

**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: Бородин Максим Андреевич

Факультет прикладной информатики

Группа K3241

Направление подготовки 09.03.03 Прикладная информатика

Образовательная программа Мобильные и сетевые технологии 2023

Преподаватель Говорова Марина Михайловна

Санкт-Петербург
2025

Цель работы

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

Программное обеспечение

СУБД PostgreSQL 1X, pgAdmin 4.

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.

Схема базы данных

Модель описывает систему «Расписание занятий и распределение аудиторного фонда» образовательной организации с учётом аудиторий, учебных планов, образовательных программ, направлений подготовки, дисциплин, преподавателей, студенческих групп и назначенных занятий. Ознакомиться с моделью можно на рисунке 1 — для отображения использован генератор ERD-схемы в pgAdmin с удобным оформлением связей между таблицами.

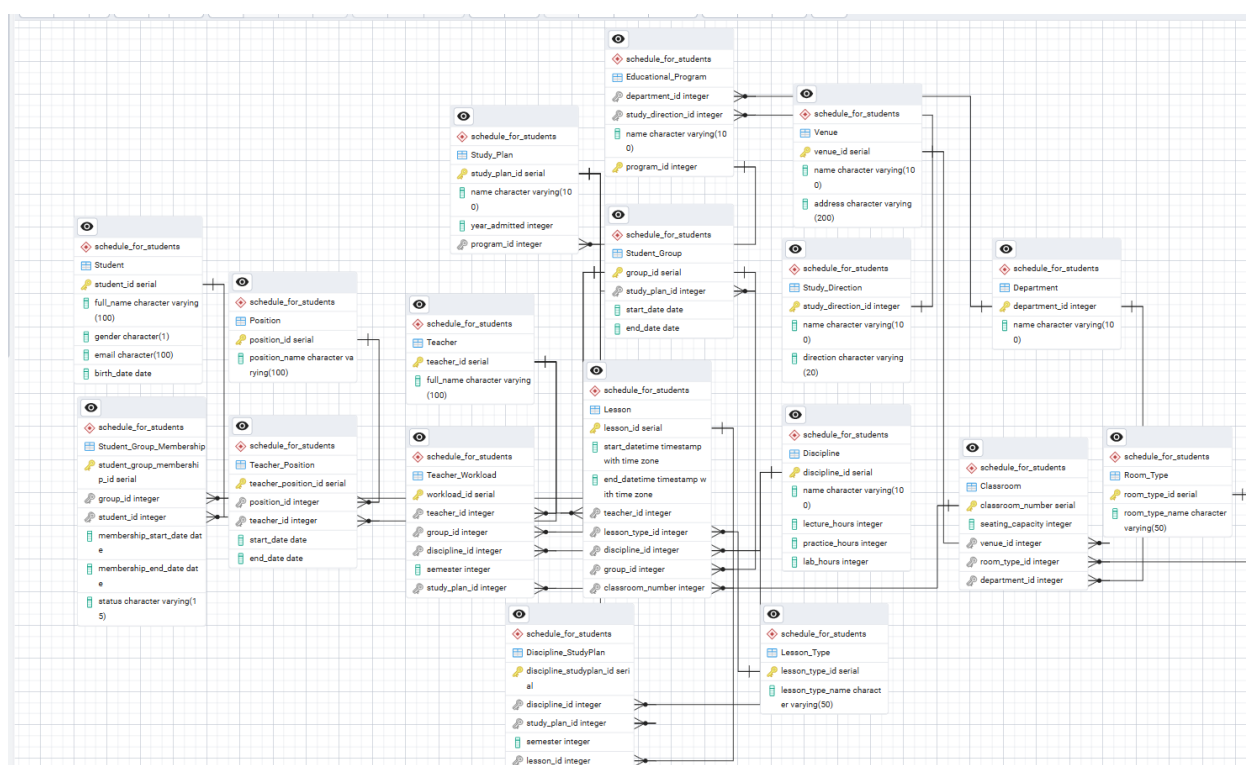


Рисунок 1 – Схема базы данных

Ход работы

1. Запросы к БД

Выполнить запросы согласно индивидуальному заданию, часть 2. В отчете привести формулировку запроса из задания, SQL-команду, скриншот выполнения запроса с результатом

1.1. Вывести загрузку преподавателей в понедельник (в часах) в текущем семестре.

```
SELECT t.full_name,  
       SUM(EXTRACT(EPOCH FROM (l.end_datetime -  
l.start_datetime))/3600) AS monday_hours  
FROM schedule_for_students."Lesson" l  
JOIN schedule_for_students."Teacher" t USING (teacher_id)  
WHERE l.start_datetime::date BETWEEN DATE '2024-09-02' AND DATE  
'2024-09-09'  
      AND EXTRACT(ISODOW FROM l.start_datetime) = 1  
GROUP BY t.full_name  
ORDER BY monday_hours DESC;
```

Результат:

	full_name character varying (100)	monday_hours numeric
1	Ivanova Maria Andreevna	3.0000000000000000
2	Kovalenko Oleg Vechaslavovich	3.0000000000000000
3	Labirov Anatoliy Romanovich	3.0000000000000000

1.2. Найти недельную нагрузку студентов каждой группы (в часах) в текущем семестре.

```
SELECT l.group_id,  
       SUM(EXTRACT(EPOCH FROM (l.end_datetime -  
l.start_datetime))/3600)  
       / (SELECT COUNT(*)  
          FROM  
schedule_for_students."Student_Group_Membership" m  
          WHERE m.group_id = l.group_id  
                AND m.membership_end_date IS NULL  
          ) AS avg_hours_per_student  
FROM schedule_for_students."Lesson" l  
WHERE l.start_datetime::date BETWEEN DATE '2024-09-02' AND  
DATE '2024-09-09'
```

```
GROUP BY l.group_id
ORDER BY l.group_id;
```

Результат:

	group_id integer	avg_hours_per_student numeric
1	1	1.2500000000000000
2	2	1.2500000000000000
3	3	1.1250000000000000

1.3. Вывести список свободных лекционных аудиторий в заданное время.

```
SELECT c.classroom_number
FROM schedule_for_students."Classroom" c
JOIN schedule_for_students."Room_Type" rt ON c.room_type_id =
rt.room_type_id
WHERE rt.room_type_name = 'Lecture Hall'
AND NOT EXISTS (
    SELECT 1
    FROM schedule_for_students."Lesson" l
    WHERE l.classroom_number = c.classroom_number
    AND l.start_datetime < TIMESTAMPTZ '2024-09-05
11:30:00+03'
    AND l.end_datetime > TIMESTAMPTZ '2024-09-05
10:00:00+03'
)
ORDER BY c.classroom_number;
```

Результат:

	classroom_number [PK] integer
1	5

1.4. Вывести количество аудиторий каждого типа.

```
SELECT rt.room_type_name,
COUNT(*) AS rooms_count
```

```
FROM schedule_for_students."Classroom" c
JOIN schedule_for_students."Room_Type" rt ON
c.room_type_id = rt.room_type_id
GROUP BY rt.room_type_name
ORDER BY rooms_count DESC;
```

Результат:

	room_type_name character varying (50)	rooms_count bigint
1	Practice_class	4
2	Lecture Hall	2
3	Labaratory_class	1
4	PE Class	1

1.5. Вывести еженедельное количество часов занятий для каждой группы.

```
SELECT l.group_id,
       SUM(EXTRACT(EPOCH FROM (l.end_datetime -
l.start_datetime)) / 3600) AS total_hours
FROM schedule_for_students."Lesson" l
WHERE l.start_datetime::date BETWEEN DATE '2024-09-02'
AND DATE '2024-09-09'
GROUP BY l.group_id
ORDER BY l.group_id;
```

Результат:

	group_id integer	total_hours numeric
1	1	15.0000000000000000
2	2	15.0000000000000000
3	3	13.5000000000000000



1.6. Найти номера аудиторий каждого типа, имеющих максимальное количество мест.

```

SELECT rt.room_type_name,
       c.classroom_number
FROM schedule_for_students."Classroom" c
JOIN schedule_for_students."Room_Type" rt ON c.room_type_id =
rt.room_type_id
WHERE c.seating_capacity = (
    SELECT MAX(c2.seating_capacity)
    FROM schedule_for_students."Classroom" c2
    WHERE c2.room_type_id = c.room_type_id
)
ORDER BY rt.room_type_name, c.classroom_number;

```

Результат:

	room_type_name character varying (50) 	classroom_number integer 
1	Laboratory_class	3
2	Lecture Hall	1
3	Lecture Hall	5
4	PE Class	8
5	Practice_class	2
6	Practice_class	4
7	Practice_class	6
8	Practice_class	7


1.7. Вывести фамилии преподавателей, которые всегда проводят практические занятия в одной и той же аудитории.

```

SELECT t.full_name
FROM schedule_for_students."Lesson" l
JOIN schedule_for_students."Lesson_Type" lt ON
l.lesson_type_id = lt.lesson_type_id
JOIN schedule_for_students."Teacher" t ON l.teacher_id =
t.teacher_id
WHERE lt.lesson_type_name = 'Practice'
GROUP BY t.full_name
HAVING COUNT(DISTINCT l.classroom_number) = 1
ORDER BY t.full_name;

```

Результат:

	full_name character varying (100) 
1	Emily Winter
2	Ivanova Maria Andreevna
3	Kovalenko Oleg Vechaslavovich
4	Kulimov Vladislav Igorevich
5	PE

2. Представления

Выполнить запросы на создание представлений согласно индивидуальному заданию, часть 3.

2.1. Создать представление: содержащее сведения о расписании по дням для любой группы.

```
DROP VIEW IF EXISTS
schedule_for_students.vw_group_daily_schedule;

CREATE VIEW schedule_for_students.vw_group_daily_schedule AS
SELECT
    l.group_id,
    DATE(l.start_datetime)          AS lesson_date,
    l.start_datetime,
    l.end_datetime,
    lt.lesson_type_name,
    d.name                          AS discipline_name,
    t.full_name                     AS teacher_name,
    c.classroom_number
FROM schedule_for_students."Lesson"    l
JOIN schedule_for_students."Lesson_Type" lt USING
(lesson_type_id)
JOIN schedule_for_students."Discipline" d USING
(discipline_id)
JOIN schedule_for_students."Teacher"    t USING (teacher_id)
JOIN schedule_for_students."Classroom"  c USING
(classroom_number)
WHERE l.start_datetime::date
      BETWEEN DATE '2024-09-02' AND DATE '2024-09-09';
```

Вывод:

```
SELECT *
FROM schedule_for_students.vw_group_daily_schedule
```

```
WHERE group_id = 3
ORDER BY lesson_date, start_datetime;
```

Результат:

Showing rows: 1 to 9 of 1								
	group_id integer	lesson_date date	start_datetime timestamp with time zone	end_datetime timestamp with time zone	lesson_type_name character varying (50)	discipline_name character varying (100)	teacher_name character varying (100)	classroom_number integer
1	3	2024-09-02	2024-09-02 10:00:00+03	2024-09-02 11:30:00+03	Lecture	Mathematical Statistics	Labirov Anatoliy Romanovich	5
2	3	2024-09-02	2024-09-02 12:00:00+03	2024-09-02 13:30:00+03	Practice	Mathematical Statistics	Labirov Anatoliy Romanovich	4
3	3	2024-09-03	2024-09-03 12:00:00+03	2024-09-03 13:30:00+03	Lecture	Engineering Basics	Ivanova Maria Andreevna	5
4	3	2024-09-03	2024-09-03 14:00:00+03	2024-09-03 15:30:00+03	Practice	Engineering Basics	Ivanova Maria Andreevna	2
5	3	2024-09-04	2024-09-04 10:00:00+03	2024-09-04 11:30:00+03	Practice	English language	Emily Winter	4
6	3	2024-09-04	2024-09-04 12:00:00+03	2024-09-04 13:30:00+03	Practice	English language	Emily Winter	4
7	3	2024-09-05	2024-09-05 10:00:00+03	2024-09-05 11:30:00+03	Practice	Computer science	Kovalenko Oleg Vechaslavovich	2
8	3	2024-09-05	2024-09-05 12:00:00+03	2024-09-05 13:30:00+03	Lecture	Computer science	Ivanova Maria Andreevna	1
9	3	2024-09-06	2024-09-06 14:00:00+03	2024-09-06 15:30:00+03	laboratory	Engineering Basics	Pupin Alibaba Sanich	3

2.2. Создать представление: которое будет содержать информацию о средней недельной нагрузке групп по направлениям

```
CREATE OR REPLACE VIEW
schedule_for_students.vw_avg_weekly_load_per_group_direction AS
WITH per_lesson AS (
    SELECT
        group_id,
        EXTRACT(EPOCH FROM (end_datetime - start_datetime))/3600 AS hours,
        date_trunc('week', start_datetime)::date AS week_start
    FROM schedule_for_students."Lesson"
    WHERE start_datetime::date
        BETWEEN DATE '2024-09-02' AND DATE '2024-09-09'
),
per_group_week AS (
    SELECT
        group_id,
        week_start,
        SUM(hours) AS weekly_hours
    FROM per_lesson
    GROUP BY group_id, week_start
),
group_avg AS (
    SELECT
        group_id,
        ROUND(AVG(weekly_hours), 2) AS avg_weekly_hours
    FROM per_group_week
    GROUP BY group_id
)
SELECT
    sd.direction AS direction_code,
```

```

    ga.group_id,
    ga.avg_weekly_hours
FROM group_avg ga
JOIN schedule_for_students."Student_Group"    g USING (group_id)
JOIN schedule_for_students."Study_Plan"      sp USING (study_plan_id)
JOIN schedule_for_students."Educational_Program" ep ON sp.program_id = ep.program_id
JOIN schedule_for_students."Study_Direction" sd ON ep.study_direction_id =
sd.study_direction_id
ORDER BY sd.direction, ga.group_id;

```

Вывод:

```

SELECT *
FROM
schedule_for_students.vw_avg_weekly_load_per_group_direction;

```

Результат:

	direction_code character varying (20) 🔒	group_id integer 🔒	avg_weekly_hours numeric 🔒
1	09.03.04	1	15.00
2	09.03.04	2	15.00
3	09.03.04	3	13.50

3. Выполнить запросы на модификацию данных (INSERT, UPDATE, DELETE) с использованием подзапросов (составить самостоятельно).

3.1. INSERT: Добавление нового преподавателя, выдача ему роли

Формулировка: Вставляет новую запись в таблицу Teacher_Position, назначая преподавателю (добавленному) должность практика с датой сегодня.

Логика:

- Первый подзапрос берёт position_id для «Assistant» из справочника Position.
- Второй подзапрос получает teacher_id преподавателя по полному имени.

```
INSERT INTO schedule_for_students."Teacher_Position" (position_id, teacher_id,
start_date)
VALUES (
  (SELECT position_id
   FROM schedule_for_students."Position"
   WHERE position_name = 'Practice Instructor'),
  (SELECT teacher_id
   FROM schedule_for_students."Teacher"
   WHERE full_name = 'Vasilev Kulebyaka Igorevich'),
  CURRENT_DATE
);
```

Результат:

```
-- Проверка после выполнения
SELECT * FROM schedule_for_students."Teacher_Position";
```

	teacher_position_id [PK] integer	position_id integer	teacher_id integer	start_date date	end_date date
1	1	1	1	2020-09-01	[null]
2	2	2	2	2025-05-14	[null]
3	3	3	3	2025-05-20	[null]
4	4	1	4	2020-09-01	[null]
5	5	2	5	2020-09-01	[null]
6	6	2	4	2020-09-01	[null]
7	7	2	8	2025-05-28	[null]

3.2. UPDATE: Добавить среднюю вместимость к лекционным залам

Формулировка: Обновляет все аудиторные записи типа «Lecture Hall», увеличивая их seating_capacity на среднее значение вместимости таких же залов.

Логика:

- В WHERE подзапросом определяется room_type_id для «Lecture Hall».
- В SET подзапрос рассчитывает среднее (AVG) текущих seating_capacity всех лекционных залов и прибавляет его.

```
UPDATE schedule_for_students."Classroom" c
SET seating_capacity = seating_capacity
+ (
    SELECT ROUND(AVG(c2.seating_capacity))
    FROM schedule_for_students."Classroom" c2
    JOIN schedule_for_students."Room_Type" rt2 ON c2.room_type_id =
rt2.room_type_id
    WHERE rt2.room_type_name = 'Lecture Hall'
)
WHERE c.room_type_id = (
    SELECT room_type_id
    FROM schedule_for_students."Room_Type"
    WHERE room_type_name = 'Lecture Hall'
);
```

Результат:

-- Проверка после выполнения

```
select * from schedule_for_students."Classroom"
where "Classroom".room_type_id = 1;
```

	classroom_number [PK] integer	seating_capacity integer	venue_id integer	room_type_id integer	department_id integer
1	1	120	1	1	1
2	5	120	1	1	1
3	9	80	1	1	1
4	10	50	1	1	1
5	11	65	1	1	1

	classroom_number [PK] integer	seating_capacity integer	venue_id integer	room_type_id integer	department_id integer
1	1	207	1	1	1
2	5	207	1	1	1
3	9	167	1	1	1
4	10	137	1	1	1
5	11	152	1	1	1

3.3. DELETE: Удалить все занятия по физической культуре

Формулировка: Удаляет из таблицы Lesson все пары, связанные с дисциплиной, физическая культура

Логика:

- Подзапрос в WHERE ... IN выбирает discipline_id по имени дисциплины из справочника Discipline.
- Основной DELETE удаляет все строки в Lesson с этим discipline_id.

```
DELETE FROM schedule_for_students."Lesson"
WHERE discipline_id IN (
  SELECT discipline_id
  FROM schedule_for_students."Discipline"
  WHERE name = 'Философия'
);
```

Результат:

```
-- Проверка до выполнения
SELECT *
FROM schedule_for_students."Lesson"
WHERE discipline_id IN (
```

```

SELECT discipline_id
FROM schedule_for_students."Discipline"
WHERE name = 'PE'
);

```

	lesson_id [PK] integer	start_datetime timestamp with time zone	end_datetime timestamp with time zone	teacher_id integer	lesson_type_id integer	discipline_id integer	group_id integer	classroom_number integer
1	16	2024-09-04 12:00:00+03	2024-09-04 13:30:00+03	7	2	5	2	8
2	19	2024-09-04 10:00:00+03	2024-09-04 11:30:00+03	7	2	5	1	8

4. Создание индексов

Выполнить запросы без индекса и создать планы запросов. Выполнить создание индексов. Выполнить запросы с индексами и создать планы запросов. Сравнить время выполнения запросов. Удалить индексы. В отчете отразить все команды, время запросов без индекса и с индексами.

Запрос 1: Создание и проверка индексов для запроса «Загрузка преподавателей в понедельник»

Формулировка: вывести для каждого преподавателя суммарное количество часов занятий, фиксируемое в понедельник недели **2024-09-02...2024-09-09**.

```
EXPLAIN (ANALYZE, BUFFERS)
SELECT t.full_name,
       SUM(EXTRACT(EPOCH FROM (l.end_datetime - l.start_datetime))/3600) AS
monday_hours
FROM   schedule_for_students."Lesson" l
JOIN   schedule_for_students."Teacher" t USING (teacher_id)
WHERE  l.start_datetime::date BETWEEN DATE '2024-09-02' AND DATE '2024-09-09'
       AND EXTRACT(ISODOW FROM l.start_datetime) = 1
GROUP BY t.full_name
ORDER BY monday_hours DESC;
```

Результат:

QUERY PLAN	
text	
1	Sort (cost=2.91..2.91 rows=1 width=250) (actual time=0.153..0.154 rows=3 loops=1)
2	Sort Key: (sum((EXTRACT(epoch FROM (l.end_datetime - l.start_datetime)) / '3600'::numeric))) DESC
3	Sort Method: quicksort Memory: 25kB
4	Buffers: shared hit=2
5	-> GroupAggregate (cost=2.87..2.90 rows=1 width=250) (actual time=0.137..0.142 rows=3 loops=1)
6	Group Key: t.full_name
7	Buffers: shared hit=2
8	-> Sort (cost=2.87..2.87 rows=1 width=234) (actual time=0.123..0.124 rows=6 loops=1)
9	Sort Key: t.full_name
10	Sort Method: quicksort Memory: 25kB
11	Buffers: shared hit=2
12	-> Hash Join (cost=1.74..2.86 rows=1 width=234) (actual time=0.090..0.096 rows=6 loops=1)
13	Hash Cond: (t.teacher_id = l.teacher_id)
14	Buffers: shared hit=2
15	-> Seq Scan on "Teacher" t (cost=0.00..1.08 rows=8 width=222) (actual time=0.014..0.015 rows=8 loops=1)
16	Buffers: shared hit=1
17	-> Hash (cost=1.73..1.73 rows=1 width=20) (actual time=0.063..0.063 rows=6 loops=1)
18	Buckets: 1024 Batches: 1 Memory Usage: 9kB
19	Buffers: shared hit=1
20	-> Seq Scan on "Lesson" l (cost=0.00..1.73 rows=1 width=20) (actual time=0.015..0.048 rows=6 loops=1)
21	Filter: (((start_datetime)::date >= '2024-09-02'::date) AND ((start_datetime)::date <= '2024-09-09'::date) AND (EXTRACT(isodow FROM start_datet...
22	Rows Removed by Filter: 23
23	Buffers: shared hit=1
23	Buffers: shared hit=1
24	Planning:
25	Buffers: shared hit=16
26	Planning Time: 0.996 ms
27	Execution Time: 0.188 ms

Создадим индексы для ускорения этого запроса:

Для оптимизации запроса мы добавили три обычных B-tree индекса:
 idx_lesson_start_date на колонку start_datetime в таблице Lesson, чтобы быстро фильтровать пары по дате;
 idx_lesson_teacher на teacher_id той же таблицы, чтобы ускорить соединение с таблицей преподавателей;
 и idx_teacher_fullname на full_name в таблице Teacher, чтобы облегчить группировку и сортировку по имени преподавателя.

```
CREATE INDEX idx_lesson_start_date
  ON schedule_for_students."Lesson"(start_datetime);
CREATE INDEX idx_lesson_teacher
  ON schedule_for_students."Lesson"(teacher_id);
CREATE INDEX idx_teacher_fullname
  ON schedule_for_students."Teacher"(full_name);
```

Результат:

6	Group Key: t.full_name
7	Buffers: shared hit=2
8	-> Sort (cost=2.87..2.87 rows=1 width=234) (actual time=0.050..0.051 rows=6 loops=1)
9	Sort Key: t.full_name
10	Sort Method: quicksort Memory: 25kB
11	Buffers: shared hit=2
12	-> Hash Join (cost=1.74..2.86 rows=1 width=234) (actual time=0.038..0.041 rows=6 loops=1)
13	Hash Cond: (t.teacher_id = l.teacher_id)
14	Buffers: shared hit=2
15	-> Seq Scan on "Teacher" t (cost=0.00..1.08 rows=8 width=222) (actual time=0.008..0.009 rows=8 loops=1)
16	Buffers: shared hit=1
17	-> Hash (cost=1.73..1.73 rows=1 width=20) (actual time=0.025..0.025 rows=6 loops=1)
18	Buckets: 1024 Batches: 1 Memory Usage: 9kB
19	Buffers: shared hit=1
20	-> Seq Scan on "Lesson" l (cost=0.00..1.73 rows=1 width=20) (actual time=0.010..0.022 rows=6 loops=1)
21	Filter: (((start_datetime)::date >= '2024-09-02'::date) AND ((start_datetime)::date <= '2024-09-09'::date) AND (EXTRACT(isodow FROM start_datet...
22	Rows Removed by Filter: 23
23	Buffers: shared hit=1
24	Planning:
25	Buffers: shared hit=56 read=3
26	Planning Time: 2.050 ms
27	Execution Time: 0.091 ms

Можно видеть, что время выполнения запроса сократилось с 0.188 мс до 0.091 мс.

Вывод

В процессе выполнения лабораторной работы созданы запросы на выборку и модификацию данных с подзапросами, включая добавление, обновление и удаление записей. Разработаны представления и индексы для оптимизации запросов, проведено сравнение времени их выполнения с индексами и без. Выявлено, что индексация эффективна для больших данных, хотя на малом объёме эффект ограничен. Работа укрепила навыки проектирования базы данных, анализа производительности и оптимизации SQL-запросов.