

**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 Дедкова Анастасия Викторовна

Факультет прикладной информатики

Группа К3240

Направление подготовки 09.03.03 Прикладная информатика

Образовательная программа Мобильные и сетевые технологии 2023

Преподаватель Говорова Марина Михайловна

**Санкт-Петербург
2024/2025**

СОДЕРЖАНИЕ

ВВЕДЕНИЕ.....	3
1 Выполнение.....	4
1.1 Схема базы данных (ЛР 3).....	4
1.2 Запросы к базе данных.....	4
1.3 Создание представлений.....	12
1.4 Запросы на модификацию данных.....	14
1.5 Создание индексов.....	18
ЗАКЛЮЧЕНИЕ.....	25
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ.....	27

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:

- 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,
- изучить графическое представление запросов и просмотреть историю запросов,
- 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)

Результат лабораторной работы 3 на схеме логической модели базы данных, сгенерированной в Generate ERD(рисунок 1).

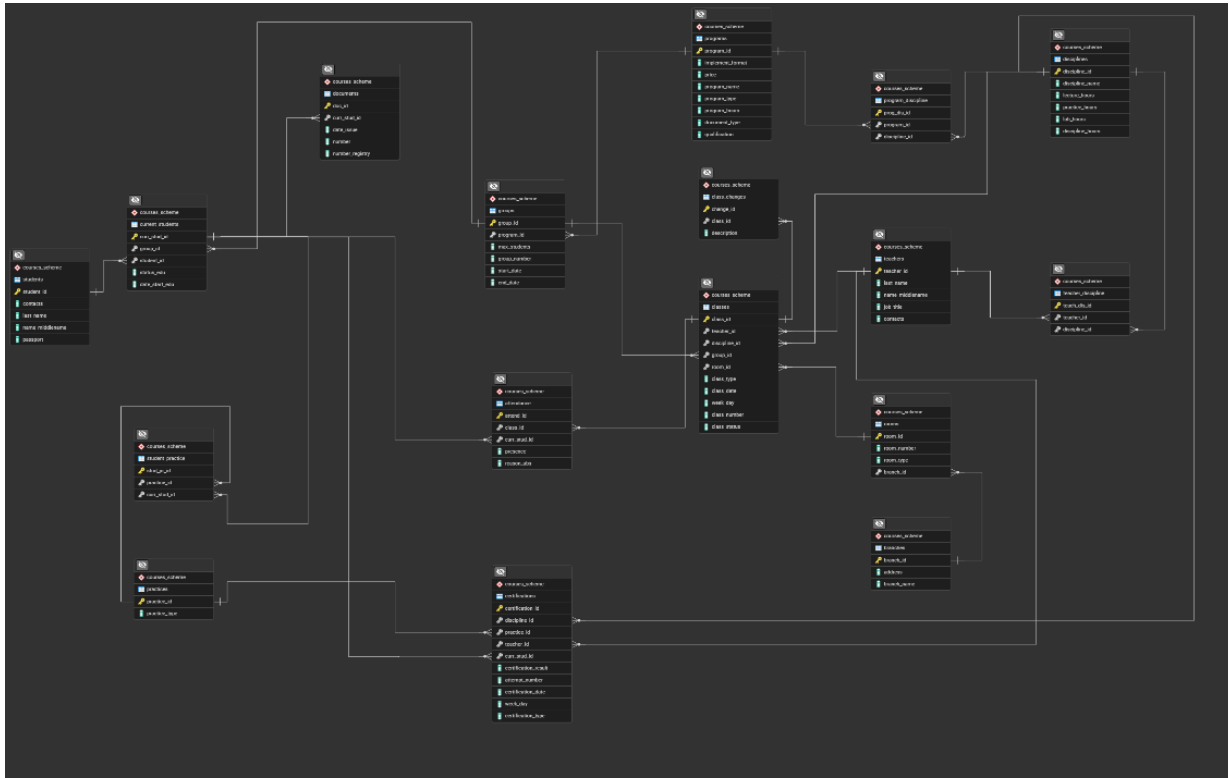


Рисунок 1 – Схема логической модели базы данных, сгенерированная в
Generate ERD [1]

1.2 Запросы к базе данных

Индивидуальное задание 2, согласно варианту 7 лабораторной работы 2.

Создать запросы:

- вывести все номера групп и программы, где количество слушателей меньше 10,

```
SELECT g.group_number, p.program_name
```

FROM courses_scheme.groups g

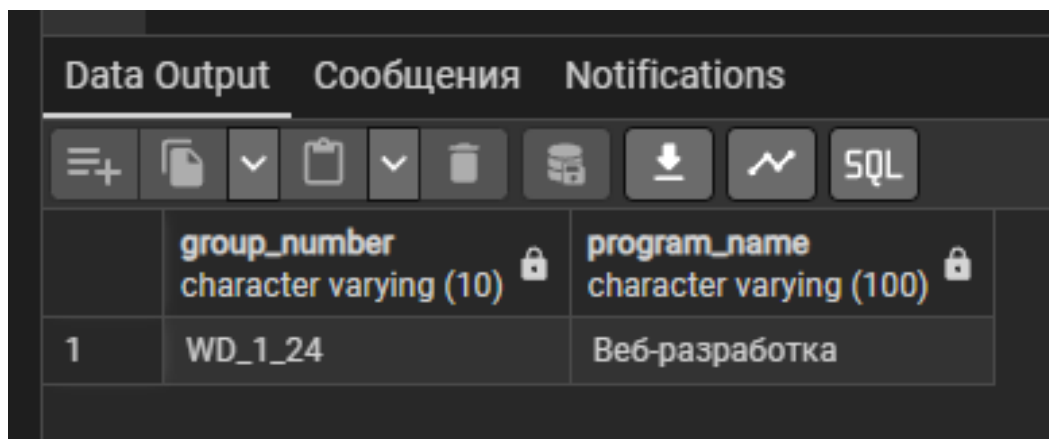
JOIN courses scheme.programs p ON g.program id = p.program id

```
JOIN courses scheme.current_students cs ON g.group_id = cs.group_id
```

GROUP BY g.group_number, p.program_name

HAVING COUNT(cs.curr_stud_id) < 10;

На рисунке 2 результат выполнения SQL команды.



	group_number character varying (10)	program_name character varying (100)
1	WD_1_24	Веб-разработка

Рисунок 2 – Результат выполнения запроса

– вывести список преподавателей с указанием количества программ, где они преподавали за истекший учебный год,

SELECT t.teacher_id,

t.last_name || ' ' || t.name_middlename AS teacher_name,

COUNT(DISTINCT p.program_id) AS program_count

FROM courses_scheme.teachers t

JOIN courses_scheme.teacher_discipline td ON t.teacher_id = td.teacher_id

JOIN courses_scheme.program_discipline pd ON td.discipline_id =
pd.discipline_id

JOIN courses_scheme.programs p ON pd.program_id = p.program_id

JOIN courses_scheme.groups g ON p.program_id = g.program_id

WHERE g.start_date >= '2023-09-01' AND g.end_date <= '2024-06-30'

GROUP BY t.teacher_id, teacher_name

ORDER BY program_count DESC;

На рисунке 3 результат выполнения SQL команды.

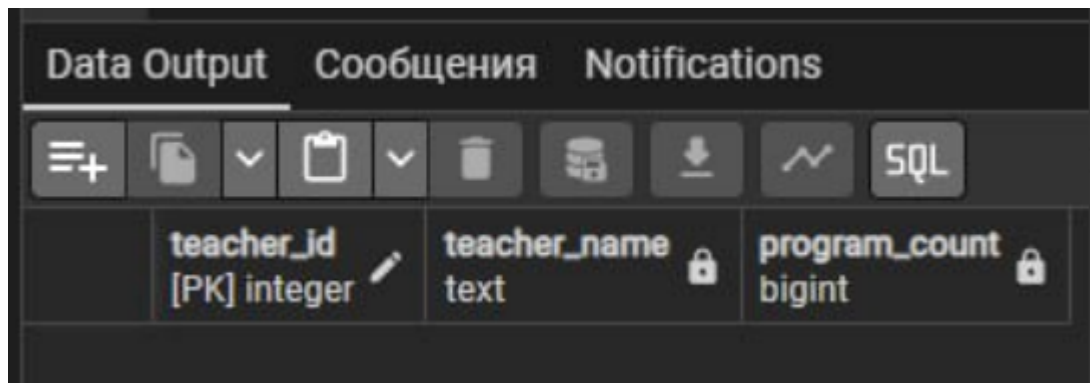


Рисунок 3 – Результат выполнения запроса

Так как база знаний создана с декабря 2024 года, за прошлый учебный год результат выполнения запроса пустой, сделаем запрос за текущий учебный год. Результат на рисунке 4.

```
SELECT t.teacher_id,
       t.last_name || ' ' || t.name_middlename AS teacher_name,
       COUNT(DISTINCT p.program_id) AS program_count
FROM courses_scheme.teachers t
JOIN courses_scheme.teacher_discipline td ON t.teacher_id = td.teacher_id
JOIN  courses_scheme.program_discipline pd ON td.discipline_id =
pd.discipline_id
JOIN courses_scheme.programs p ON pd.program_id = p.program_id
JOIN courses_scheme.groups g ON p.program_id = g.program_id
WHERE g.start_date >= '2024-09-01' AND g.end_date <= '2025-06-30'
GROUP BY t.teacher_id, teacher_name
ORDER BY program_count DESC;
```

	teacher_id [PK] integer	teacher_name text	program_count bigint
1	24	Малинин Алексей Ильич	2
2	10	Павлова Елена Николаевна	2
3	3	Зайцева Екатерина Петровна	1
4	7	Крылов Денис Сергеевич	1
5	8	Бурова Софья Александровна	1
6	9	Михайлов Кирилл Игоревич	1
7	11	Федосеева Дарья Павловна	1
8	12	Прохоров Илья Олегович	1
9	18	Орлова Ксения Андреевна	1
10	1	Терехова Мария Ивановна	1
11	27	Андреева Милена Кирилловна	1
12	2	Королёв Максим Сергеевич	1

Рисунок 4 – Результат выполнения запроса

– вывести список преподавателей, которые не проводят занятия на третьей паре ни в один из дней недели,

```
SELECT t.teacher_id, t.last_name, t.name_middlename
FROM courses_scheme.teachers t
WHERE NOT EXISTS (
    SELECT 1
    FROM courses_scheme.classes c
    WHERE c.teacher_id = t.teacher_id
    AND c.class_number = '3'
)
ORDER BY t.teacher_id;
```

На рисунке 5 результат выполнения SQL команды.

Data Output Сообщения Notifications			
	teacher_id [PK] integer	last_name character varying (30)	name_middlename character varying (40)
1	1	Терехова	Мария Ивановна
2	3	Зайцева	Екатерина Петровна
3	4	Лебедева	Анастасия Андреевна
4	5	Туманов	Роман Викторович
5	6	Степанова	Ольга Дмитриевна
6	7	Крылов	Денис Сергеевич
7	8	Бурова	Софья Александровна
8	10	Павлова	Елена Николаевна
9	11	Федосеева	Дарья Павловна
10	12	Прохоров	Илья Олегович
11	13	Кузьмина	Наталья Юрьевна
12	14	Соловьёв	Сергей Максимович
13	15	Васильева	Валерия Дмитриевна
14	16	Ларионов	Виктор Александрович
15	17	Виноградова	Алина Сергеевна
16	18	Орлова	Ксения Андреевна
17	20	Белкина	Диана Игоревна
18	21	Щербаков	Артём Петрович
19	23	Кузовкова	Виктория Евгеньевна
20	24	Малинин	Алексей Ильич
21	25	Черкасова	Маргарита Сергеевна
22	26	Фролов	Денис Романович
23	27	Андреева	Милена Кирилловна
24	28	Назаров	Ростислав Владимирович
25	30	Толстомятова	Людмила Константиновна

Рисунок 5 – Результат выполнения запроса

– вывести список свободных лекционных аудиторий на ближайший понедельник,

```
SELECT r.room_id, r.room_number
FROM courses_scheme.rooms r
```

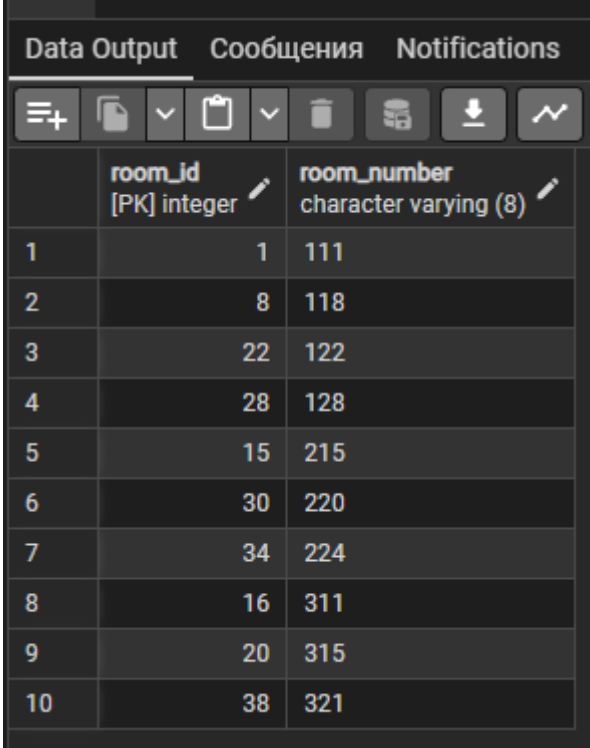


```

WHERE r.room_type = 'лекционная'
AND r.room_id NOT IN (
    SELECT c.room_id
    FROM courses_scheme.classes c
    WHERE c.class_date = '2025-05-19'
)
ORDER BY r.room_number;

```

На рисунке 6 результат выполнения SQL команды.



	room_id [PK] integer	room_number character varying (8)
1	1	111
2	8	118
3	22	122
4	28	128
5	15	215
6	30	220
7	34	224
8	16	311
9	20	315
10	38	321

Рисунок 6 – Результат выполнения запроса

– вычислить общее количество обучающихся по каждой программе за последний год,

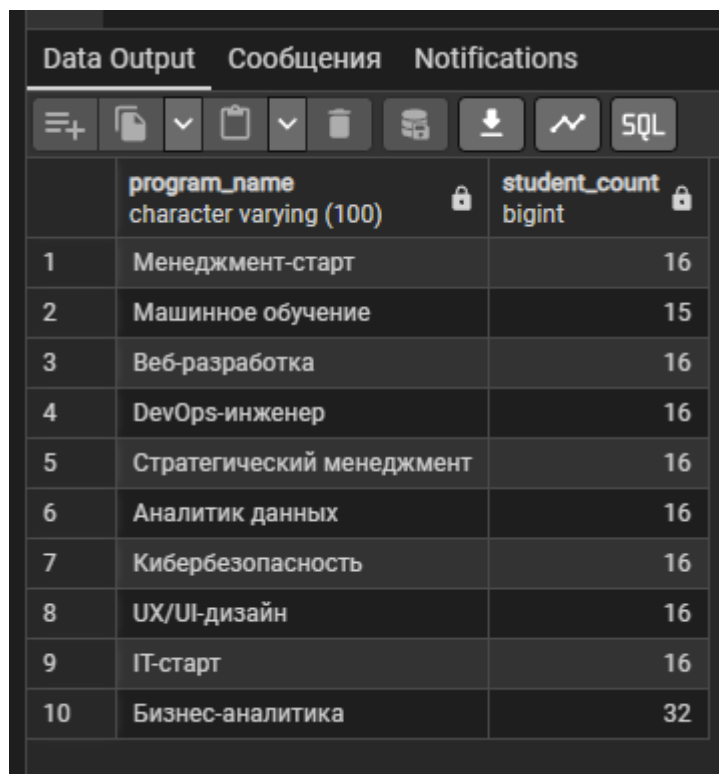
```

SELECT p.program_name, COUNT(s.curr_stud_id) AS student_count
FROM courses_scheme.current_students s
JOIN courses_scheme.groups g ON s.group_id = g.group_id
JOIN courses_scheme.programs p ON g.program_id = p.program_id

```

WHERE s.date_start_edu BETWEEN '2024-09-01' AND '2025-06-30'
GROUP BY p.program_name;

На рисунке 7 результат выполнения SQL команды.



	program_name character varying (100)	student_count bigint
1	Менеджмент-старт	16
2	Машинное обучение	15
3	Веб-разработка	16
4	DevOps-инженер	16
5	Стратегический менеджмент	16
6	Аналитик данных	16
7	Кибербезопасность	16
8	UX/UI-дизайн	16
9	IT-старт	16
10	Бизнес-аналитика	32

Рисунок 7 – Результат выполнения запроса

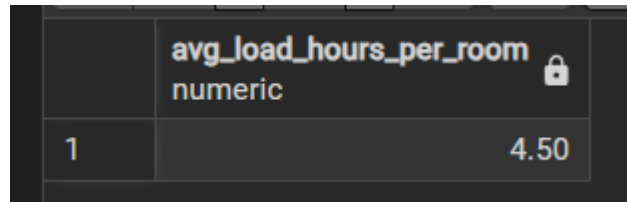
– вычислить среднюю загруженность компьютерных классов в неделю за последний месяц (в часах),

```
SELECT
    ROUND((COUNT(*)::numeric / 4 / COUNT(DISTINCT c.room_id)) * 1.5,
2) AS avg_load_hours_per_room
FROM
    courses_scheme.classes c
JOIN
    courses_scheme.rooms r ON c.room_id = r.room_id
WHERE
    r.room_type = 'компьютерная'
```

AND c.class_status = 'Yes'

AND c.class_date BETWEEN '2025-04-01' AND '2025-04-28';

На рисунке 8 результат выполнения SQL команды.



	avg_load_hours_per_room	numeric
1		4.50

Рисунок 8 – Результат выполнения запроса

– найти самые популярные программы за последние 3 года.

SQL команда:

SELECT

p.program_name,

COUNT(DISTINCT cs.student_id) AS students_count

FROM

courses_scheme.programs p

JOIN

courses_scheme.groups g ON p.program_id = g.program_id

JOIN

courses_scheme.current_students cs ON g.group_id = cs.group_id

WHERE

cs.date_start_edu >= CURRENT_DATE - INTERVAL '3 years'

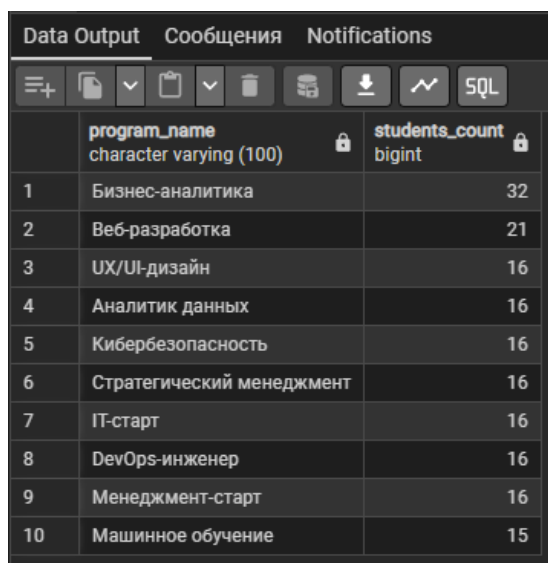
GROUP BY

p.program_name

ORDER BY

students_count DESC;

На рисунке 9 результат выполнения SQL команды.



	program_name character varying (100)	students_count bigint
1	Бизнес-аналитика	32
2	Веб-разработка	21
3	UX/UI-дизайн	16
4	Аналитик данных	16
5	Кибербезопасность	16
6	Стратегический менеджмент	16
7	IT-старт	16
8	DevOps-инженер	16
9	Менеджмент-старт	16
10	Машинное обучение	15

Рисунок 9 – Результат выполнения запроса

1.3 Создание представлений.

Индивидуальное задание 3, согласно варианту 7 лабораторной работы 2.

Создать представление:

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

```
CREATE OR REPLACE VIEW courses_scheme.program_discipline_hours  
AS
```

```
SELECT  
    p.program_id,  
    p.program_name,  
    d.discipline_id,  
    d.discipline_name,  
    (d.lecture_hours + d.practice_hours + d.lab_hours) AS total_hours  
FROM
```

```

courses_scheme.programs p
JOIN
courses_scheme.program_discipline pd ON p.program_id = pd.program_id
JOIN
courses_scheme.disciplines d ON pd.discipline_id = d.discipline_id
ORDER BY
p.program_name,
d.discipline_name;

```

SELECT * FROM courses_scheme.program_discipline_hours;

На рисунке 10 результат выполнения SQL команды.

	program_id integer	program_name character varying (100)	discipline_id integer	discipline_name character varying (40)	total_hours integer
1	8	DevOps-инженер	23	Data Engineering	60
2	8	DevOps-инженер	12	DevOps-практики	60
3	8	DevOps-инженер	46	Lean Six Sigma	60
4	8	DevOps-инженер	25	Автоматизация тестирования	65
5	8	DevOps-инженер	16	Компьютерные сети	65
6	8	DevOps-инженер	50	Нормализация баз данных	50
7	8	DevOps-инженер	17	Облачные вычисления	60
8	8	DevOps-инженер	26	Тестирование ПО	50
9	1	IT-старт	6	Веб-разработка	60
Total rows: 80		Query complete 00:00:00.111			

Рисунок 10 – Результат выполнения запроса

– общий доход по каждой программе за последний год.

```

CREATE OR REPLACE VIEW courses_scheme.program_income_last_year
AS
SELECT
p.program_id,
p.program_name,
COALESCE(p.price, 0) * COALESCE(COUNT(DISTINCT
cs.curr_stud_id), 0) AS total_income_last_year
FROM
courses_scheme.programs p
LEFT JOIN

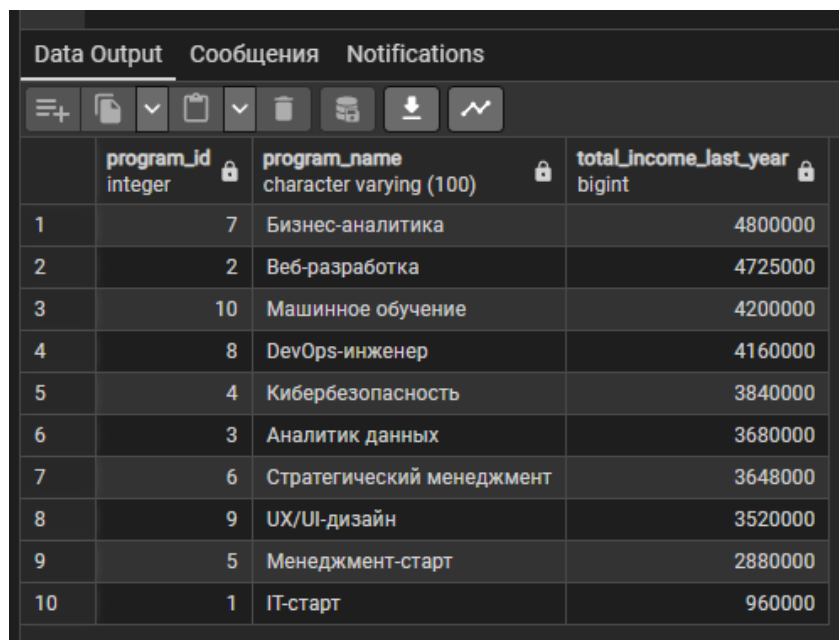
```

```

courses_scheme.groups g ON p.program_id = g.program_id
LEFT JOIN
courses_scheme.current_students cs ON g.group_id = cs.group_id
AND cs.date_start_edu >= CURRENT_DATE - INTERVAL '1 year'
GROUP BY
p.program_id,
p.program_name,
p.price
ORDER BY
total_income_last_year DESC;

```

На рисунке 11 результат выполнения SQL команды.



	program_id integer	program_name character varying (100)	total_income_last_year bigint
1	7	Бизнес-аналитика	4800000
2	2	Веб-разработка	4725000
3	10	Машинное обучение	4200000
4	8	DevOps-инженер	4160000
5	4	Кибербезопасность	3840000
6	3	Аналитик данных	3680000
7	6	Стратегический менеджмент	3648000
8	9	UX/UI-дизайн	3520000
9	5	Менеджмент-старт	2880000
10	1	IT-старт	960000

Рисунок 11 – Результат выполнения запроса

1.4 Запросы на модификацию данных

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

Создан запрос на добавление студента 331 в группу DA_1.

На рисунке 12 выполнение запроса

```

SELECT

    s.student_id,

    st.last_name,

    st.name_middlename,

    s.status_edu

FROM courses_scheme.current_students s

JOIN courses_scheme."groups" g ON s.group_id = g.group_id

JOIN courses_scheme.students st ON s.student_id = st.student_id

WHERE g.group_number = 'DA_1';

```

до выполнения запроса на модификацию данных

	student_id integer	last_name character varying (30)	name_middlename character varying (40)	status_edu courses_scheme.status_type
1	158	Голубева	Аделина Константиновна	студент
2	169	Зимин	Артём Васильевич	студент
3	180	Лапина	Лариса Тимуровна	студент
4	202	Савельев	Борис Олегович	студент
5	213	Беляев	Владимир Александрович	студент
6	224	Киселёв	Даниил Игоревич	студент
7	235	Фролова	Алла Алексеевна	студент
8	246	Чернова	Оксана Игоревна	студент
9	257	Гаврилова	Лидия Григорьевна	студент
10	268	Князев	Тимур Фёдорович	студент
11	279	Николаева	Ирина Викторовна	студент
12	290	Громова	Милана Андреевна	студент
13	301	Андреев	Алексей Сергеевич	студент
14	312	Попов	Иван Алексеевич	студент
15	323	Иванова	Анна Павловна	студент
16	191	Ильина	Снежана Ильинична	в академическом отпуске

Рисунок 12 – Данные до выполнения запроса на модификацию.

Запрос на модификацию:

```

INSERT INTO courses_scheme.current_students (group_id,
student_id, status_edu, date_start_edu)

```

```

SELECT
    group_id,
    331,
    'студент',
    start_date
FROM courses_scheme."groups"
WHERE group_number = 'DA_1'
AND NOT EXISTS (
    SELECT 1 FROM courses_scheme.current_students
    WHERE student_id = 331
    AND group_id = courses_scheme."groups".group_id
)
LIMIT 1;

```

На рисунке 13 результат выполнения SQL команды.

	student_id integer	last_name character varying (30)	name_middlename character varying (40)	status_edu courses_scheme.status_type
9	257	Гаврилова	Лидия Григорьевна	студент
10	268	Князев	Тимур Фёдорович	студент
11	279	Николаева	Ирина Викторовна	студент
12	290	Громова	Милана Андреевна	студент
13	301	Андреев	Алексей Сергеевич	студент
14	312	Попов	Иван Алексеевич	студент
15	323	Иванова	Анна Павловна	студент
16	191	Ильина	Снежана Ильинична	в академическом отпуске
17	331	Дедкова	Анастасия Викторовна	студент

Рисунок 13 – Результат выполнения запроса

Создан запрос на изменение аудитории в расписании. Запись, которую будем изменять рисунок 14.

	class_id [PK] integer	teacher_id integer	discipline_id integer	group_id integer	room_id integer	class_type courses_scheme.class_type	class_date date	week_day courses_scheme.week_day_type	class_number courses_scheme.class_number	class_status courses_scheme.class_status_type
4	4	12	24	2	1	лекционное	2025-02-04	вторник	1	Yes

Рисунок 14 – Запись занятия до изменений


```

UPDATE courses_scheme.classes
SET room_id = 16
WHERE class_id = 4
AND EXISTS (
    SELECT 1 FROM courses_scheme.classes WHERE class_id = 4
)
RETURNING class_id, room_id;

```

На рисунке 15 – результат изменения аудитории

	class_id [PK] integer ↗	teacher_id integer ↗	discipline_id integer ↗	group_id integer ↗	room_id integer ↗	class_type courses_scheme.class_type ↗	class_date date ↗	week_day courses_scheme.week_day_type ↗	class_number courses_scheme.class_number ↗	class_status courses_scheme.class_status_type ↗
4	4	12	24	2	16	лекционное	2025-02-04	вторник	1	Yes

Рисунок 15 – Запись занятия после изменений

Создадим запрос для удаления студента из группы.

На рисунке 16 список студентов группы до выполнения запроса.

	student_id integer 🔒	last_name character varying (30) 🔒	name_middlename character varying (40) 🔒	status_edu courses_scheme.status_type 🔒
9	257	Гаврилова	Лидия Григорьевна	студент
10	268	Князев	Тимур Фёдорович	студент
11	279	Николаева	Ирина Викторовна	студент
12	290	Громова	Милана Андреевна	студент
13	301	Андреев	Алексей Сергеевич	студент
14	312	Попов	Иван Алексеевич	студент
15	323	Иванова	Анна Павловна	студент
16	191	Ильина	Снежана Ильинична	в академическом отпуске
17	331	Дедкова	Анастасия Викторовна	студент

Рисунок 16 – Список группы DA_1 до удаления студента

```

DELETE FROM courses_scheme.current_students cs
WHERE student_id = 331
AND group_id = (

```

```

SELECT group_id
FROM courses_scheme."groups"
WHERE group_number = 'DA_1'
LIMIT 1
)
AND EXISTS (
SELECT 1 FROM courses_scheme.current_students
WHERE student_id = 331
AND group_id = (
SELECT group_id FROM courses_scheme."groups" WHERE
group_number = 'DA_1' LIMIT 1
)
);

```

На рисунке 17 результат выполнения SQL запроса.

	student_id integer	last_name character varying (30)	name_middlename character varying (40)	status_edu courses_scheme.status_type
9	257	Гаврилова	Лидия Григорьевна	студент
10	268	Князев	Тимур Фёдорович	студент
11	279	Николаева	Ирина Викторовна	студент
12	290	Громова	Милана Андреевна	студент
13	301	Андреев	Алексей Сергеевич	студент
14	312	Попов	Иван Алексеевич	студент
15	323	Иванова	Анна Павловна	студент
16	191	Ильина	Снежана Ильинична	в академическом отпуске
17	331	Дедкова	Анастасия Викторовна	студент

Рисунок 17 – Список группы DA_1 после удаления студента

1.5 Создание индексов

Выполним запрос 1 до создания индексов и зафиксируем время (рисунок 18).

Запрос 1:

```

EXPLAIN ANALYZE
SELECT g.group_number, p.program_name
FROM courses_scheme.groups g
JOIN courses_scheme.programs p ON g.program_id = p.program_id
JOIN courses_scheme.current_students cs ON g.group_id =
cs.group_id
GROUP BY g.group_number, p.program_name
HAVING COUNT(cs.curr_stud_id) < 10;

```

Рисунок 18 – План запроса 1 до создания индексов

```
CREATE INDEX idx_group_number ON courses_scheme.groups
(group_number);
```


	QUERY PLAN
	text
1	HashAggregate (cost=22.19..24.44 rows=60 width=256) (actual time=0.132..0.135 rows=1 loops=1)
2	Group Key: g.group_number, p.program_name
3	Filter: (count(cs.curr_stud_id) < 10)
4	Batches: 1 Memory Usage: 40kB
5	Rows Removed by Filter: 11
6	-> Hash Join (cost=14.56..20.84 rows=180 width=260) (actual time=0.047..0.088 rows=180 loops=1)
7	Hash Cond: (cs.group_id = g.group_id)
8	-> Seq Scan on current_students cs (cost=0.00..3.80 rows=180 width=8) (actual time=0.010..0.019...
9	-> Hash (cost=14.41..14.41 rows=12 width=260) (actual time=0.029..0.029 rows=12 loops=1)
10	Buckets: 1024 Batches: 1 Memory Usage: 9kB
11	-> Hash Join (cost=1.27..14.41 rows=12 width=260) (actual time=0.023..0.027 rows=12 loops=1)
12	Hash Cond: (p.program_id = g.program_id)
13	-> Seq Scan on programs p (cost=0.00..12.20 rows=220 width=222) (actual time=0.005..0.0...
14	-> Hash (cost=1.12..1.12 rows=12 width=46) (actual time=0.012..0.013 rows=12 loops=1)
15	Buckets: 1024 Batches: 1 Memory Usage: 9kB
16	-> Seq Scan on groups g (cost=0.00..1.12 rows=12 width=46) (actual time=0.006..0.007 ...
17	Planning Time: 1.156 ms
18	Execution Time: 0.179 ms

Рисунок 20 – План запроса 1 с составными индексами.

Время выполнения запроса до создания индекса 0,264 мс, после создания составного индекса по атрибутам group_id, program_id время выполнения запроса сократилось до 0,179 мс.

Удаление составного индекса для запроса 1:

DROP INDEX IF EXISTS courses_scheme.idx_group_program;

Запрос 2 на вывод номеров групп, из которых были отчислены студенты за период с января по май 2025 г.

Запрос 2:

EXPLAIN ANALYZE

SELECT DISTINCT

g.group_number,

g.group_id

FROM

courses_scheme.current_students cs

JOIN

courses_scheme."groups" g ON cs.group_id = g.group_id

WHERE

cs.status_edu = 'отчислен'

AND cs.date_start_edu BETWEEN '2025-01-01' AND
'2025-05-01';

На рисунке 21 план запроса 2 до создания индексов.

Data Output Сообщения План выполнения X Notifications	
<div> <div>+</div> <div>SQL</div> </div>	
	QUERY PLAN text
1	HashAggregate (cost=22.19..24.44 rows=60 width=256) (actual time=0.138..0.141 rows=1 loops=1)
2	Group Key: g.group_number, p.program_name
3	Filter: (count(cs.curr_stud_id) < 10)
4	Batches: 1 Memory Usage: 40kB
5	Rows Removed by Filter: 11
6	-> Hash Join (cost=14.56..20.84 rows=180 width=260) (actual time=0.055..0.095 rows=180 loops=1)
7	Hash Cond: (cs.group_id = g.group_id)
8	-> Seq Scan on current_students cs (cost=0.00..3.80 rows=180 width=8) (actual time=0.011..0.019...
9	-> Hash (cost=14.41..14.41 rows=12 width=260) (actual time=0.034..0.035 rows=12 loops=1)
10	Buckets: 1024 Batches: 1 Memory Usage: 9kB
11	-> Hash Join (cost=1.27..14.41 rows=12 width=260) (actual time=0.025..0.029 rows=12 loops=1)
12	Hash Cond: (p.program_id = g.program_id)
13	-> Seq Scan on programs p (cost=0.00..12.20 rows=220 width=222) (actual time=0.005..0.0...
14	-> Hash (cost=1.12..1.12 rows=12 width=46) (actual time=0.013..0.013 rows=12 loops=1)
15	Buckets: 1024 Batches: 1 Memory Usage: 9kB
16	-> Seq Scan on groups g (cost=0.00..1.12 rows=12 width=46) (actual time=0.006..0.008 ...
17	Planning Time: 1.126 ms
18	Execution Time: 0.197 ms

Рисунок 21 – План запроса 2 до создания индексов

Создание простого индекса для запроса 2:

CREATE INDEX idx_status_edu ON courses_scheme.current_students
(status_edu);

На рисунке 22 план запроса 2 после создания простого индекса.

Время выполнения запроса до создания индекса 0,197 мс, после создания простого индекса по атрибуту status_edu время выполнения запроса сократилось до 0,091 мс.

Data Output		Сообщения		План выполнения		Notifications	
<div><div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div></div><div>SQL</div></div>							
		QUERY PLAN					
		text					
1	Unique (cost=5.36..5.37 rows=1 width=42) (actual time=0.063..0.064 rows=1 loops=1)						
2	-> Sort (cost=5.36..5.37 rows=1 width=42) (actual time=0.062..0.063 rows=1 loops=1)						
3	Sort Key: g.group_number, g.group_id						
4	Sort Method: quicksort Memory: 25kB						
5	-> Hash Join (cost=4.18..5.35 rows=1 width=42) (actual time=0.054..0.056 rows=1 loops=1)						
6	Hash Cond: (g.group_id = cs.group_id)						
7	-> Seq Scan on groups g (cost=0.00..1.12 rows=12 width=42) (actual time=0.008..0.009 rows=12 loops=1)						
8	-> Hash (cost=4.17..4.17 rows=1 width=4) (actual time=0.038..0.039 rows=1 loops=1)						
9	Buckets: 1024 Batches: 1 Memory Usage: 9kB						
10	-> Index Scan using idx_status_edu on current_students cs (cost=0.14..4.17 rows=1 width=4) (actual time=0.035..0.036 ro...						
11	Index Cond: (status_edu = 'отчислен'::courses_scheme.status_type)						
12	Filter: ((date_start_edu >= '2025-01-01'::date) AND (date_start_edu <= '2025-05-01'::date))						
13	Rows Removed by Filter: 1						
14	Planning Time: 0.977 ms						
15	Execution Time: 0.091 ms						

Рисунок 22 – План запроса 2 после создания простого индекса.

Удаление простого индекса для запроса 2:

`DROP INDEX IF EXISTS courses_scheme.idx_status_edu;`

Создание простого частичного индекса для запроса 2:

`CREATE INDEX idx_status_date_partial ON`

`courses_scheme.current_students (date_start_edu)`

`WHERE status_edu = 'отчислен';`

План запроса после создания простого частичного индекса для запроса 2 (рисунок 23).

Время выполнения запроса до создания индекса 0,197 мс, после создания простого частичного индекса по атрибуту `date_start_edu` время выполнения запроса сократилось до 0,082 мс.

Время выполнения запроса до создания индекса 0,197 мс, после создания составного индекса по атрибутам status_edu, date_start_edu время выполнения запроса сократилось до 0,071 мс.

Удаление составного индекса для запроса 2:

```
DROP INDEX IF EXISTS courses_scheme.idx_status_date_start;
```

Правильное применение индексов помогает увеличить скорость выполнения запросов к базе данных, 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.

ЗАКЛЮЧЕНИЕ

В ходе выполнения лабораторной работы была успешно достигнута цель — овладение практическими навыками создания представлений и запросов на выборку данных к базе данных PostgreSQL, использования подзапросов при модификации данных и индексов. Работа включала следующие ключевые этапы:

- 1) создание запросов на выборку данных, что позволило эффективно извлекать информацию из базы данных на основе различных критериев,
- 2) создание подзапросов для модификации данных (INSERT, UPDATE, DELETE), что обеспечило освоение принципов работы с подзапросами в реальных задачах,
- 3) создание представлений, что позволило организовать удобный доступ к данным для их агрегации и дальнейшего использования,
- 4) изучение графического представления запросов и анализ истории запросов, что улучшило управление запросами и их производительность,

- 5) создание индексов для двух произвольных запросов, что позволило улучшить производительность запросов и сравнить время выполнения с индексами и без них.

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

Выводы: В результате выполнения лабораторной работы были приобретены важные практические навыки в работе с базой данных PostgreSQL, включая создание и модификацию запросов, работу с представлениями и индексами. Также была получена практика в оптимизации запросов для повышения производительности и работе с большими объемами данных. Эти навыки являются необходимыми для эффективного управления данными и создания масштабируемых информационных систем.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Схема логической модели базы данных, сгенерированной в Generate ERD – URL:

<https://drive.google.com/file/d/1VgOtZzphInmDvkbinfkXhg52vFYwghQa/view?usp=sharing>