

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

**Отчет**

по лабораторной работе №4 «ЛР 4 Запросы на выборку и модификацию данных.  
**Представления. Работ с индексами»**

по дисциплине «Проектирование и реализация баз данных»

Автор: Зеленин Денис Сергеевич

Факультет: ИКТ

Группа: К3241

Преподаватель: Говорова М.М.



## Оглавление

1. Запросы к базе данных.....	3
2. Представления.....	9
3. Кастом запросы.....	10
4. Индексы.....	11
Вывод.....	12

**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.

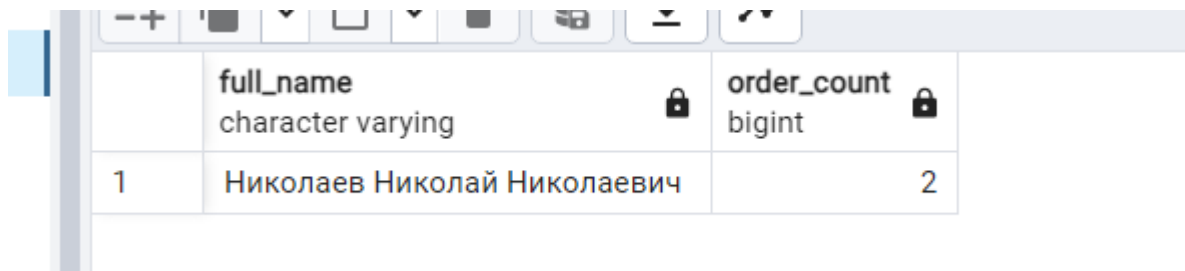
**Практическое задание:**

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 базе данных

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

```
select employee.full_name, COUNT(order_.order_id) AS order_count from order_
join employee on order_.fk_service_number = employee.service_number
where end_point = 'Улица Гагарина, 45'
group by employee.full_name
ORDER BY order_count DESC
LIMIT 1;
```



	full_name character varying	order_count bigint
1	Николаев Николай Николаевич	2

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

```
select * from car
where last_date_to < '2023-01-01' and mileage > 250000;
```

Сколько раз каждый пассажир воспользовался услугами таксопарка?

```
SELECT
    fk_client_id,
    COUNT(*) AS trip_count FROM order_
GROUP BY
    fk_client_id
ORDER BY
    trip_count DESC;
```

	fk_client_id character varying	trip_count bigint
1	C5	2
2	C1	1
3	C2	1
4	C3	1
5	C4	1
6	C7	1
7	C9	1
8	C6	1
9	C8	1
Total rows: 10 of 10    Query complete 00:00:00		

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

```
SELECT
    client.client_id,
    client.client_phone_number,
    client.name,
    COUNT(order_.fk_client_id) AS trip_count
FROM
    client
LEFT JOIN
    order_ ON client.client_id = order_.fk_client_id
GROUP BY
    client.client_id, client.name
ORDER BY
    trip_count DESC
LIMIT 1;
```

```

1  SELECT
2      client.client_id,
3      client.client_phone_number,
4      client.name,
5      COUNT(order_.fk_client_id) AS trip_count
6  FROM
7      client
8  LEFT JOIN
9      order_ ON client.client_id = order_.fk_client_id
10 GROUP BY
11     client.client_id, client.name
12 ORDER BY
13     trip_count DESC
14 LIMIT 1;

```

Data Output   Сообщения   Notifications

	client_id [PK] character varying (255)	client_phone_number bigint	name character varying	trip_count bigint
1	C5	6543210987	Андрей	2

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

```

SELECT
    employee.service_number,
    employee.full_name,
    employee.phone_number,
    car.price
FROM
    order_
JOIN
    car ON order_.fk_car_id = car.car_id
JOIN
    employee AS employee ON order_.fk_service_number = employee.service_number
ORDER BY
    car.price DESC
LIMIT 1;

```

```

1  SELECT
2      employee.service_number,
3      employee.full_name,
4      employee.phone_number,
5      car.price
6  FROM
7      order_
8  JOIN
9      car ON order_.fk_car_id = car.car_id
10 JOIN
11     employee AS employee ON order_.fk_service_number = employee.service_number
12 ORDER BY
13     car.price DESC
14 LIMIT 1;

```

Data Output

Сообщения

Notifications

+

📄

▼

📋

▼

🗑️

🗄️

⬇️

📈

	service_number integer	full_name character varying	phone_number character varying (20)	price bigint
1	3688	Григорьев Григорий Григорьевич	1213141516	9000000

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

```
SELECT
    client.client_id,
    client.name
FROM
    order_
JOIN
    client ON order_.fk_client_id = client.client_id
JOIN
    employee ON order_.fk_service_number::integer = employee.service_number
GROUP BY
    client.client_id, client.name
HAVING
    COUNT(order_.fk_service_number) > 1;
```

Data Output   Сообщения   Notifications		
<div> <div>≡</div> <div>📄</div> <div>▼</div> <div>📋</div> <div>▼</div> <div>🗑️</div> <div>🗄️</div> <div>⬇️</div> <div>📈</div> </div>		
	<b>client_id</b> [PK] character varying (255) ✎	<b>name</b> character varying ✎
1	C7	Анастасия
2	C5	Андрей

Какие автомобили имеют пробег больше среднего пробега для своей марки.

```
SELECT car_specifications.brand, car_specifications.model_name, car.mileage
FROM car
```

```
JOIN car_specifications ON car.fk_car_specifications = car_specifications.specifications_id
WHERE car.mileage > (
```

```
    SELECT AVG(car_avg.mileage)
```

```
    FROM car AS car_avg
```

```
    JOIN car_specifications AS spec_avg ON car_avg.fk_car_specifications =
spec_avg.specifications_id
```

```
    WHERE spec_avg.brand = car_specifications.brand
```

```
    GROUP BY spec_avg.brand
```

```
)
```

```
ORDER BY car_specifications.brand, car_specifications.model_name;
```

Data Output   Сообщения   Notifications			
<div> <div>≡</div> <div>📄</div> <div>▼</div> <div>📋</div> <div>▼</div> <div>🗑️</div> <div>🗄️</div> <div>⬇️</div> <div>📈</div> </div>			
	<b>brand</b> character varying 🔒	<b>model_name</b> character varying 🔒	<b>mileage</b> bigint 🔒
1	Audi	Q3	324532
2	Mercedes-Benz	CLA	432532
3	Nissan	Maxima	432565

## 2 Представления

содержащее сведения о незанятых на данный момент водителях;

Запрос История запросов



Scratch I

```
1 CREATE VIEW free_drivers AS
2 SELECT employee.full_name ,employee.employee_address, employee.phone_number,employee.service_number
3 FROM work_schedule
4 JOIN employee ON work_schedule.fk_service_number = employee.service_number
5 WHERE work_schedule.status IN ('отпуск', 'не вышел');
6
```

	full_name character varying	employee_address character varying	phone_number character varying (20)	service_number integer
1	Кузнецова Анна Владимировна	ул. Гоголя, 4	5556667778	3684
2	Павлов Павел Павлович	ул. Чехова, 5	4443332221	3685
3	Орлова Ольга Анатольевна	ул. Тургенева, 7	9990001112	3687
4	Григорьев Григорий Григорьевич	ул. Шекспира, 8	1213141516	3688

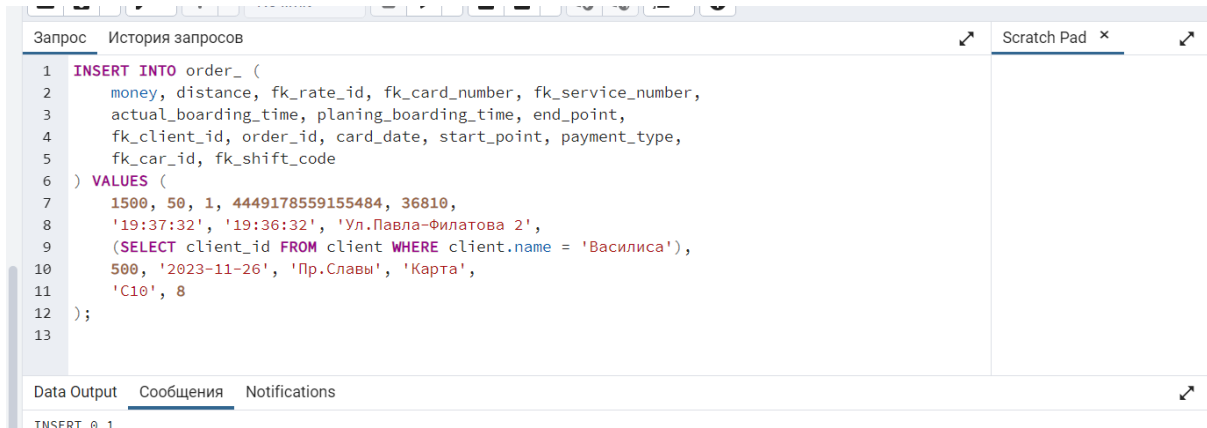
Зарплата всех водителей за вчерашний день.

```
1 CREATE VIEW drivers_money AS
2 SELECT employee.full_name, employee.service_number, sum(money) as total_earned from order_
3 join employee on order_.fk_service_number = employee.service_number
4 group by fk_service_number, employee.full_name, employee.service_number
5 order by total_earned desc
6
```

	full_name character varying	service_number integer	total_earned numeric
1	Иванов Иван Иванович	3681	1554
2	Николаев Николай Николаевич	3689	1352
3	Кузнецова Анна Владимировна	3684	816
4	Григорьев Григорий Григорьевич	3688	800
5	Орлова Ольга Анатольевна	3687	343
6	Павлов Павел Павлович	3685	330
7	Федоров Федор Федорович	36810	190



### 3 Кастом запросы



The screenshot shows a database query editor with a tab labeled 'Запрос' (Query) and 'История запросов' (Query History). The main area contains an SQL INSERT statement. To the right is a 'Scratch Pad' tab. At the bottom, there are tabs for 'Data Output', 'Сообщения' (Messages), and 'Notifications'. The status bar at the very bottom indicates 'INSERT @ 1'.

```
1 INSERT INTO order_ (
2     money, distance, fk_rate_id, fk_card_number, fk_service_number,
3     actual_boarding_time, planing_boarding_time, end_point,
4     fk_client_id, order_id, card_date, start_point, payment_type,
5     fk_car_id, fk_shift_code
6 ) VALUES (
7     1500, 50, 1, 4449178559155484, 36810,
8     '19:37:32', '19:36:32', 'Ул.Павла-Филатова 2',
9     (SELECT client_id FROM client WHERE client.name = 'Василиса'),
10    500, '2023-11-26', 'Пр.Славы', 'Карта',
11    'C10', 8
12 );
13
```

Вставка данных по имени клиента

```
INSERT INTO order_ (
    money, distance, fk_rate_id, fk_card_number, fk_service_number,
    actual_boarding_time, planing_boarding_time, end_point,
    fk_client_id, order_id, card_date, start_point, payment_type,
    fk_car_id, fk_shift_code
) VALUES (
    1500, 50, 1, 4449178559155484, 36810,
    '19:37:32', '19:36:32', 'Ул.Павла-Филатова 2',
    (SELECT client_id FROM client WHERE client.name = 'Василиса'),
    500, '2023-11-26', 'Пр.Славы', 'Карта',
    'C10', 8
);
```

Обновление данных по имени

```
UPDATE order_
SET
    money = 2000,
    distance = 60,
    fk_rate_id = (
        SELECT rate_id
        FROM rate
        WHERE rate_id = 3
    ),
    fk_card_number = 4863642573169460,
    fk_service_number = 3684,
    actual_boarding_time = '20:45:00',
    planing_boarding_time = '20:30:00',
    end_point = 'Ул. Новая 15',
    fk_client_id = (SELECT client_id FROM client WHERE name = 'Георгий'),
    card_date = '2023-11-27',
    start_point = 'Проспект Мира',
    payment_type = 'Наличные',
    fk_car_id = 'C3',
    fk_shift_code = 10
WHERE order_id = 'or26';
```

Удаление данных о клиенте по его имени

```
DELETE FROM order_  
WHERE fk_client_id = (  
    SELECT client_id  
    FROM client  
    WHERE name = 'Георгий'  
);
```

#### 4 Индексы

Создание индексов

Запрос История запросов

```
1 create index idx_year_of_manufactured on car (year_of_manufactured);  
2 create index idx_price_state_number on car (state_number, price )  
3  
  
1 select * from car  
2 where price < 10000000 and state_number like 'A%'
```

С индексом

Data Output Сообщения Notifications

Запрос выполнен успешно. Общее время выполнения: 51 msec.  
обработано строк: 2.

	QUERY PLAN	
	text	
1	Seq Scan on car (cost=0.00..1.15 rows=1 width=410)	
2	Filter: ((price < 10000000) AND ((state_number)::text ~~ 'A%':tex...	

Без индекса

Data Output Сообщения Notifications

Запрос выполнен успешно. Общее время выполнения: 78 msec.  
обработано строк: 2.

## История запросов

The screenshot shows the 'History' window in pgAdmin. At the top, there is a toggle switch for 'Show queries generated internally by pgAdmin?' which is turned on. Below this are two buttons: 'Remove' and 'Remove All'. The main area is divided into two sections: 'Today - 27.11.2023' and 'Yesterday - 26.11.2023'. The 'Today' section contains a list of queries, each with a timestamp and a truncated SQL statement. The queries are as follows:

Timestamp	Query
14:05:18	UPDATE order_ SET money = 2000, distance = 60...
14:05:03	SELECT * FROM public.payment_method ORDER BY ...
14:04:36	UPDATE order_ SET money = 2000, distance = 60...
14:04:07	UPDATE order_ SET money = 2000, distance = 60...
14:03:39	UPDATE order_ SET money = 2000, distance = 60...
14:03:16	UPDATE order_ SET money = 2000, distance = 60...
14:03:00	UPDATE order_ SET money = 2000, distance = 60...
14:02:17	SELECT * FROM public.rate ORDER BY rate_id ASC
14:02:05	UPDATE order_ SET money = 2000, distance = 60...
14:01:54	UPDATE order_ SET money = 2000, distance = 60...
13:41:43	SELECT * FROM public.work_schedule ORDER BY s...

The 'Yesterday' section contains one query:

Timestamp	Query
	INSERT INTO order_ ( money, distance, fk_rate...

## Вывод

Лабораторная работа по PostgreSQL включала изучение написания SQL-запросов, создание индексов для оптимизации производительности, а также создание представлений для абстрагирования сложных запросов. В процессе выполнения лабораторной работы были использованы различные типы запросов, созданы простые и составные индексы, а также представления для улучшения читаемости кода. Оптимизация запросов проводилась с целью повышения эффективности выполнения операций в базе данных.