

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

**Отчет**

по лабораторной работе «Запросы на выборку и модификацию данных, представления и индексы в PostgreSQL»

по дисциплине «Проектирование и реализация баз данных»

Автор: Коротин А.М.

Факультет: ИКТ

Группа: К32391

Преподаватель: Говорова М.М.



Санкт-Петербург 2023

**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.

#### Вариант 14

**Задание 2.** Создать запросы:

- Вывести данные о водителе, который чаще всех доставляет пассажиров на заданную улицу.
- Вывести данные об автомобилях, которые имеют пробег более 250 тысяч километров и которые не проходили ТО в текущем году.
- Сколько раз каждый пассажир воспользовался услугами таксопарка?
- Вывести данные пассажира, который воспользовался услугами таксопарка максимальное число раз.
- Вывести данные о водителе, который ездит на самом дорогом автомобиле.
- Вывести данные пассажира, который всегда ездит с одним и тем же водителем.
- Какие автомобили имеют пробег больше среднего пробега для своей марки.

**Задание 3.** Создать представление:

- содержащее сведения о незанятых на данный момент водителях;
- зарплата всех водителей за вчерашний день.

**Ход работы:**

**Задание 2:**

1. Вывести данные о водителе, который чаще всех доставляет пассажиров на заданную улицу.

Query Query History

```

1 SELECT w.first_name, w.last_name
2 FROM worker w
3 WHERE w.id IN
4     (SELECT o.worker_id
5      FROM ordr o
6      WHERE o.address_to = 'MCF'
7      GROUP BY o.worker_id
8      HAVING COUNT(*) = (SELECT MAX(drive_count) FROM
9                          (SELECT COUNT(*) AS drive_count FROM ordr
10                           WHERE ordr.address_to = 'MCF'
11                            GROUP BY ordr.worker_id) as _));

```

Data Output Messages Notifications

	first_name character varying (50)	last_name character varying (50)
1	Bruh	421

2. Вывести данные об автомобилях, которые имеют пробег более 250 тысяч. километров и которые не проходили ТО в текущем году.

Query Query History

```

1 SELECT car.government_number from car
2 WHERE mileage > 250000
3 AND EXTRACT(YEAR FROM maintenance_date) < 2023;

```

Data Output Messages Notifications

	government_number character (9)
1	A018AA

3. Сколько раз каждый пассажир воспользовался услугами таксопарка?

Query

Query History

1

2

3

4

5

SELECT

u1.first\_name,

u1.last\_name,

u2.count

FROM

usr u1

JOIN

(SELECT

usr\_id,

COUNT(\*)

FROM

ordr

GROUP BY

usr\_id)

as

u2

ON

u1.id =

u2.usr\_id;

Data Output

Messages

Notifications

	<div>first_name</div> <div>character varying (50)</div>	<div>last_name</div> <div>character varying (50)</div>	<div>count</div> <div>bigint</div>
1	Вася	Пупкин	88
2	123421	412421	86
3	Alexey	4124	76

4. Вывести данные пассажира, который воспользовался услугами таксопарка максимальное число раз.

Query

Query History

```

1 select u.first_name, u.last_name
2     FROM usr u
3     WHERE u.id IN (SELECT o.usr_id
4                     FROM ordr o
5                     GROUP BY usr_id
6                     HAVING COUNT(*) = (SELECT MAX(ride_count) FROM
7                                         (SELECT COUNT(*) as ride_count
8                                          FROM ordr
9                                          GROUP BY usr_id) AS _));

```

Data Output

Messages

Notifications

+

📄

▼

📋

🗑️

🗄️

⬇️

📈

	first_name character varying (50) 🔒	last_name character varying (50) 🔒
1	Вася	Пупкин

5. Вывести данные о водителе, который ездит на самом дорогом автомобиле.

Query

Query History

```

1 SELECT w.first_name, w.last_name
2     FROM worker w
3     WHERE w.id IN (SELECT DISTINCT(o.worker_id)
4                     FROM ordr o
5                     WHERE o.car_id IN (SELECT car.id |
6                                         FROM car
7                                         WHERE car.market_price = (SELECT MAX(market_price)
8                                                                    FROM car)));

```

Data Output

Messages

Notifications

+

📄

▼

📋

🗑️

🗄️

⬇️

📈

	first_name character varying (50) 🔒	last_name character varying (50) 🔒
1	Фамилия	Имя
2	444	123
3	Bruh	421
4	Нереальная фамилия	Крутое имя

6. Вывести данные пассажира, который всегда ездит с одним и тем же водителем.

Query

Query History

1

2

3

4

5

6

SELECT

u.first\_name, u.last\_name

FROM

usr u

WHERE

u.id IN (SELECT

usr\_id

FROM

ordr

GROUP BY

usr\_id

HAVING

COUNT(DISTINCT(worker\_id)) = 1);

Data Output

Messages

Notifications

first\_name

character varying (50)

last\_name

character varying (50)

Таких пользователей нет в связи со спецификой генерации данных. Добавим тестовые данные для проверки запроса:

Query

Query History

1

insert into

usr (first\_name, last\_name)

values ('unique', 'user')

returning id;

Data Output

Messages

Notifications

id

[PK] integer

1

4

QueryQuery History

```

1 insert into odr (usr_id, worker_id, car_id, time_start_plan, address_from, address_to, payment_format, distance_km, time_start_fact, time_
2 values (4,
3 25,
4 7,
5 '2023-03-14 09:05:00',
6 'example',
7 'example to',
8 'by cash',
9 1,
10 '2023-03-14 09:05:00',
11 '2023-03-14 09:15:00');
12

```

Data OutputMessagesNotifications

INSERT 0 1

Query returned successfully in 274 msec.

Как можем видеть, запрос работает исправно

QueryQuery History

```

1 SELECT u.first_name, u.last_name
2 FROM usr u
3 WHERE u.id IN (SELECT usr_id
4 FROM odr
5 GROUP BY usr_id
6 HAVING COUNT(DISTINCT(worker_id)) = 1);

```

Data OutputMessagesNotifications

first\_name

character varying (50)

last\_name

character varying (50)

1

unique

user

## 7. Какие автомобили имеют пробег больше среднего пробега для своей марки.

QueryQuery History

```

1 SELECT car_al.government_number
2 FROM car car_al
3 WHERE car_al.mileage > (SELECT AVG(c1.mileage)
4 FROM car c1
5 JOIN car_type t1 ON c1.type_id = t1.id
6 WHERE brand = (SELECT brand
7 FROM car_type ct
8 WHERE ct.id = car_al.type_id));

```

Data OutputMessagesNotifications

government\_number

character (9)

1

A018AA

### Задание 3 Создать представление:

1. содержащее сведения о незанятых на данный момент водителях

The screenshot shows a SQL query editor with a 'Query' tab. The query is as follows:

```
1 CREATE OR REPLACE VIEW free_drivers AS
2     SELECT w.id, w.first_name, w.last_name FROM worker w
3         WHERE w.id IN (SELECT worker_id FROM order
4             WHERE time_end > NOW());
5
6
7 SELECT * FROM free_drivers;
```

Below the query editor, there is a 'Data Output' tab which is currently empty. The 'Messages' and 'Notifications' tabs are also visible.

2. зарплата всех водителей за вчерашний день

The screenshot shows a SQL query editor with a 'Query' tab. The query is as follows:

```
1 CREATE OR REPLACE VIEW ytd_salary AS
2     SELECT w.id, w.first_name, w.last_name,
3         (SELECT SUM(o.distance_km * r1.price_per_km) FROM order o
4             JOIN car c1 ON o.car_id = c1.id
5             JOIN car_type ct ON c1.id = ct.id
6             JOIN rate r1 ON ct.rate_id = r1.id
7             WHERE o.worker_id = w.id
8             -- AND DATE(o.time_end) = (SELECT CURRENT_DATE - interval '1 day')
9         )
10     AS salary FROM worker w;
11
12
13 SELECT * FROM ytd_salary;
```

Below the query editor, the 'Data Output' tab is active, showing the results of the query. The output is a table with 5 columns: id, first\_name, last\_name, and salary. The data is as follows:

	id	first_name	last_name	salary
1	23	Фамилия	Имя	133.95000000000002
2	24	444	123	162.44999999999996
3	25	Bruh	421	199.49999999999991
4	26	Нереальная фамилия	Крутое имя	190.94999999999999

Условие на вчерашний день опущено, потому что у всех водителей 0 зарплаты в этот день в связи со спецификой генерации данных



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

1) Поставить пользовательскую оценку 3 для заказов, которые были исполнены на машинах, стоимостью < 5.000.00

The screenshot shows a SQL IDE interface with a 'Query' tab selected. The query editor contains the following SQL statement:

```
1 UPDATE ordr
2   SET usr_rate = 3
3   WHERE car_id IN (SELECT car.id FROM car
4                     WHERE market_price < 5000000);
```

An 'Execute/Refresh' button with 'F5' is visible above the query. Below the query editor, the 'Data Output' tab is selected, showing the result: 'UPDATE 91' and 'Query returned successfully in 132 msec.'

2) Удалить все заказы, исполненные на Renault Logan

The screenshot shows a SQL IDE interface with a 'Query' tab selected. The query editor contains the following SQL statement:

```
1 DELETE FROM ordr
2   WHERE car_id IN (SELECT car.id
3                     FROM car
4                     WHERE car.type_id IN (SELECT ct.id
5                                           FROM car_type ct
6                                           WHERE ct.model = 'Logan' AND
7                                             ct.brand = 'Renault'));
```

Below the query editor, the 'Data Output' tab is selected, showing the result: 'DELETE 160' and 'Query returned successfully in 132 msec.'

3) Пользователя с наиболее встречающимся именем в заказах.

Query Query History

```
1 INSERT INTO usr (first_name, last_name)
2     SELECT u.first_name, '-' from usr u
3     JOIN ordr o ON u.id = o.usr_id
4     GROUP BY u.first_name
5     ORDER BY COUNT(*) DESC
6     LIMIT 1;
```

Data Output Messages Notifications

INSERT 0 1

Query returned successfully in 124 msec.

### 3. Создать простые и составные индексы и сравнить время выполнения

## 1) Выполнение без индекса

The screenshot shows a database query execution interface. The top section displays the SQL query being executed:

```
1 explain select * from ordr o
2     where o.usr_id = (select u.id from usr u where
3                                     u.first_name = 'Вася' AND
4                                     u.first_name = 'Пупкин')
5     AND o.address_from = 'СПБ';
```

Below the query, the 'Data Output' tab is active, showing the 'QUERY PLAN' for the query. The plan consists of the following steps:

Step	Operation
1	Seq Scan on ordr o (cost=13.75..21.12 rows=3 width=106)
2	Filter: ((usr_id = \$0) AND ((address_from)::text = 'СПБ'::text))
3	InitPlan 1 (returns \$0)
4	-> Result (cost=0.00..13.75 rows=1 width=4)
5	One-Time Filter: false
6	-> Seq Scan on usr u (cost=0.00..13.75 rows=1 width=...
7	Filter: ((first_name)::text = 'Вася'::text)

At the bottom of the interface, the status bar indicates: 'Total rows: 7 of 7' and 'Query complete 00:00:00.150'.

## 2) Простой индекс

Query

Query History

```

1 create index from_index on ordr(address_from);
2
3 explain select * from ordr o
4     where o.usr_id = (select u.id from usr u where
5                                     u.first_name = 'Вася' AND
6                                     u.first_name = 'Пупкин')
7     AND o.address_from = 'СПБ';

```

Data Output

Messages

Notifications

+

📄

▼

📋

🗑️

🗄️

⬇️

📈

QUERY PLAN	
	text
1	Seq Scan on ordr o (cost=13.75..21.12 rows=3 width=106)
2	Filter: ((usr_id = \$0) AND ((address_from)::text = 'СПБ'::text))
3	InitPlan 1 (returns \$0)
4	-> Result (cost=0.00..13.75 rows=1 width=4)
5	One-Time Filter: false
6	-> Seq Scan on usr u (cost=0.00..13.75 rows=1 width=...)
7	Filter: ((first_name)::text = 'Вася'::text)

Total rows: 7 of 7

Query complete 00:00:00.122


3) Составной индекс

Query Query History

```
1 explain select * from ordr o
2   where o.usr_id = (select u.id from usr u where
3                                     u.first_name = 'Вася' AND
4                                     u.first_name = 'Пупкин')
5   AND o.address_from = 'СПБ'
6   AND o.address_to = 'МСГ';
```

Data Output Messages Notifications



	QUERY PLAN	
	text	
1	Seq Scan on ordr o (cost=13.75..21.34 rows=1 width=106)	
2	Filter: ((usr_id = \$0) AND ((address_from)::text = 'СПБ'::text) AND ((address_to)::text = 'МСГ'::te...	
3	InitPlan 1 (returns \$0)	
4	-> Result (cost=0.00..13.75 rows=1 width=4)	
5	One-Time Filter: false	
6	-> Seq Scan on usr u (cost=0.00..13.75 rows=1 width=4)	
7	Filter: ((first_name)::text = 'Вася'::text)	

Total rows: 7 of 7 Query complete 00:00:00.122

Query

Query History

```

1 create index composite_index on ordr(address_from, address_to);
2
3 explain select * from ordr o
4     where o.usr_id = (select u.id from usr u where
5                                     u.first_name = 'Вася' AND
6                                     u.first_name = 'Пупкин')
7     AND o.address_from = 'СПБ'
8     AND o.address_to = 'МСГ';

```

Data Output

Messages

Notifications

≡+

📄

▼

📋

🗑️

🗄️

⬇️

📈

QUERY PLAN

text

🔒

1	Seq Scan on ordr o (cost=13.75..21.34 rows=1 width=106)
2	Filter: ((usr_id = \$0) AND ((address_from)::text = 'СПБ'::text) AND ((address_to)::text = 'МСГ'::te...
3	InitPlan 1 (returns \$0)
4	-> Result (cost=0.00..13.75 rows=1 width=4)
5	One-Time Filter: false
6	-> Seq Scan on usr u (cost=0.00..13.75 rows=1 width=4)
7	Filter: ((first_name)::text = 'Вася'::text)

Total rows: 7 of 7

Query complete 00:00:00.130

## Вывод

В ходе данной лабораторной работы мной были изучены сложные вложенные SQL запросы, а также индексы.

Последние не повлияли на скорость выполнения запросов из-за того, что количество строк в таблицах (порядка 200) не настолько больше, чтобы заметно затруднять поиск.