane_count
5 FROM
6     schema.route r
7 JOIN
8     schema.plane p ON p.id_plane = r.id_plane
9 JOIN
10    schema.model m ON p.id_model = m.id_model
11 WHERE
12     DATE(r.date_departure) BETWEEN CURRENT_DATE - INTERVAL '1 month' AND CI
13 GROUP BY
14     m.type_of_plane;
```

Data Output Messages Notifications

CREATE VIEW

Query returned successfully in 478 msec.

Query Query History

```
1 SELECT * FROM schema.aircraft_count_by_type
```

Data Output Messages Notifications

	type_of_plane character varying (20)	airplane_count bigint
1	Narrow-body	2
2	Wide-body	2

### 3. Запросы на модификацию данных

- INSERT – вставить в таблицу с кассами кассы, находящиеся в аэропортах

Листинг:

```
INSERT INTO schema.ticket_office(country, city, address)
SELECT country, city, name
FROM schema.airport;
```

Query

Query History

1

INSERT INTO schema.ticket\_office(country, city, address)

2

SELECT country, city, name

3

FROM schema.airport;

Data Output

Messages

Notifications

INSERT 0 3

Query returned successfully in 74 msec.

	id_ticket_office [PK] integer	city character varying (20)	address character varying (50)	country character varying (20)
1	1	City 1	Address 1	Country 1
2	2	City 2	Address 2	Country 2
3	3	City 3	Address 3	Country 3
4	4	Moscow	Airport 1	Country 1
5	5	City 2	Airport 2	Country 2
6	6	City 3	Airport 3	Country 3

- UPDATE – обновить дату последнего ремонта на сегодняшнюю для всех самолетов типа wide-body

Листинг:

```
UPDATE schema.plane
SET date_last_repair = CURRENT_DATE
WHERE id_model IN (SELECT id_model FROM schema.model WHERE type_of_plane = 'Wide-body');
```

Query	Query History
<pre>1 UPDATE schema.plane 2 SET date_last_repair = CURRENT_DATE 3 WHERE id_model IN (SELECT id_model FROM schema.model WHERE type_of_plane = 'Wide-body');</pre>	

Data Output	Messages	Notifications
UPDATE 1		
Query returned successfully in 165 msec.		

	id_plane [PK] integer	tail_number character varying (20)	flight_hours integer	date_last_repair date	id_model integer	id_company integer
1	1	AA001	1000	2023-01-01	1	1
2	2	BB002	2000	2023-11-10	2	2
3	3	CC003	500	2021-09-29	1	2

- **DELETE** – удалить все самолеты, принадлежащие компаниям определенной страны  
Листинг:

**DELETE FROM** schema.plane

**WHERE** id\_company IN (SELECT id\_company FROM schema.company WHERE country = 'Country 3');

Query
Query History

```

1 DELETE FROM schema.plane
2 WHERE id_company IN (SELECT id_company FROM schema.company WHERE country = 'Country 3');
3

```

Data Output
Messages
Notifications

DELETE 1

Query returned successfully in 178 msec.

#### 4. Создание индексов Листинг:

```
CREATE INDEX idx_plane ON schema.route (id_plane);
```

```
CREATE INDEX idx_fuel ON schema.plane (flight_hours, id_model);
```

Без индексов:

The screenshot shows a database query interface with a 'Query' tab selected. The query is as follows:

```
1 SELECT (EXTRACT(EPOCH FROM s.time_arrival) - EXTRACT (EPOCH FROM
2 FROM schema.route r
3 JOIN schema.schedule s ON r.id_schedule = s.id_schedule;
```

Below the query editor, there are tabs for 'Data Output', 'Messages', and 'Notifications'. The 'Messages' tab is active, displaying the following status:

Successfully run. Total query runtime: 98 msec.  
6 rows affected.

The screenshot shows a database query interface with a 'Query' tab selected. The query is as follows:

```
1 SELECT r.id_route,
2         p.fuel_rate AS fuel_rate_per_hour,
3         (EXTRACT(EPOCH FROM s.time_arrival) - EXTRA
4 FROM schema.route r
5 JOIN schema.schedule s ON r.id_schedule = s.id_sch
6 JOIN schema.plane pl ON r.id_plane = pl.id_plane
7 JOIN schema.model p ON pl.id_model = p.id_model;
```

Below the query editor, there are tabs for 'Data Output', 'Messages', and 'Notifications'. The 'Messages' tab is active, displaying the following status:

Successfully run. Total query runtime: 116 msec.  
6 rows affected.

С индексами:

Query Query History

```
1 SELECT (EXTRACT(EPOCH FROM s.time_arrival) - EXTRACT (EPOCH F
2 FROM schema.route r
3 JOIN schema.schedule s ON r.id_schedule = s.id_schedule;
```

Data Output Messages Notifications

Successfully run. Total query runtime: 80 msec.  
6 rows affected.

Query Query History

```
1 SELECT r.id_route,
2         p.fuel_rate AS fuel_rate_per_hour,
3         (EXTRACT(EPOCH FROM s.time_arrival) - EXTRACT (EPOCH
4 FROM schema.route r
5 JOIN schema.schedule s ON r.id_schedule = s.id_schedule
6 JOIN schema.plane pl ON r.id_plane = pl.id_plane
7 JOIN schema.model p ON pl.id_model = p.id_model;
8
```

Data Output Messages Notifications

Successfully run. Total query runtime: 79 msec.  
6 rows affected.

## **Вывод**

В ходе лабораторной работы я освоил работу с различными SQL-запросами к базе данных, также создание представлений и индексов. Также сравнил время работы SELECT запросов с индексами и без. Разумеется, с индексами время меньше.