

**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»

Обучающиеся Савченко Анастасия Сергеевна

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

СОДЕРЖАНИЕ

| | |
|--|----|
| ВВЕДЕНИЕ..... | 3 |
| 1 Выполнение..... | 4 |
| 1.1 Схема базы данных (ЛР 3)..... | 4 |
| 1.2 Запросы к БД..... | 4 |
| 1.3 Представления..... | 15 |
| 1.4 Запросы на модификацию данных..... | 17 |
| 1.5 Создание индексов..... | 22 |
| ЗАКЛЮЧЕНИЕ..... | 29 |

ВВЕДЕНИЕ

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

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 посмотреть 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 Выполнение

1.1 Схема базы данных (ЛР 3)

Схема логической модели базы данных, сгенерированной в Generate ERD для лабораторной работы 3.

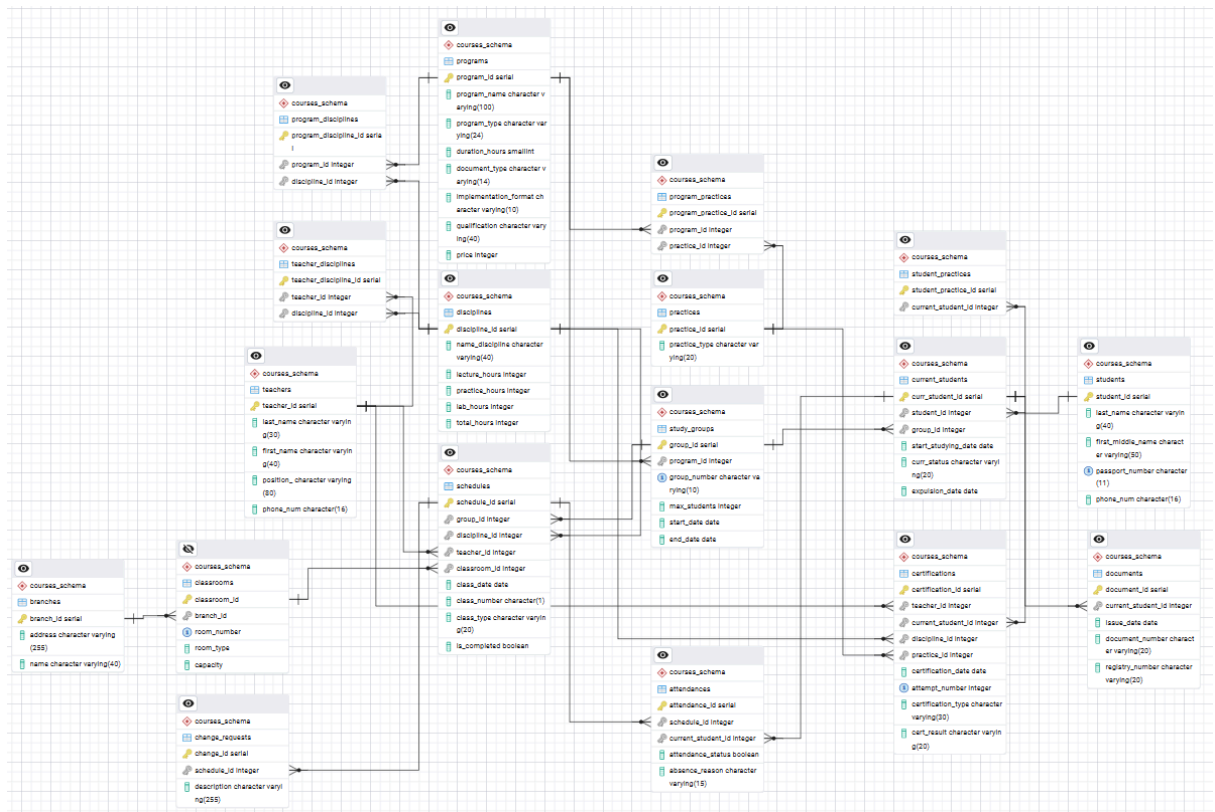


Рисунок 1 – Схема логической модели базы данных, сгенерированная в Generate ERD

1.2 Запросы к БД

Формулировка:

ЛР2: Вариант 7 БД «Курсы»

Задание 2. Создать запросы ...

ВЫПОЛНЕНИЕ:

● Вывести все номера групп и программы, где количество слушателей меньше 10.

SQL-команда:

SELECT

sg.group_number AS "группа",

```

p.program_name AS "Уч программа",
COUNT(cs.curr_student_id) AS student_count
FROM courses_schema.study_groups sg
JOIN courses_schema.programs p ON sg.program_id = p.program_id
LEFT JOIN courses_schema.current_students cs ON sg.group_id =
cs.group_id
GROUP BY sg.group_number, p.program_name
HAVING COUNT(cs.curr_student_id) < 10;

```

Результат:

| | группа character varying (10) 🔒 | Уч программа character varying (100) 🔒 | student_count bigint 🔒 |
|---|------------------------------------|---|---------------------------|
| 1 | ML-01 | Машинное обучение | 4 |
| 2 | SQL-01 | Основы SQL | 5 |
| 3 | DES-01 | Графический дизайн | 5 |
| 4 | JS-01 | Интенсив по JavaScript | 4 |
| 5 | MAN-01 | Менеджмент в образовании | 4 |

Рисунок 2 – Результат выполнения запроса

- Вывести список преподавателей с указанием количества программ, где они преподавали за истекший учебный год.

SQL-команда:

```

SELECT
t.teacher_id AS "ID преподавателя",
t.last_name AS "Фамилия",
t.first_name AS "Имя",
COUNT(DISTINCT sg.program_id) AS "Количество программ" --
DISTINCT чтобы, если преподаватель ведет 3 дисциплины в одной программе,
посчитать только один раз.
FROM courses_schema.teachers t
JOIN courses_schema.teacher_disciplines td ON t.teacher_id = td.teacher_id

```

```

JOIN courses_schema.program_disciplines pd ON td.discipline_id =
pd.discipline_id
JOIN courses_schema.programs p ON pd.program_id = p.program_id
JOIN courses_schema.study_groups sg ON p.program_id = sg.program_id
WHERE sg.start_date >= '2023-09-01'
AND sg.end_date <= '2024-06-30'
GROUP BY t.teacher_id, t.last_name, t.first_name;

```

Результат:

| | ID преподавателя integer | Фамилия character varying (30) | Имя character varying (40) | Количество программ bigint |
|---|-----------------------------|-----------------------------------|-------------------------------|-------------------------------|
| 1 | 1 | Соколов | Андрей Викторович | 1 |
| 2 | 2 | Федорова | Ольга Ивановна | 2 |
| 3 | 3 | Лебедев | Михаил Сергеевич | 1 |
| 4 | 4 | Ковалева | Екатерина Петровна | 1 |
| 5 | 5 | Новиков | Артем Сергеевич | 2 |
| 6 | 6 | Полякова | Анна Владимировна | 2 |

Рисунок 3 – Результат выполнения запроса

- Вывести список преподавателей, которые не проводят занятия на третьей паре ни в один из дней недели.

```

SELECT
    t.teacher_id AS "ID преподавателя",
    t.last_name AS "Фамилия",
    t.first_name AS "Имя"
FROM courses_schema.teachers t
WHERE NOT EXISTS (
    SELECT 1
    FROM courses_schema.schedules s
        JOIN courses_schema.teacher_disciplines td ON s.discipline_id =
td.discipline_id

```

WHERE td.teacher_id = t.teacher_id
AND s.class_number = '3'

);

Результат:

| | ID преподавателя integer | Фамилия character varying (30) | Имя character varying (40) |
|---|-----------------------------|-----------------------------------|-------------------------------|
| 1 | 5 | Новиков | Артем Сергеевич |
| 2 | 6 | Полякова | Анна Владимировна |
| 3 | 4 | Ковалева | Екатерина Петровна |
| 4 | 1 | Соколов | Андрей Викторович |
| 5 | 3 | Лебедев | Михаил Сергеевич |

• Вывести список свободных лекционных аудиторий на ближайший понедельник.

SQL-команда:

```
SELECT r.classroom_id, r.room_number
FROM courses_schema.classrooms r
WHERE r.room_type = 'lecture'
AND r.classroom_id NOT IN (
    SELECT c.classroom_id
    FROM courses_schema.schedules c
    WHERE c.class_date = CURRENT_DATE + ((8 - EXTRACT(DOW
FROM CURRENT_DATE))::int % 7)
)
ORDER BY r.room_number;
```

Результат:

| | classroom_id [PK] integer | room_number character varying (8) |
|---|------------------------------|--------------------------------------|
| 1 | 1 | 101 |
| 2 | 4 | 201 |
| 3 | 6 | 301 |
| 4 | 8 | 401 |

- Вычислить общее количество обучающихся по каждой программе за последний год.

SQL-команда:

```

SELECT
    p.program_id,
    p.program_name AS "Программа",
    COUNT(DISTINCT cs.curr_student_id) AS "Количество обучающихся",
    MIN(sg.start_date) AS "Дата нач первой группы",
    MAX(sg.end_date) AS "Дата оконч последней группы"
FROM
    courses_schema.programs p
LEFT JOIN
    courses_schema.study_groups sg ON p.program_id = sg.program_id
LEFT JOIN
    courses_schema.current_students cs ON sg.group_id = cs.group_id
    AND cs.curr_status = 'обучается'
GROUP BY
    p.program_id, p.program_name
ORDER BY
    "Количество обучающихся" DESC;

```

Результат:

| | program_id [PK] integer | Программа character varying (100) | Количество обучающихся bigint | Дата нач первой группы date | Дата оконч последней группы date |
|---|----------------------------|--------------------------------------|----------------------------------|--------------------------------|-------------------------------------|
| 1 | 1 | Веб-разработка с нуля | 17 | 2023-09-01 | 2024-01-15 |
| 2 | 3 | Педагогика высшей школы | 9 | 2023-10-01 | 2024-03-01 |
| 3 | 4 | Интенсив по Python | 9 | 2023-11-01 | 2023-12-10 |
| 4 | 2 | Анализ данных | 8 | 2023-09-15 | 2023-11-30 |
| 5 | 5 | Основы SQL | 5 | 2024-01-15 | 2024-03-15 |
| 6 | 6 | Графический дизайн | 5 | 2024-02-01 | 2024-05-01 |
| 7 | 7 | Менеджмент в образовании | 4 | 2024-01-20 | 2024-06-20 |
| 8 | 8 | Интенсив по JavaScript | 4 | 2024-03-01 | 2024-05-01 |
| 9 | 9 | Машинное обучение | 4 | 2024-04-01 | 2024-07-01 |

- Вычислить среднюю загруженность компьютерных классов в неделю за последний месяц (в часах).

SQL-команда:

SELECT

ROUND(COUNT(*) * 1.5 / 4.0, 2) AS avg_weekly_load_hours

FROM courses_schema.schedules s

JOIN courses_schema.classrooms r ON s.classroom_id = r.classroom_id

WHERE r.computer_class_yes_no = 'yes'

AND s.class_date BETWEEN (DATE '2023-10-03' - INTERVAL '1 month')AND
DATE '2023-10-03';

Результат:

| | avg_weekly_load_hours numeric |
|---|----------------------------------|
| 1 | 0.38 |

- Найти самые популярные программы за последние 3 года.

SQL-команда:

SELECT

p.program_name,

COUNT(DISTINCT cs.curr_student_id) AS total_students_count

FROM

courses_schema.programs p



JOIN

```

courses_schema.study_groups sg ON p.program_id = sg.program_id
JOIN
courses_schema.current_students cs ON sg.group_id = cs.group_id
WHERE
cs.start_studying_date >= CURRENT_DATE - INTERVAL '3 years'
GROUP BY
p.program_name
ORDER BY
total_students_count DESC
LIMIT 3; -- топ-Х самых популярных программ

```

Результат:

| | program_name character varying (100)  | total_students_count bigint  |
|---|---|--|
| 1 | Веб-разработка с нуля | 21 |
| 2 | Анализ данных | 11 |
| 3 | Интенсив по Python | 10 |

1.3 Представления

- для потенциальных слушателей, содержащее перечень специальностей, изучаемых на них дисциплин и количество часов.

SQL-команда:

```

CREATE OR REPLACE VIEW courses_schema.program_disciplines_view
AS
SELECT
p.program_id,
p.program_name AS "Программа",
p.program_type AS "Тип программы",
d.name_discipline AS "Дисциплина",
d.total_hours AS "Всего часов",

```

```

d.lecture_hours AS "Лекции",
d.practice_hours AS "Практика",
d.lab_hours AS "Лабораторные"
FROM
  courses_schema.programs p
JOIN
  courses_schema.program_disciplines pd ON p.program_id =
pd.program_id
JOIN
  courses_schema.disciplines d ON pd.discipline_id = d.discipline_id
ORDER BY
  p.program_name, d.name_discipline;

```

Чтобы просмотреть результат созданного представления:

```
SELECT * FROM courses_schema.program_disciplines_view;
```

Результат:

| | program_id integer | Программа character varying (100) | Тип программы character varying (24) | Дисциплина character varying (40) | Всего часов integer | Лекции integer | Практика integer | Лабораторные integer |
|----|-----------------------|--------------------------------------|---|--------------------------------------|------------------------|-------------------|---------------------|-------------------------|
| 1 | 2 | Анализ данных | повышение квалификации | Python для анализа данных | 60 | 20 | 30 | 10 |
| 2 | 2 | Анализ данных | повышение квалификации | Базы данных | 40 | 10 | 15 | 15 |
| 3 | 2 | Анализ данных | повышение квалификации | Основы статистики | 40 | 15 | 15 | 10 |
| 4 | 1 | Веб-разработка с нуля | с нуля | HTML/CSS | 40 | 10 | 20 | 10 |
| 5 | 1 | Веб-разработка с нуля | с нуля | JavaScript | 60 | 15 | 25 | 20 |
| 6 | 6 | Графический дизайн | повышение квалификации | Adobe Photoshop | 50 | 10 | 30 | 10 |
| 7 | 8 | Интенсив по JavaScript | интенсив | Продвинутый JavaScript | 60 | 15 | 25 | 20 |
| 8 | 4 | Интенсив по Python | интенсив | Python для анализа данных | 60 | 20 | 30 | 10 |
| 9 | 9 | Машинное обучение | повышение квалификации | Нейронные сети | 70 | 25 | 25 | 20 |
| 10 | 7 | Менеджмент в образовании | профпереподготовка | Управление проектами | 40 | 20 | 20 | 0 |
| 11 | 5 | Основы SQL | с нуля | Основы SQL | 60 | 15 | 25 | 20 |
| 12 | 3 | Педагогика высшей школы | профпереподготовка | Методика преподавания | 50 | 25 | 25 | 0 |
| 13 | 3 | Педагогика высшей школы | профпереподготовка | Педагогика | 50 | 30 | 20 | 0 |

- общий доход по каждой программе за последний год.

SQL-команда:

```

CREATE OR REPLACE VIEW courses_scheme.program_income_last_year
AS
SELECT
  p.program_id,
  p.program_name,

```

```




SUM(p.price) AS total_money
FROM
  courses_schema.programs p
JOIN
  courses_schema.study_groups sg ON p.program_id = sg.program_id
JOIN
  courses_schema.current_students cs ON sg.group_id = cs.group_id
WHERE
  cs.start_studying_date >= DATE '2024-10-03' - INTERVAL '1 year'
GROUP BY
  p.program_id, p.program_name
ORDER BY total_money desc;

```

Чтобы просмотреть результат созданного представления:

```
SELECT * FROM courses_schema.program_income_last_year;
```

Результат:

| | program_id [PK] integer  | program_name character varying (100)  | total_money bigint  |
|---|---|--|--|
| 1 | 3 | Педагогика высшей школы | 200000 |
| 2 | 9 | Машинное обучение | 200000 |
| 3 | 7 | Менеджмент в образовании | 180000 |
| 4 | 6 | Графический дизайн | 160000 |
| 5 | 4 | Интенсив по Python | 150000 |
| 6 | 1 | Веб-разработка с нуля | 125000 |
| 7 | 5 | Основы SQL | 90000 |
| 8 | 8 | Интенсив по JavaScript | 80000 |

Чтобы удалить ранее созданное представление (view):

```
DROP VIEW IF EXISTS courses_schema.program_income_last_year;
```

1.4 Запросы на модификацию данных

Формулировка:

Составить 3 запроса на модификацию данных (INSERT, UPDATE, DELETE) с использованием подзапросов. Изучить графическое представление запросов и посмотреть историю запросов.

ВЫПОЛНЕНИЕ:

-- INSERT с подзапросом

-- Добавим нового студента в группу с самым ранним сроком начала обучения.

```
INSERT INTO courses_schema.current_students (  
    student_id,  
    group_id,  
    start_studying_date,  
    curr_status  
)
```

```
SELECT  
    66, -- ID нового студента в табл students  
    (SELECT group_id  
     FROM courses_schema.study_groups  
     ORDER BY start_date  
     LIMIT 1),  
    CURRENT_DATE,  
    'обучается';
```

До выполнения запроса на модификацию данных

| | curr_student_id [PK] integer | student_id integer | group_id integer | start_studying_date date | curr_status character varying (20) | expulsion_date date |
|----|---------------------------------|-----------------------|---------------------|-----------------------------|---------------------------------------|------------------------|
| 66 | 66 | 57 | 6 | 2024-01-25 | обучается | [null] |
| 67 | 67 | 58 | 7 | 2024-02-10 | обучается | [null] |
| 68 | 68 | 59 | 7 | 2024-02-10 | обучается | [null] |
| 69 | 69 | 60 | 8 | 2024-01-30 | обучается | [null] |
| 70 | 70 | 61 | 8 | 2024-01-30 | обучается | [null] |
| 71 | 71 | 62 | 9 | 2024-03-10 | обучается | [null] |
| 72 | 72 | 63 | 9 | 2024-03-10 | обучается | [null] |
| 73 | 73 | 64 | 10 | 2024-04-10 | обучается | [null] |
| 74 | 74 | 65 | 10 | 2024-04-10 | обучается | [null] |

После выполнения запроса на модификацию данных

| | curr_student_id [PK] integer | student_id integer | group_id integer | start_studying_date date | curr_status character varying (20) | expulsion_date date |
|----|---------------------------------|-----------------------|---------------------|-----------------------------|---------------------------------------|------------------------|
| 66 | 66 | 57 | 6 | 2024-01-25 | обучается | [null] |
| 67 | 67 | 58 | 7 | 2024-02-10 | обучается | [null] |
| 68 | 68 | 59 | 7 | 2024-02-10 | обучается | [null] |
| 69 | 69 | 60 | 8 | 2024-01-30 | обучается | [null] |
| 70 | 70 | 61 | 8 | 2024-01-30 | обучается | [null] |
| 71 | 71 | 62 | 9 | 2024-03-10 | обучается | [null] |
| 72 | 72 | 63 | 9 | 2024-03-10 | обучается | [null] |
| 73 | 73 | 64 | 10 | 2024-04-10 | обучается | [null] |
| 74 | 74 | 65 | 10 | 2024-04-10 | обучается | [null] |
| 75 | 77 | 66 | 1 | 2025-05-29 | обучается | [null] |

-- UPDATE с подзапросом

-- величить цену программ +10%, где количество студентов больше 10.

```

UPDATE courses_schema.programs
SET price = price * 1.1 -- +10%
WHERE program_id IN (
    SELECT p.program_id
    FROM courses_schema.programs p
    JOIN courses_schema.study_groups sg ON p.program_id =
sg.program_id
    JOIN courses_schema.current_students cs ON sg.group_id =
cs.group_id
    GROUP BY p.program_id
    HAVING COUNT(cs.curr_student_id) > 10

```

);

До выполнения запроса на модификацию данных

| | program_id [PK] integer | program_name character varying (100) | program_type character varying (24) | duration_hours smallint | document_type character varying (14) | implementation_format character varying (10) | qualification character varying (40) | price integer |
|---|----------------------------|---|--|----------------------------|---|---|---|------------------|
| 1 | 1 | Веб-разработка с нуля | с нуля | 120 | сертификат | онлайн | Frontend-разработчик | 25000 |
| 2 | 2 | Анализ данных | повышение квалификации | 72 | удостоверение | смешанный | Data Analyst | 35000 |
| 3 | 3 | Педагогика высшей школы | профпереподготовка | 250 | диплом | очно | Преподаватель высшей школы | 40000 |
| 4 | 4 | Интенсив по Python | интенсив | 40 | сертификат | онлайн | Python-разработчик | 15000 |
| 5 | 5 | Основы SQL | с нуля | 60 | сертификат | онлайн | SQL-разработчик | 18000 |
| 6 | 6 | Графический дизайн | повышение квалификации | 90 | удостоверение | смешанный | Графический дизайнер | 32000 |
| 7 | 7 | Менеджмент в образовании | профпереподготовка | 300 | диплом | очно | Менеджер образования | 45000 |
| 8 | 8 | Интенсив по JavaScript | интенсив | 48 | сертификат | онлайн | Специалист по JavaScript | 20000 |
| 9 | 9 | Машинное обучение | повышение квалификации | 120 | удостоверение | смешанный | Data Scientist | 50000 |

После выполнения запроса на модификацию данных

```
UPDATE 2
```

```
Query returned successfully in 67 msec.
```

| | program_id [PK] integer | program_name character varying (100) | price integer | students_count bigint |
|---|----------------------------|---|------------------|--------------------------|
| 1 | 1 | Веб-разработка с нуля | 27500 | 22 |
| 2 | 2 | Анализ данных | 38500 | 11 |
| 3 | 3 | Педагогика высшей школы | 40000 | 10 |
| 4 | 4 | Интенсив по Python | 15000 | 10 |
| 5 | 5 | Основы SQL | 18000 | 5 |
| 6 | 6 | Графический дизайн | 32000 | 5 |
| 7 | 7 | Менеджмент в образовании | 45000 | 4 |
| 8 | 8 | Интенсив по JavaScript | 20000 | 4 |
| 9 | 9 | Машинное обучение | 50000 | 4 |

```
(SELECT
  p.program_id,
  p.program_name,
  p.price,
  COUNT(DISTINCT cs.curr_student_id) AS students_count
FROM
  courses_schema.programs p
LEFT JOIN
```

```

    courses_schema.study_groups sg ON p.program_id = sg.program_id
LEFT JOIN
    courses_schema.current_students cs ON sg.group_id = cs.group_id
GROUP BY
    p.program_id,
    p.program_name,
    p.price
ORDER BY
    students_count DESC;
)

```

-- DELETE с подзапросом

-- Запрос на удаление групп без обучающихся

ЗАПРОС ДЛЯ ПРОВЕРКИ до после

```

(SELECT
    sg.group_id,
    sg.group_number,
    COUNT(cs.curr_student_id) AS students_count
FROM
    courses_schema.study_groups sg
LEFT JOIN
    courses_schema.current_students cs ON sg.group_id = cs.group_id
GROUP BY
    sg.group_id,
    sg.group_number
ORDER BY
    sg.group_id ASC;)

```

До выполнения запроса на модификацию данных

| | group_id [PK] integer | group_number character varying (10) | students_count bigint |
|----|--------------------------|--|--------------------------|
| 1 | 1 | WEB-01 | 12 |
| 2 | 2 | WEB-02 | 10 |
| 3 | 3 | DA-01 | 11 |
| 4 | 4 | PED-01 | 10 |
| 5 | 5 | PY-01 | 10 |
| 6 | 6 | SQL-01 | 5 |
| 7 | 7 | DES-01 | 5 |
| 8 | 8 | MAN-01 | 4 |
| 9 | 9 | JS-01 | 4 |
| 10 | 10 | ML-01 | 4 |
| 11 | 11 | WEB-03 | 0 |
| 12 | 12 | DA-02 | 0 |

SQL-команда:

```
DELETE FROM courses_schema.study_groups
WHERE group_id NOT IN (
    SELECT DISTINCT group_id
    FROM courses_schema.current_students -- Подзапрос
);
```

После выполнения запроса на модификацию данных

DELETE 2

Query returned successfully in 85 msec.

| | group_id [PK] integer | group_number character varying (10) | students_count bigint |
|----|--------------------------|--|--------------------------|
| 1 | 1 | WEB-01 | 12 |
| 2 | 2 | WEB-02 | 10 |
| 3 | 3 | DA-01 | 11 |
| 4 | 4 | PED-01 | 10 |
| 5 | 5 | PY-01 | 10 |
| 6 | 6 | SQL-01 | 5 |
| 7 | 7 | DES-01 | 5 |
| 8 | 8 | MAN-01 | 4 |
| 9 | 9 | JS-01 | 4 |
| 10 | 10 | ML-01 | 4 |


1.5 Создание индексов

Для запроса 1 “Вывести все номера групп и программы, где количество слушателей меньше 10. ”

Выполнение запроса без индексов и получение плана:

```
EXPLAIN ANALYZE
SELECT
    sg.group_number AS "группа",
    p.program_name AS "Уч программа",
    COUNT(cs.curr_student_id) AS student_count
FROM courses_schema.study_groups sg
JOIN courses_schema.programs p ON sg.program_id = p.program_id
LEFT JOIN courses_schema.current_students cs ON sg.group_id =
cs.group_id
GROUP BY sg.group_number, p.program_name
HAVING COUNT(cs.curr_student_id) < 10;
```

Без индексов:

| | QUERY PLAN | |
|----|--|---|
| | text |  |
| 1 | HashAggregate (cost=56.26..68.01 rows=313 width=264) (actual time=0.167..0.173 rows=5 loops=1) | |
| 2 | Group Key: sg.group_number, p.program_name | |
| 3 | Filter: (count(cs.curr_student_id) < 10) | |
| 4 | Batches: 1 Memory Usage: 73kB | |
| 5 | Rows Removed by Filter: 5 | |
| 6 | -> Hash Join (cost=44.75..49.21 rows=940 width=260) (actual time=0.079..0.127 rows=75 loops=1) | |
| 7 | Hash Cond: (sg.program_id = p.program_id) | |
| 8 | -> Hash Right Join (cost=31.15..33.08 rows=940 width=46) (actual time=0.037..0.069 rows=75 loops=1) | |
| 9 | Hash Cond: (cs.group_id = sg.group_id) | |
| 10 | -> Seq Scan on current_students cs (cost=0.00..1.74 rows=74 width=8) (actual time=0.008..0.014 rows=7 ...) | |
| 11 | -> Hash (cost=19.40..19.40 rows=940 width=46) (actual time=0.015..0.015 rows=10 loops=1) | |
| 12 | Buckets: 1024 Batches: 1 Memory Usage: 9kB | |
| 13 | -> Seq Scan on study_groups sg (cost=0.00..19.40 rows=940 width=46) (actual time=0.009..0.010 row ...) | |
| 14 | -> Hash (cost=11.60..11.60 rows=160 width=222) (actual time=0.032..0.032 rows=9 loops=1) | |
| 15 | Buckets: 1024 Batches: 1 Memory Usage: 9kB | |
| 16 | -> Seq Scan on programs p (cost=0.00..11.60 rows=160 width=222) (actual time=0.023..0.024 rows=9 loo ...) | |
| 17 | Planning Time: 0.317 ms | |
| 18 | Execution Time: 0.273 ms | |

Запрос выполняется быстро, но использует полное сканирование таблиц и хэш-джойны, так как индексы не задействованы.

Для ускорения запроса можно создать индексы:

На current_students.group_id

На study_groups.program_id

На study_groups.grou_number

-- Простой индекс по group_id в таблице current_students (используется в соединении)

```
CREATE INDEX idx_cs_group_id ON  
courses_schema.current_students(group_id);
```

Выполнение запроса 1 после составления простого индекса:

| QUERY PLAN | |
|------------|---|
| text | |
| 1 | HashAggregate (cost=16.99..17.93 rows=25 width=264) (actual time=0.155..0.159 rows=5 loops=1) |
| 2 | Group Key: sg.group_number, p.program_name |
| 3 | Filter: (count(cs.curr_student_id) < 10) |
| 4 | Batches: 1 Memory Usage: 24kB |
| 5 | Rows Removed by Filter: 5 |
| 6 | -> Hash Right Join (cost=13.65..16.43 rows=75 width=260) (actual time=0.070..0.108 rows=75 loops=1) |
| 7 | Hash Cond: (cs.group_id = sg.group_id) |
| 8 | -> Seq Scan on current_students cs (cost=0.00..1.75 rows=75 width=8) (actual time=0.009..0.015 rows=75... |
| 9 | -> Hash (cost=13.52..13.52 rows=10 width=260) (actual time=0.055..0.056 rows=10 loops=1) |
| 10 | Buckets: 1024 Batches: 1 Memory Usage: 9kB |
| 11 | -> Hash Join (cost=1.23..13.52 rows=10 width=260) (actual time=0.046..0.053 rows=10 loops=1) |
| 12 | Hash Cond: (p.program_id = sg.program_id) |
| 13 | -> Seq Scan on programs p (cost=0.00..11.60 rows=160 width=222) (actual time=0.022..0.023 rows... |
| 14 | -> Hash (cost=1.10..1.10 rows=10 width=46) (actual time=0.018..0.019 rows=10 loops=1) |
| 15 | Buckets: 1024 Batches: 1 Memory Usage: 9kB |
| 16 | -> Seq Scan on study_groups sg (cost=0.00..1.10 rows=10 width=46) (actual time=0.010..0.013 ... |
| 17 | Planning Time: 0.347 ms |
| 18 | Execution Time: 0.212 ms |

Результат после создания простого индекса:

Время выполнения запроса сократилось с 0,273 мс до 0,212 мс.

Индекс ускорил соединение по group_id в таблице current_students

Удаление простого индекса для запроса 1:

```
DROP INDEX IF EXISTS idx_cs_group_id;
```

-- Составной индекс по (program_id, group_number) в таблице study_groups (соед и группировка)

```
CREATE INDEX idx_sg_program_group ON
courses_schema.study_groups(program_id, group_number);
```

Выполнение запроса 1 после составления составного индекса:

| | QUERY PLAN |
|----|---|
| | text |
| 1 | HashAggregate (cost=16.99..17.93 rows=25 width=264) (actual time=0.140..0.144 rows=5 loops=1) |
| 2 | Group Key: sg.group_number, p.program_name |
| 3 | Filter: (count(cs.curr_student_id) < 10) |
| 4 | Batches: 1 Memory Usage: 24kB |
| 5 | Rows Removed by Filter: 5 |
| 6 | -> Hash Right Join (cost=13.65..16.43 rows=75 width=260) (actual time=0.054..0.083 rows=75 loops=1) |
| 7 | Hash Cond: (cs.group_id = sg.group_id) |
| 8 | -> Seq Scan on current_students cs (cost=0.00..1.75 rows=75 width=8) (actual time=0.007..0.013 rows=75... |
| 9 | -> Hash (cost=13.52..13.52 rows=10 width=260) (actual time=0.042..0.043 rows=10 loops=1) |
| 10 | Buckets: 1024 Batches: 1 Memory Usage: 9kB |
| 11 | -> Hash Join (cost=1.23..13.52 rows=10 width=260) (actual time=0.036..0.041 rows=10 loops=1) |
| 12 | Hash Cond: (p.program_id = sg.program_id) |
| 13 | -> Seq Scan on programs p (cost=0.00..11.60 rows=160 width=222) (actual time=0.016..0.017 rows... |
| 14 | -> Hash (cost=1.10..1.10 rows=10 width=46) (actual time=0.014..0.015 rows=10 loops=1) |
| 15 | Buckets: 1024 Batches: 1 Memory Usage: 9kB |
| 16 | -> Seq Scan on study_groups sg (cost=0.00..1.10 rows=10 width=46) (actual time=0.008..0.009 ... |
| 17 | Planning Time: 0.271 ms |
| 18 | Execution Time: 0.186 ms |

Результат после создания составного индекса:

Время выполнения запроса сократилось с 0,273 мс до 0,186 мс.

Составной индекс ускорил соединение и группировку по полям program_id и group_number в таблице study_groups.

Удаление составного индекса для запроса 1:

```
DROP INDEX IF EXISTS courses_schema.idx_sg_program_group;
```

Проверка удаления индексов:

```
SELECT indexname, tablename
FROM pg_indexes
WHERE schemaname = 'courses_schema';
```

Для запроса 2 “Вывести список преподавателей с указанием количества программ, где они преподавали за истекший учебный год.”

```
EXPLAIN ANALYZE
SELECT
    t.teacher_id AS "ID преподавателя",
    t.last_name AS "Фамилия",
    t.first_name AS "Имя",
    COUNT(DISTINCT sg.program_id) AS "Количество программ" --
DISTINCT чтобы, если преподаватель ведет 3 дисциплины в одной
программе, посчитать только один раз.
FROM courses_schema.teachers t
JOIN courses_schema.teacher_disciplines td ON t.teacher_id =
td.teacher_id
JOIN courses_schema.program_disciplines pd ON td.discipline_id =
pd.discipline_id
JOIN courses_schema.programs p ON pd.program_id = p.program_id
JOIN courses_schema.study_groups sg ON p.program_id =
sg.program_id
WHERE sg.start_date >= '2023-09-01'
AND sg.end_date <= '2024-06-30'
GROUP BY t.teacher_id, t.last_name, t.first_name;
```

Выполнение запроса без индексов и получение плана:

| | QUERY PLAN | |
|----|--|---|
| | text | |
| 8 | -> Hash Join (cost=10.30..49.68 rows=130 width=8) (actual time=0.227..0.236 rows=14 loops=1) | |
| 9 | Hash Cond: (td.discipline_id = pd.discipline_id) | |
| 10 | -> Seq Scan on teacher_disciplines td (cost=0.00..30.40 rows=2040 width=8) (actual time=0.025..0.027 rows=11 loops=1) | |
| 11 | -> Hash (cost=10.14..10.14 rows=13 width=8) (actual time=0.185..0.187 rows=14 loops=1) | |
| 12 | Buckets: 1024 Batches: 1 Memory Usage: 9kB | |
| 13 | -> Nested Loop (cost=0.30..10.14 rows=13 width=8) (actual time=0.095..0.173 rows=14 loops=1) | |
| 14 | -> Nested Loop (cost=0.14..9.41 rows=1 width=8) (actual time=0.057..0.096 rows=9 loops=1) | |
| 15 | -> Seq Scan on study_groups sg (cost=0.00..1.15 rows=1 width=4) (actual time=0.035..0.041 rows=9 loops=1) | |
| 16 | Filter: ((start_date >= '2023-09-01'::date) AND (end_date <= '2024-06-30'::date)) | |
| 17 | Rows Removed by Filter: 1 | |
| 18 | -> Index Only Scan using programs_pkey on programs p (cost=0.14..8.16 rows=1 width=4) (actual time=0.004..0.004 rows=1 loops=9) | |
| 19 | Index Cond: (program_id = sg.program_id) | |
| 20 | Heap Fetches: 9 | |
| 21 | -> Index Only Scan using program_disciplines_program_id_discipline_id_key on program_disciplines pd (cost=0.15..0.63 rows=10 width=8) (actual time=0.007..0.007 rows=14 loops=1) | |
| 22 | Index Cond: (program_id = p.program_id) | |
| 23 | Heap Fetches: 14 | |
| 24 | -> Hash (cost=11.70..11.70 rows=170 width=180) (actual time=0.088..0.088 rows=6 loops=1) | |
| 25 | Buckets: 1024 Batches: 1 Memory Usage: 9kB | |
| 26 | -> Seq Scan on teachers t (cost=0.00..11.70 rows=170 width=180) (actual time=0.063..0.066 rows=6 loops=1) | |
| 27 | Planning Time: 1.137 ms | |
| 28 | Execution Time: 0.534 ms | ✓ Successfully run. Total query runtime: 85 msec. |

--Простой индекс на teacher_disciplines.teacher_id

CREATE INDEX idx_td_teacher_id ON
courses_schema.teacher_disciplines(teacher_id);

Выполнение запроса после составления простого индекса:

| | QUERY PLAN | |
|----|--|--|
| | text | |
| 6 | -> Nested Loop (cost=1.69..15.29 rows=1 width=184) (actual time=0.147..0.271 rows=14 loops=1) | |
| 7 | -> Hash Join (cost=1.55..11.45 rows=1 width=8) (actual time=0.135..0.216 rows=14 loops=1) | |
| 8 | Hash Cond: (pd.discipline_id = td.discipline_id) | |
| 9 | -> Nested Loop (cost=0.30..10.14 rows=13 width=8) (actual time=0.091..0.148 rows=14 loops=1) | |
| 10 | -> Nested Loop (cost=0.14..9.41 rows=1 width=8) (actual time=0.076..0.106 rows=9 loops=1) | |
| 11 | -> Seq Scan on study_groups sg (cost=0.00..1.15 rows=1 width=4) (actual time=0.025..0.031 rows=9 loops=1) | |
| 12 | Filter: ((start_date >= '2023-09-01'::date) AND (end_date <= '2024-06-30'::date)) | |
| 13 | Rows Removed by Filter: 1 | |
| 14 | -> Index Only Scan using programs_pkey on programs p (cost=0.14..8.16 rows=1 width=4) (actual time=0.007..0.007 rows=1 loops=9) | |
| 15 | Index Cond: (program_id = sg.program_id) | |
| 16 | Heap Fetches: 9 | |
| 17 | -> Index Only Scan using program_disciplines_program_id_discipline_id_key on program_disciplines pd (cost=0.15..0.63 rows=10 width=8) (actual time=0.003..0.003 rows=14 loops=1) | |
| 18 | Index Cond: (program_id = p.program_id) | |
| 19 | Heap Fetches: 14 | |
| 20 | -> Hash (cost=1.11..1.11 rows=11 width=8) (actual time=0.027..0.028 rows=11 loops=1) | |
| 21 | Buckets: 1024 Batches: 1 Memory Usage: 9kB | |
| 22 | -> Seq Scan on teacher_disciplines td (cost=0.00..1.11 rows=11 width=8) (actual time=0.016..0.018 rows=11 loops=1) | |
| 23 | -> Index Scan using teachers_pkey on teachers t (cost=0.14..3.80 rows=1 width=180) (actual time=0.002..0.002 rows=1 loops=14) | |
| 24 | Index Cond: (teacher_id = td.teacher_id) | |
| 25 | Planning Time: 1.085 ms | |
| 26 | Execution Time: 0.415 ms | |

Результат после создания простого индекса:

Время выполнения запроса сократилось с 0,534 мс до 0,415 мс.

Индекс позволил ускорить соединение по teacher_disciplines.teacher_id

--Составной индекс на schedules по (discipline_id, class_number)

CREATE INDEX idx_schedules_discipline_class ON
courses_schema.schedules(discipline_id, class_number);

Выполнение запроса после составления составного индекса:

| QUERY PLAN | |
|------------|--|
| text | |
| 7 | -> Hash Join (cost=1.55..11.45 rows=1 width=8) (actual time=0.078..0.115 rows=14 loops=1) |
| 8 | Hash Cond: (pd.discipline_id = td.discipline_id) |
| 9 | -> Nested Loop (cost=0.30..10.14 rows=13 width=8) (actual time=0.050..0.080 rows=14 loops=1) |
| 10 | -> Nested Loop (cost=0.14..9.41 rows=1 width=8) (actual time=0.040..0.055 rows=9 loops=1) |
| 11 | -> Seq Scan on study_groups sg (cost=0.00..1.15 rows=1 width=4) (actual time=0.018..0.021 rows=9 loops=1) |
| 12 | Filter: ((start_date >= '2023-09-01'::date) AND (end_date <= '2024-06-30'::date)) |
| 13 | Rows Removed by Filter: 1 |
| 14 | -> Index Only Scan using programs_pkey on programs p (cost=0.14..8.16 rows=1 width=4) (actual time=0.003..0.003 rows=1 loops=9) |
| 15 | Index Cond: (program_id = sg.program_id) |
| 16 | Heap Fetches: 9 |
| 17 | -> Index Only Scan using program_disciplines_program_id_discipline_id_key on program_disciplines pd (cost=0.15..0.63 rows=10 width=8) (actual time=0.002..0.002 rows=14 loops=1) |
| 18 | Index Cond: (program_id = p.program_id) |
| 19 | Heap Fetches: 14 |
| 20 | -> Hash (cost=1.11..1.11 rows=11 width=8) (actual time=0.017..0.017 rows=11 loops=1) |
| 21 | Buckets: 1024 Batches: 1 Memory Usage: 9kB |
| 22 | -> Seq Scan on teacher_disciplines td (cost=0.00..1.11 rows=11 width=8) (actual time=0.008..0.010 rows=11 loops=1) |
| 23 | -> Index Scan using teachers_pkey on teachers t (cost=0.14..3.80 rows=1 width=180) (actual time=0.001..0.001 rows=1 loops=14) |
| 24 | Index Cond: (teacher_id = td.teacher_id) |
| 25 | Planning Time: 0.804 ms |
| 26 | Execution Time: 0.224 ms |

Результат после создания составного индекса:

Время выполнения запроса сократилось с 0,534 мс до 0,274 мс.

Составной индекс значительно ускорил фильтрацию по полям discipline_id и class_number

Удаление составного индекса:

DROP INDEX IF EXISTS courses_schema.idx_schedules_discipline_class;

Мини-вывод по индексам:

Создание индексов улучшает производительность запросов, особенно при соединениях и фильтрации по нескольким колонкам. Индексы имеет смысл применять для самых частых запросов, подбирая тип индекса и атрибуты, для которых он будет создан, под конкретный запрос, исходя из оптимального соотношения параметров времени выполнения запроса и размера созданного индекса.

Простой индекс помогает ускорить поиск по одному полю, а составной индекс — по нескольким связанным полям.

При небольшом объеме данных улучшения мало заметны, но при росте таблиц индексы становятся критически важными для быстрого выполнения запросов.

После тестирования индексы были удалены, чтобы не влиять на другие операции

ЗАКЛЮЧЕНИЕ

В ходе выполнения лабораторной работы была достигнута цель — овладение практическими навыками работы с PostgreSQL, включая:

- Создание запросов на выборку данных с использованием JOIN, GROUP BY и подзапросов;
- Разработку представлений (VIEW) для упрощения доступа к данным;
- Модификацию данных (INSERT, UPDATE, DELETE) с применением подзапросов;
- Оптимизацию производительности запросов через индексы (простые и составные) с анализом планов выполнения (EXPLAIN ANALYZE).

Выполнены задачи:

- Составлено 7 запросов на выборку данных, включая анализ популярности программ, загруженности аудиторий и списка преподавателей;
- Создано 2 представления: для потенциальных слушателей (дисциплины и часы) и дохода по программам;
- Реализовано 3 запроса на модификацию данных с подзапросами;
- Протестированы индексы для запросов 1 и 2.

Результат лабораторной работы подтверждает освоение работы с инструментом Query Tool в pgAdmin 4 и его возможностями не только выполнять запросы, но и эффективно анализировать их выполнение.