

**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**

Обучающийся Саид Наваф

Факультет прикладной информатики

Группа К3241

Направление подготовки 09.03.03 Прикладная информатика

Образовательная программа Мобильные и сетевые технологии 2023

Преподаватель Говорова Марина Михайловна

Санкт-Петербург

2025-2026

1. Цель работы:

овладеть практическими навыками создания представлений и запросов на выборку данных к базе данных PostgreSQL, использования подзапросов при модификации данных и индексов.

2. Практическое задание:

1. Создать запросы и представления на выборку данных к базе данных PostgreSQL (согласно индивидуальному заданию лабораторной работы №2, 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.

3. Схема базы данных (ЛР 3)

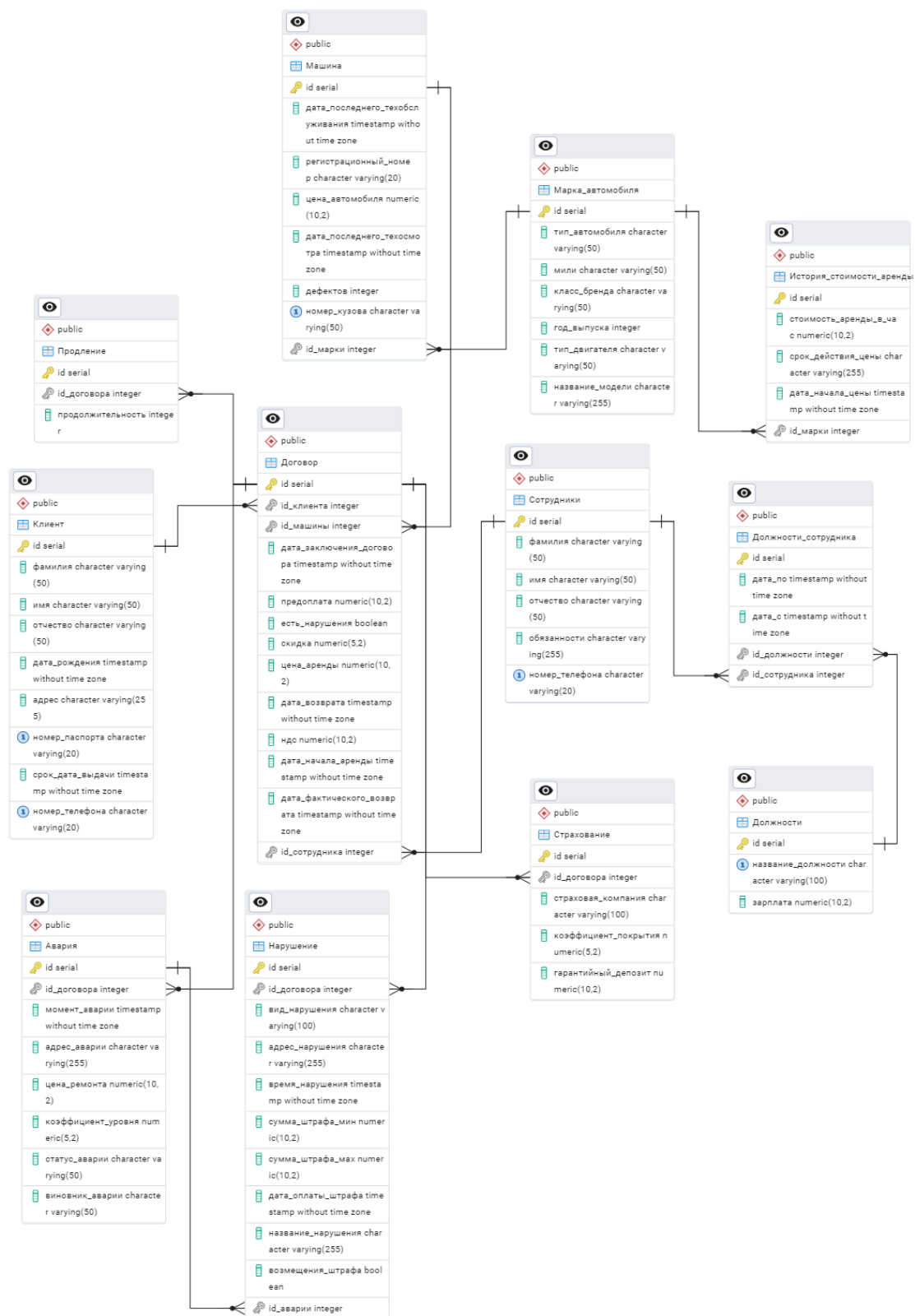


Рисунок 1 – ERD-схема базы данных

4 Выполнение:

4.1 Запросы к базе данных

Запрос №1

Описание запроса:

Вывести данные о клиенте, который чаще всего арендовал автомобили в компании, включая количество его аренд и среднюю продолжительность проката.

SQL-код запроса:

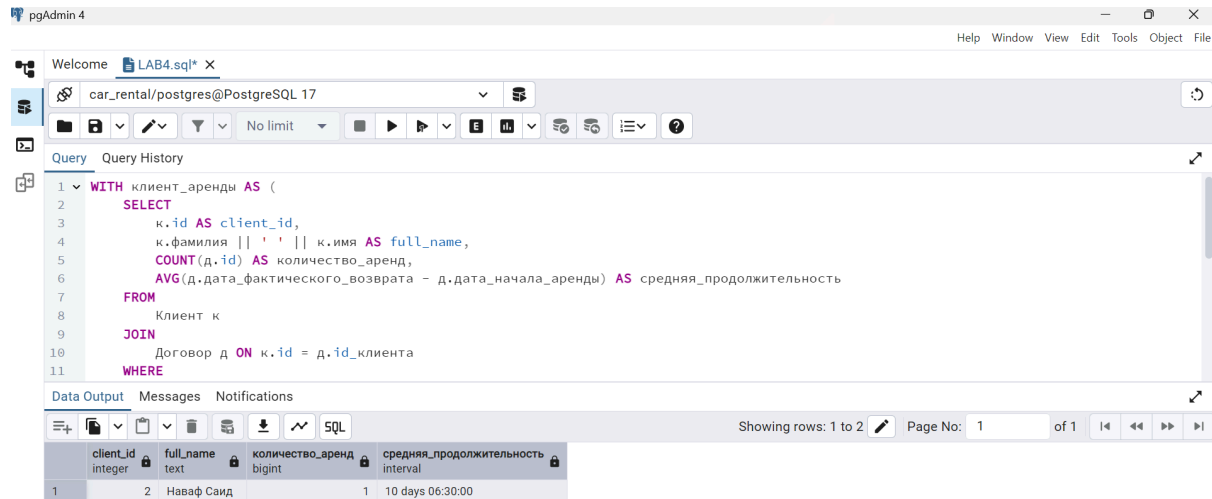
```
WITH клиент_аренды AS (  
  
    SELECT  
  
        к.id AS client_id,  
  
        к.фамилия || ' ' || к.имя AS full_name,  
  
        COUNT(д.id) AS количество_аренд,  
  
        AVG(д.дата фактического возврата - д.дата начала аренды) AS средняя продолжительность  
  
    FROM  
  
        Клиент к  
  
    JOIN  
  
        Договор д ON к.id = д.id клиента  
  
    WHERE  
  
        д.дата фактического возврата IS NOT NULL  
  
    GROUP BY  
  
        к.id, к.фамилия, к.имя  
  
)  
  
SELECT  
  
    client_id,  
  
    full_name,  
  
    количество_аренд,  
  
    средняя_продолжительность  
  
FROM  
  
    клиент_аренды  
  
WHERE
```

количество_аренд = (SELECT MAX(количество_аренд) FROM клиент_аренды)

ORDER BY

средняя_продолжительность DESC;

Скриншот выполнения запроса:



Запрос №2

Описание запроса:

Вывести данные о договорах аренды, где стоимость аренды превышает среднюю стоимость по всем договорам, а дата возврата автомобиля просрочена.

SQL-код запроса:

SELECT

д.id,

с.фамилия || ' ' || с.имя AS клиент,

т.регистрационный_номер,

д.цена_аренды,

д.дата_возврата,

CURRENT_DATE - д.дата_возврата AS дней_просрочки

FROM

Договор d

JOIN

Клиент c ON d.id_клиента = c.id

JOIN

Машина m ON d.id_машины = m.id

WHERE

d.цена_аренды > (SELECT AVG(цена_аренды) FROM Договор)

AND d.дата_фактического_возврата IS NULL

AND d.дата_возврата < CURRENT_DATE;

Скриншот выполнения запроса:

The screenshot shows a PostgreSQL query editor interface. The query is as follows:

```
1 SELECT
2     d.id,
3     c.фамилия || ' ' || c.имя AS клиент,
4     m.регистрационный_номер,
5     d.цена_аренды,
6     d.дата_возврата,
7     CURRENT_DATE - d.дата_возврата AS дней_просрочки
8 FROM
9     Договор d
10 JOIN
11     Клиент c ON d.id_клиента = c.id
12 JOIN
13     Машина m ON d.id_машины = m.id
14 WHERE
15     d.цена_аренды > (SELECT AVG(цена_аренды) FROM Договор)
16     AND d.дата_фактического_возврата IS NULL
17     AND d.дата_возврата < CURRENT_DATE;
```

The results are displayed in a table with the following columns: id, клиент, регистрационный_номер, цена_аренды, дата_возврата, and дней_просрочки. The first row shows data for a client named Сидоров Алексей, registration number M789OK79, rental price 4000.00, return date 2023-08-20 18:00:00, and a delay of 630 days 06:00:00.

id	клиент	регистрационный_номер	цена_аренды	дата_возврата	дней_просрочки
3	Сидоров Алексей	M789OK79	4000.00	2023-08-20 18:00:00	630 days 06:00:00

Total rows: 1 Query complete 00:00:00.074

Запрос №3

Описание запроса:

Вывести статистику по клиентам: количество договоров, общую сумму платежей и среднюю продолжительность аренды в днях.

SQL-код запроса

```
SELECT

    c.id AS client_id,

    c.фамилия || ' ' || c.имя AS client_name,

    COUNT(d.id) AS contract_count,

    SUM(d.цена_аренды) AS total_payment,

    ROUND(AVG(

        EXTRACT(DAY FROM (d.дата_фактического_возврата - d.дата_начала_аренды))

    )) AS avg_rental_days

FROM

    Клиент c

LEFT JOIN

    Договор d ON c.id = d.id_клиента

WHERE

    d.дата_фактического_возврата IS NOT NULL

GROUP BY

    c.id, c.фамилия, c.имя

HAVING

    COUNT(d.id) > 0

ORDER BY

    contract_count DESC;
```

Скриншот выполнения запроса:

Query

Query History

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

SELECT
c.id AS client_id,
с.фамилия || ' ' || с.имя AS client_name,
COUNT(d.id) AS contract_count,
SUM(d.цена_аренды) AS total_payment,
ROUND(AVG(
EXTRACT(DAY FROM (d.дата_фактического_возврата - d.дата_начала_аренды))
)) AS avg_rental_days
FROM
Клиент c
LEFT JOIN
Договор d ON c.id = d.id_клиента
WHERE
d.дата_фактического_возврата IS NOT NULL
GROUP BY
c.id, с.фамилия, с.имя

Data Output

Messages

Notifications

≡

📄

▼

📋

▼

🗑️

🗑️

📥

⬇️

📈

SQL

Shc

	client_id integer	client_name text	contract_count bigint	total_payment numeric	avg_rental_days numeric
1	2	Наваф Саид	1	3500.00	10
2	1	Иванов Иван	1	3000.00	9

Запрос №4

Описание запроса:

Вывести данные клиента, который заключил максимальное количество договоров аренды автомобилей.

SQL-код запроса:

WITH client_contracts AS (

SELECT

c.id AS client_id,

с.фамилия || ' ' || с.имя AS client_name,

COUNT(d.id) AS contract_count

FROM

Клиент c

JOIN

Договор d ON c.id = d.id_клиента

GROUP BY


```

        c.id, c.фамилия, c.имя
    )
SELECT
    client_id,
    client_name,
    contract_count
FROM
    client_contracts
WHERE
    contract_count = (SELECT MAX(contract_count) FROM client_contracts)
ORDER BY
    client_id;

```

Скриншот выполнения запроса:

[Query](#)
[Query History](#)

9

Договор d ON c.id = d.id_клиента

10

GROUP BY

11

c.id, c.фамилия, c.имя

12

)

13

SELECT

14

client_id,

15

client_name,

16

contract_count

17

FROM

18

client_contracts

19

WHERE

20

contract_count = (SELECT MAX(contract_count) FROM client_contracts)

21

ORDER BY

22

client_id;

[Data Output](#)
[Messages](#)
[Notifications](#)

≡+

📄

▼

📋

▼

🗑️

🗄️

⬇️

📈

SQL

	client_id integer	client_name text	contract_count bigint
1	1	Иванов Иван	1
2	2	Наваф Саид	1
3	3	Сидоров Алексей	1

Запрос №5

Описание запроса:

Вывести данные о водителе, который в настоящее время арендует автомобиль с максимальной стоимостью.

SQL-код запроса:

```
SELECT
    s.id AS driver_id,
    s.фамилия || ' ' || s.имя AS driver_name,
    m.цена_автомобиля AS car_price,
    m.регистрационный_номер AS license_plate,
    ma.название_модели AS car_model
FROM
    Сотрудники s
JOIN
    Договор d ON s.id = d.id_сотрудника
JOIN
    Машина m ON d.id_машины = m.id
JOIN
    Марка_автомобиля ma ON m.id_марки = ma.id
WHERE
    d.дата_фактического_возврата IS NULL
    AND m.цена_автомобиля = (
        SELECT MAX(m2.цена_автомобиля)
        FROM Машина m2
        JOIN Договор d2 ON m2.id = d2.id_машины
        WHERE d2.дата_фактического_возврата IS NULL
    )
ORDER BY
    m.цена_автомобиля DESC
LIMIT 1;
```

Скриншот выполнения запроса:

Query Query History

```
1 SELECT
2     s.id AS driver_id,
3     s.фамилия || ' ' || s.имя AS driver_name,
4     m.цена_автомобиля AS car_price,
5     m.регистрационный_номер AS license_plate,
6     ma.название_модели AS car_model
7 FROM
8     Сотрудники s
9 JOIN
10    Договор d ON s.id = d.id_сотрудника
11 JOIN
12    Машина m ON d.id_машины = m.id
13 JOIN
14    Марка_автомобиля ma ON m.id_марки = ma.id
```

Data Output Messages Notifications

	driver_id integer	driver_name text	car_price numeric (10,2)	license_plate character varying (20)	car_model character varying (255)
1	2	Кузнецова Ольга	2200000.00	M789OK79	Toyota Prius

Запрос №6

Описание запроса:

Вывести данные о клиентах, которые арендовали автомобили только у одного конкретного сотрудника (менеджера по аренде).

SQL-код запроса:

```
WITH client_employee_pairs AS (
    SELECT
        d.id_клиента,
        c.фамилия || ' ' || c.имя AS client_name,
        d.id_сотрудника,
        s.фамилия || ' ' || s.имя AS employee_name,
        COUNT(d.id) AS contract_count
    FROM
        Договор d
        JOIN Клиент c ON d.id_клиента = c.id
        JOIN Сотрудник s ON d.id_сотрудника = s.id
    WHERE contract_count = 1
)
```

JOIN

Клиент с ON d.id_клиента = c.id

JOIN

Сотрудники s ON d.id_сотрудника = s.id

GROUP BY

d.id_клиента, c.фамилия, c.имя, d.id_сотрудника, s.фамилия, s.имя

)

SELECT

id_клиента,

client_name,

id_сотрудника,

employee_name,

contract_count

FROM

client_employee_pairs

WHERE

id_клиента IN (

SELECT id_клиента

FROM Договор

GROUP BY id_клиента

HAVING COUNT(DISTINCT id_сотрудника) = 1

)

ORDER BY

contract_count DESC;

Скриншот выполнения запроса:

Query

Query History

20

id_сотрудника,

21

employee_name,

22

contract_count

23

FROM

24

client_employee_pairs

25

WHERE

26

id_клиента IN (

27

SELECT id_клиента

28

FROM Договор

29

GROUP BY id_клиента

30

HAVING COUNT(DISTINCT id_сотрудника) = 1

31

)

32

ORDER BY

33

contract_count DESC;

Data Output

Messages

Notifications

≡

+

📄

▼

📋

▼

🗑️

🗄️

⬇️

📈

SQL

	id_клиента integer	client_name text	id_сотрудника integer	employee_name text	contract_count bigint
1	2	Наваф Саид	1	Смирнов Дмитрий	1
2	3	Сидоров Алексей	2	Кузнецова Ольга	1
3	1	Иванов Иван	1	Смирнов Дмитрий	1

Представление №1

Описание представления:

Представление отображает список всех неработающих сотрудников, включая уволенных и никогда не трудоустроенных, с указанием их последней должности и даты увольнения.

SQL-код представления:

```
DROP VIEW IF EXISTS незанятые_сотрудники;
```

```
CREATE VIEW незанятые_сотрудники AS
```

```
SELECT
```

```
    s.id AS employee_id,
```

```
    s.фамилия || ' ' || s.имя || COALESCE(' ' || s.отчество, '') AS full_name,
```

```

s.номер_телефона AS phone,

d.название_должности AS position,

MAX(ds.дата_по) AS last_work_date

FROM

    Сотрудники s

LEFT JOIN

    Должности_сотрудника ds ON s.id = ds.id_сотрудника

LEFT JOIN

    Должности d ON ds.id_должности = d.id

WHERE

    ds.id_сотрудника IS NULL

    OR (ds.дата_по < CURRENT_DATE

    AND NOT EXISTS (

        SELECT 1 FROM Должности_сотрудника

        WHERE id_сотрудника = s.id

        AND дата_по >= CURRENT_DATE

    ))

GROUP BY

    s.id, s.фамилия, s.имя, s.отчество, s.номер_телефона, d.название_должности

ORDER BY

    last_work_date DESC NULLS LAST;

SELECT * FROM незанятые_сотрудники;

```

Скриншоты создания и отображения представления:

Query Query History

```
1
2 DROP VIEW IF EXISTS незанятые_сотрудники;
3
4 CREATE VIEW незанятые_сотрудники AS
5 SELECT
6     s.id AS employee_id,
7     s.фамилия || ' ' || s.имя || COALESCE(' ' || s.отчество, '') AS full_name,
8     s.номер_телефона AS phone,
9     d.название_должности AS position,
10    MAX(ds.дата_по) AS last_work_date
11 FROM
12     Сотрудники s
13 LEFT JOIN
14     Должности_сотрудника ds ON s.id = ds.id сотрудника
```

Data Output Messages Notifications

Successfully run. Total query runtime: 110 msec.
7 rows affected.

Total rows: 7 Query complete 00:00:00.110

Query Query History

```
1 SELECT * FROM незанятые_сотрудники;
```

Data Output Messages Notifications

Showing rows: 1 to 7						
	employee_id integer	full_name text	phone character varying (20)	position character varying (100)	last_work_date timestamp without time zone	
1	4	Иванов Петр Сергеевич	+79054445566	Водитель	2022-02-15 00:00:00	
2	1	Смирнов Дмитрий Алексеевич	+79051112233	Менеджер по аренде	2022-01-10 00:00:00	
3	6	Васильев Алексей Михайлович	+79056667788	Водитель	2021-06-01 00:00:00	
4	2	Кузнецова Ольга Викторовна	+79052223344	Технический специалист	2021-05-15 00:00:00	
5	3	Попов Андрей Игоревич	+79053334455	Директор филиала	2020-03-01 00:00:00	

Представление №2

Описание представления:

Отображает автомобили, доступные для аренды в текущий момент
(не находящиеся в действующих договорах аренды)

SQL-код представления:

```
CREATE OR REPLACE VIEW свободные_автомобили AS

SELECT

    м.id,

    м.регистрационный_номер,

    мар.название_модели,

    мар.тип_автомобиля,

    мар.класс_бренда,

    м.цена_автомобиля,

    м.дата_последнего_техобслуживания,

    м.дата_последнего_техосмотра

FROM

    Машина м

JOIN

    Марка_автомобиля мар ON м.id_марки = мар.id

LEFT JOIN

    Договор д ON м.id = д.id_машины

    AND д.дата_начала_аренды <= CURRENT_DATE

    AND (д.дата_фактического_возврата IS NULL OR д.дата_фактического_возврата >= CURRENT_DATE)

WHERE

    д.id IS NULL

    AND м.дефектов = 0

ORDER BY

    мар.класс_бренда DESC,

    м.цена_автомобиля;
```


Query Query History

```
2  ✓ CREATE OR REPLACE VIEW свободные_автомобили AS
3  SELECT
4      м.id,
5      м.регистрационный_номер,
6      мар.название_модели,
7      мар.тип_автомобиля,
8      мар.класс_бренда,
9      м.цена_автомобиля,
10     м.дата_последнего_техобслуживания,
11     м.дата_последнего_техосмотра
12 FROM
```

Data Output Messages Notifications

CREATE VIEW

Query returned successfully in 79 msec.

Query Query History

```
1 SELECT * FROM свободные_автомобили;
```

Data Output Messages Notifications

Showing rows: 1 to 1 Page No: 1 of 1

	id	регистрационный_номер	название_модели	тип_автомобиля	класс_бренда	цена_автомобиля	дата_последнего_техобслуживания	дата_последнего_техосмотра
	integer	character varying (20)	character varying (255)	character varying (50)	character varying (50)	numeric (10,2)	timestamp without time zone	timestamp without time zone
1	1	A123BC77	Toyota Camry	Седан	Премиум	2500000.00	2023-05-15 00:00:00	2023-05-20 00:00:00

4.3 Запросы на модификацию данных

INSERT-запрос

Описание запроса:

Добавление нового автомобиля марки "Toyota Camry" в автопарк с проверкой уникальности регистрационного номера. Запрос автоматически устанавливает

```

INSERT INTO Машина (
    дата_последнего_техобслуживания,
    регистрационный_номер,
    цена_автомобиля,
    дата_последнего_техосмотра,
    дефектов,
    номер_кузова,
    id_марки
)
SELECT
    '2023-05-15',
    'A123AA177',
    4200000.00,
    '2023-11-20',
    0,
    'X9FGH45KL6789123',
    id
FROM Марка_автомобиля
WHERE название_модели = 'Toyota Camry'
AND NOT EXISTS (
    SELECT 1 FROM Машина
    WHERE регистрационный_номер = 'A123AA177'
)
RETURNING *;

```

SQL-код запроса:

[Query](#) [Query History](#)

```

1  ▾ INSERT INTO Машина (
2      дата_последнего_техобслуживания,
3      регистрационный_номер,
4      цена_автомобиля,
5      дата_последнего_техосмотра,
6      дефектов,
7      номер_кузова,
8      id_марки
9  )
10 SELECT

```

[Data Output](#) [Messages](#) [Notifications](#)

Successfully run. Total query runtime: 180 msec.
0 rows affected.

Query

Query History

1

2

SELECT * FROM Машина

ORDER BY id DESC;

Data Output

Messages

Notifications

SQL

Showing rows: 1 to 4

Page No: 1 of 1

	id [PK] integer	дата_последнего_техобслуживания timestamp without time zone	регистрационный_номер character varying (20)	цена_автомобиля numeric (10,2)	дата_последнего_техосмотра timestamp without time zone	дефектов integer	номер_кузова character varying (50)	id_марки integer
1	21	2023-05-15 00:00:00	A123AA177	4200000.00	2023-11-20 00:00:00	0	X9FGH45KL6789123	1
2	3	2023-07-01 00:00:00	M789OK79	2200000.00	2023-07-05 00:00:00	0	JT2BF22K1W0123456	3
3	2	2023-06-10 00:00:00	X456YK78	1800000.00	2023-06-15 00:00:00	1	XW8ZZZ5NZJG000001	2
4	1	2023-05-15 00:00:00	A123BC77	2500000.00	2023-05-20 00:00:00	0	XTA210997654321	1

UPDATE-запрос

Описание запроса:

Обновить дату последнего технического обслуживания на текущую дату для всех автомобилей в таблице Машина, которые соответствуют следующим условиям:

SQL-код запроса:

```
UPDATE public."Машина"

SET "дата_последнего_техобслуживания" = CURRENT_DATE

WHERE "id_марки" = 1

AND id IN (

    SELECT "id_машины"

    FROM public."Договор"

    WHERE "дата_фактического_возврата" IS NULL

        OR "дата_фактического_возврата" > CURRENT_DATE

);
```

Query Query History

```
1  UPDATE public."Машина"
2  SET "дата_последнего_техобслуживания" = CURRENT_DATE
3  WHERE "id_марки" = 1
4  AND id IN (
5      SELECT "id_машины"
6      FROM public."Договор"
7      WHERE "дата_фактического_возврата" IS NULL
8           OR "дата_фактического_возврата" > CURRENT_DATE
9  );
```

Data Output Messages Notifications

UPDATE 0

Query returned successfully in 114 msec.

```
1  SELECT *
2  FROM public."Машина"
3  ORDER BY "дата_последнего_техобслуживания" DESC;
```

Data Output Messages Notifications

	id [PK] integer	дата_последнего_техобслуживания timestamp without time zone	регистрационный_номер character varying (20)	цена_автомобиля numeric (10,2)	дата_последнего_техосмотра timestamp without time zone	дефектов integer	номер_кузова character varying (50)	id_марки integer
1	3	2023-07-01 00:00:00	M789OK79	2200000.00	2023-07-05 00:00:00	0	JT2BF22K1W0123456	3
2	2	2023-06-10 00:00:00	X456YK78	1800000.00	2023-06-15 00:00:00	1	XW8ZZZ5NZJG000001	2
3	1	2023-05-15 00:00:00	A123BC77	2500000.00	2023-05-20 00:00:00	0	XTA210997654321	1

DELETE-запрос

Описание запроса:

Удалить из таблицы Нарушение все записи о нарушениях, которые связаны с договорами, где автомобиль принадлежит марке с id_марки = 2 (например, Volkswagen Tiguan, согласно тестовым данным), и где нарушение было оплачено (поле возмещения_штрафа = TRUE), а также дата оплаты штрафа (дата_оплаты_штрафа) раньше текущей даты минус 6 месяцев.

SQL-код запроса:

```

DELETE FROM public."Нарушение"

WHERE "id_договора" IN (

    SELECT d.id

    FROM public."Договор" d

    JOIN public."Машина" m ON d."id_машины" = m.id

    WHERE m."id_марки" = 2

)

AND "возмещения_штрафа" = TRUE

AND "дата_оплаты_штрафа" < CURRENT_DATE - INTERVAL '6 months';

```

Скриншоты выполнения запроса:

Query
Query History

```

1  ▾ DELETE FROM public."Нарушение"
2  WHERE "id_договора" IN (
3      SELECT d.id
4      FROM public."Договор" d
5      JOIN public."Машина" m ON d."id_машины" = m.id
6      WHERE m."id_марки" = 2
7  )
8  AND "возмещения_штрафа" = TRUE
9  AND "дата_оплаты_штрафа" < CURRENT_DATE - INTERVAL '6 months';

```

Data Output
Messages
Notifications

DELETE 1

Query returned successfully in 142 msec.

Query
Query History

```

1  ▾ SELECT *
2  FROM public."Нарушение"
3  ORDER BY "дата_оплаты_штрафа" DESC;

```

Data Output
Messages
Notifications

Showing rows: 1 to 1
Page No: 1
of 1

	id [PK] integer	id_договора integer	вид_нарушения character varying (100)	адрес_нарушения character varying (255)	время_нарушения timestamp without time zone	сумма_штрафа_мин numeric (10,2)	сумма_штрафа_маж numeric (10,2)	дата_оплаты_штрафа timestamp without time zone	н с
1		1	2 Превышение скорости	ул. Ленина, 15	2023-08-13 12:45:00	500.00	1500.00	[null]	f

4.4 Создание индексов

Простой индекс

SQL-код запроса:

```
SELECT * FROM "Договор" WHERE "цена_аренды" > 3000  
  
ORDER BY "цена_аренды" DESC;
```

SQL-код индекса:

```
CREATE INDEX idx_rental_price ON "Договор"("цена_аренды");
```

Query

Query History

1

SELECT * FROM "Договор"

2

ORDER BY id Asc

Data Output

Messages

Notifications

Проверим скорость выполнения запроса без индекса:

Query	Query History
<pre>1 SELECT * FROM "Договор" WHERE "цена_аренды" > 3000 2 ORDER BY "цена_аренды" DESC;</pre>	
Data Output	Messages Notifications
Successfully run. Total query runtime: 157 msec. 2 rows affected.	

Среднее время выполнения запроса 157 ms

Теперь рассмотрим раздел Explain:

Query

Query History

1

2

SELECT

*

FROM

"Договор"

WHERE

"цена_аренды"

>

3000

ORDER BY

"цена_аренды"

DESC;

Data Output

Messages

Explain

×

Notifications

Graphical

Analysis

Statistics

🔍

🔄

🔍

📄

↓

📊

Договор

→

📊


📊

Sort

Теперь создадим индекс по столбцу "цена_аренды" таблицы "Договор", что позволяет ускорить поиск и сортировку по этому полю.

Query	Query History
1	CREATE INDEX idx_rental_price ON "Договор"("цена_аренды");
<div> Data Output Messages Explain X Notifications </div>	
CREATE INDEX Query returned successfully in 74 msec.	

Выполним тот же запрос, но на этот раз при созданном индексе:

Query	Query History
1 	SELECT * FROM "Договор" WHERE "цена_аренды" > 3000
2	ORDER BY "цена_аренды" DESC ;
<div> Data Output Messages Explain X Notifications </div>	
Successfully run. Total query runtime: 99 msec. 2 rows affected.	

Среднее время выполнения запроса снизилось до 99 мс. Разница заметна, но не настолько ощутима, так как и без индекса выполнение запроса происходит достаточно быстро (возможно, в том числе и за счёт параллельных вычислений).

Проверим раздел Explain:

Query

Query History

1

2

SELECT

*

FROM

"Договор"

WHERE

"цена_аренды"

>

3000

ORDER BY

"цена_аренды"

DESC;

Data Output

Messages

Explain

×

Notifications

Graphical

Analysis

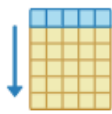
Statistics

⌕


↕

⌕

↓


Договор

→


Sort

Теперь удалим индекс:

Query Query History

1 DROP INDEX idx_rental_price;

Data Output Messages Notifications

DROP INDEX

Query returned successfully in 68 msec.

Итоговое время:

Без индекса: 157 мс.

С индексом: 99 мс.

Составной индекс

SQL-код запроса:

SELECT *

FROM "Машина"

WHERE EXTRACT(YEAR FROM "дата_последнего_техосмотра") >= 2020

AND "дефектов" = 0

ORDER BY "цена_автомобиля" DESC;

SQL-код индекса:

```
CREATE INDEX idx_car_brand_year_price_defects ON "Марка_автомобиля"
```

```
("год_выпуска", "id") INCLUDE ("название_модели");
```

Скриншоты работы с индексом:

QueryQuery History

1 SELECT * FROM "Машина"

2 ORDER BY id asc

3

Data OutputMessagesNotifications								
Showing rows: 1 to 3Page No: 1 of 1								
	id [PK] integer	дата_последнего_техобслуживания timestamp without time zone	регистрационный_номер character varying (20)	цена_автомобиля numeric (10,2)	дата_последнего_техосмотра timestamp without time zone	дефектов integer	номер_кузова character varying (50)	id_марки integer
1	1	2023-05-15 00:00:00	A123BC77	2500000.00	2023-05-20 00:00:00	0	XTA210997654321	1
2	2	2023-06-10 00:00:00	X456YK78	1800000.00	2023-06-15 00:00:00	1	XW8ZZZ5NZJG000001	2
3	3	2023-07-01 00:00:00	M789OK79	2200000.00	2023-07-05 00:00:00	0	JT2BF22K1W0123456	3

Проверим скорость выполнения запроса без индекса:

QueryQuery History

1 SELECT *

2 FROM "Машина"

3 WHERE EXTRACT(YEAR FROM "дата_последнего_техосмотра") >= 2020

4 AND "дефектов" = 0

5 ORDER BY "цена_автомобиля" DESC;

Data Output	Messages	Notifications
Successfully run. Total query runtime: 112 msec. 2 rows affected.		

время выполнения 112 мс.

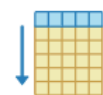
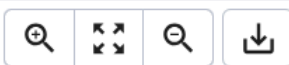
Рассмотрим раздел Explain:

Query Query History

```
1 SELECT *
2 FROM "Машина"
3 WHERE EXTRACT(YEAR FROM "дата_последнего_техосмотра") >= 2020
4 AND "дефектов" = 0
5 ORDER BY "цена_автомобиля" DESC;
```

Data Output Messages Explain X Notifications

Graphical Analysis Statistics



Машина



Sort

Теперь создадим составной индекс для столбцов "Марка_автомобиля"

Query Query History

```
1 CREATE INDEX idx_car_brand_year_price_defects ON "Марка_автомобиля"
2 ("год_выпуска", "id") INCLUDE ("название_модели");
```

Data Output Messages Explain X Notifications

CREATE INDEX

Query returned successfully in 64 msec.

Проверим скорость работы запроса при созданном составном индексе:

Как можно заметить, созданный индекс применяется при выполнении запроса.

Теперь удалим индекс:

```
Query  Query History
1  DROP INDEX IF EXISTS idx_машина_maintenance_price_defects;|

Data Output  Messages  Explain X  Notifications
NOTICE:  index "idx_машина_maintenance_price_defects" does not exist, skipping
DROP INDEX

Query returned successfully in 69 msec.
```

Итоговое время:

Без индекса: 112 мс.

С индексом: 110 мс.

5 Выводы

В рамках данной работы удалось выполнить все поставленные практические задачи:

- Создать запросы и представления на выборку данных к базе данных PostgreSQL (согласно индивидуальному заданию лабораторной работы №2, часть 2 и 3).

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

- 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.

При работе с индексами удалось выяснить, что при большом количестве обрабатываемых данных простые и составные индексы способны ускорить выполнение определённых SQL-команд.

