

Министерство науки и высшего образования Российской Федерации
федеральное государственное автономное образовательное
учреждение высшего образования
“Национальный исследовательский университет ИТМО”

Факультет инфокоммуникационных технологий

ЛАБОРАТОРНАЯ РАБОТА №2

**ЗАПРОСЫ НА ВЫБОРКУ И МОДИФИКАЦИЮ ДАННЫХ, ПРЕДСТАВЛЕНИЯ И
ИНДЕКСЫ В PostgreSQL**

**по дисциплине:
«Проектирование и реализация баз данных»**

Выполнил студент:

Ким Даниил Дмитриевич

Группа №K32391

Преподаватель:

Говорова Марина Михайловна

Санкт-Петербург
2023

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, pgadmin 4

Практическое задание:

1. Создать запросы и представления на выборку данных к базе данных PostgreSQL (согласно индивидуальному заданию, часть 2 и 3).
2. Составить 3 запроса на модификацию данных (INSERT, UPDATE, DELETE) с использованием подзапросов.
3. Изучить графическое представление запросов и просмотреть историю запросов.
4. Создать простой и составной индексы для двух произвольных запросов и сравнить время выполнения запросов без индексов и с индексами. Для получения плана запроса использовать команду EXPLAIN.

Индивидуальное задание (вариант):

Вариант 7. БД «Курсы»

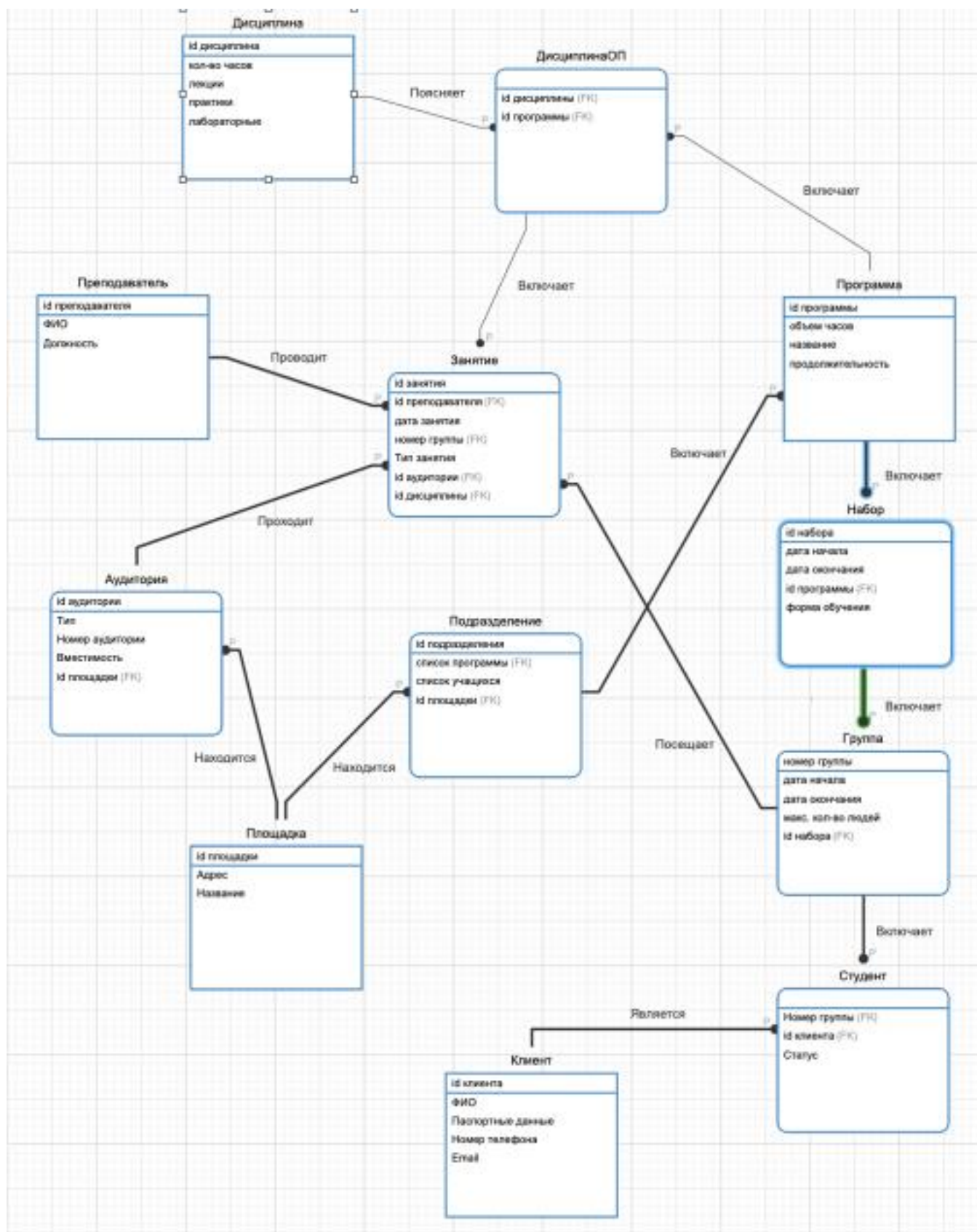
Описание предметной области: Сеть учебных подразделений занимается организацией внебюджетного образования.

Имеется несколько образовательных программ краткосрочных курсов, предназначенных для определенных специальностей, связанных с программным обеспечением ИТ. Каждая программа имеет определенную длительность и свой перечень изучаемых дисциплин. Одна дисциплина может относиться к нескольким программам. На каждую программу может быть набрано несколько групп обучающихся. По каждой дисциплине могут проводиться лекционные, лабораторные/практические занятия и практика определенном объеме часов. По каждой дисциплине и практике проводится аттестация в формате экзамен/дифзачет/зачет.

Подразделение обеспечивает следующие ресурсы: учебные классы, лекционные аудитории и преподавателей. Необходимо составить расписание занятий.

БД должна содержать следующий минимальный набор сведений: Фамилия слушателя. Имя слушателя. Паспортные данные. Контакты. Код программы. Программа. Тип программы. Объем часов. Номер группы. максимальное количество человек в группе (для набора). Дата начала обучения. Дата окончания обучения. Название дисциплины. Количество часов. Дата занятий. Номер пары. Номер аудитории. Тип аудитории. Адрес площадки. Вид занятий (лекционные, практические или лабораторные). Фамилия преподавателя. Имя и отчество преподавателя. Должность преподавателя. Дисциплины, которые может вести преподаватель.

Схема базы данных:



Выполнение:

Запросы к базе данных:

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

```
databases-lab-1=# SELECT g.group_number, p.program_name
FROM lesson._group g
JOIN lesson._set s ON g.id_set = s.id_set
JOIN lesson._program p ON s.id_program = p.id_program
GROUP BY g.group_number, p.program_name
HAVING COUNT(*) < 10;
```

group_number	program_name
K32392	Программакласс
J32391	Программа
L23450	Программасупер
K32391	Русский

(4 rows)

С помощью условия HAVING выбираются только те группы и программы, у которых количество записей меньше 10.

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

```
databases-lab-1=# SELECT t.name_teacher, COUNT(DISTINCT p.id_program) AS program_count
FROM lesson.teacher t
JOIN lesson.lesson l ON t.id_teacher = l.id_teacher
JOIN lesson._group g ON l.group_number = g.group_number
JOIN lesson._set s ON g.id_set = s.id_set
JOIN lesson._program p ON s.id_program = p.id_program
WHERE EXTRACT(YEAR FROM l.lesson_date) = EXTRACT(YEAR FROM CURRENT_DATE) - 1
GROUP BY t.name_teacher;
```

name_teacher	program_count
--------------	---------------

(0 rows)

```
databases-lab-1=# SELECT t.name_teacher, COUNT(DISTINCT p.id_program) AS program_count
FROM lesson.teacher t
JOIN lesson.lesson l ON t.id_teacher = l.id_teacher
JOIN lesson._group g ON l.group_number = g.group_number
JOIN lesson._set s ON g.id_set = s.id_set
JOIN lesson._program p ON s.id_program = p.id_program
WHERE EXTRACT(YEAR FROM l.lesson_date) = EXTRACT(YEAR FROM CURRENT_DATE) - 0
GROUP BY t.name_teacher;
```

name_teacher	program_count
Ирина	1
Николай	1
Татьяна	1
Федор	1

(4 rows)

В этом запросе таблица lesson._group связана с таблицей lesson.lesson через поле group_number, а затем связывается с таблицей lesson._set и таблицей lesson._program, чтобы получить информацию о программе.

Здесь в первом запросе нет ни одного преподавателя, так как внесенные даты не были в

этом временном промежутке. Поэтому чтобы проверить работоспособность запроса я использовал последние даты.

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

```
databases-lab-1=#
databases-lab-1=# SELECT t.name_teacher
FROM lesson.teacher t
WHERE t.id_teacher NOT IN (
    SELECT DISTINCT l.id_teacher
    FROM lesson.lesson l
    WHERE EXTRACT(ISODOW FROM l.lesson_date) = 3 AND l.lesson_type = 'Лекция'
);
 name_teacher
-----
Николай
Федор
Ирина
Татьяна
(4 rows)
```

В этом запросе мы выбираем имена преподавателей из таблицы lesson.teacher, которые имеют id_teacher, отсутствующий во вложенном подзапросе. Во вложенном подзапросе мы выбираем уникальные id_teacher из таблицы lesson.lesson, где lesson_date соответствует третьему дню недели (ISODOW) и lesson_type равен 'Лекция'. Таким образом, результат будет содержать только преподавателей, которые не проводят занятия на третьей паре ни в один из дней недели.

5. Вывести список свободных лекционных аудиторий на ближайший понедельник.

```
databases-lab-1=# SELECT a.audience_number
FROM lesson.audience a
WHERE a._type = 'Лекция'
    AND a.id_audience NOT IN (
        SELECT l.id_audience
        FROM lesson.lesson l
        WHERE l.lesson_date = (
            SELECT MIN(lesson_date)
            FROM lesson.lesson
            WHERE EXTRACT(ISODOW FROM lesson_date) = 1
        )
    );
 audience_number
-----
                201
(1 row)
```

В этом запросе мы выбираем номера аудиторий (audience_number) из таблицы lesson.audience, где тип аудитории (_type) равен 'Лекция', и id_audience отсутствует во вложенном подзапросе. Во вложенном подзапросе мы выбираем id_audience из таблицы lesson.lesson, где lesson_date равна наименьшей дате (MIN(lesson_date)).

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

```
databases-lab-1=# SELECT p.program_name, COUNT(DISTINCT stu.id_student) AS total_students
FROM lesson._program p
INNER JOIN lesson._set s ON p.id_program = s.id_program
INNER JOIN lesson._group g ON s.id_set = g.id_set
INNER JOIN lesson.student stu ON g.group_number = stu.group_number
WHERE EXTRACT(YEAR FROM g.date_from) = EXTRACT(YEAR FROM CURRENT_DATE) - 1
GROUP BY p.program_name;
 program_name | total_students 
-----+-----
(0 rows)
```

```
databases-lab-1=# SELECT p.program_name, COUNT(DISTINCT stu.id_student) AS total_students
FROM lesson._program p
INNER JOIN lesson._set s ON p.id_program = s.id_program
INNER JOIN lesson._group g ON s.id_set = g.id_set
INNER JOIN lesson.student stu ON g.group_number = stu.group_number
WHERE EXTRACT(YEAR FROM g.date_from) = EXTRACT(YEAR FROM CURRENT_DATE) - 0
GROUP BY p.program_name;
 program_name | total_students 
-----+-----
Программа    | 1
Программакласс | 1
Программасупер | 1
Русский      | 1
(4 rows)
```

Здесь я также использовал второй запрос для проверки работы запросы со свежими датами.

В этом запросе мы объединяем таблицы lesson._program, lesson._set, lesson._group и lesson.student по их соответствующим ключам. Затем мы используем условие EXTRACT(YEAR FROM g.date_from) = EXTRACT(YEAR FROM CURRENT_DATE) - 1, чтобы выбрать только группы, начавшие обучение в предыдущем году. Затем с помощью функции агрегации COUNT(DISTINCT s.id_student) мы подсчитываем общее количество уникальных студентов для каждой программы.

7. Найти самые популярные программы за последние 3 года.

```
databases-lab-1=# SELECT p.program_name, COUNT(*) AS program_count
FROM lesson._set s
JOIN lesson._group g ON s.id_set = g.id_set
JOIN lesson._program p ON s.id_program = p.id_program
WHERE EXTRACT(YEAR FROM s.date_from) >= EXTRACT(YEAR FROM CURRENT_DATE) - 3
GROUP BY p.program_name
ORDER BY program_count DESC;
 program_name | program_count 
-----+-----
Программакласс | 1
Программа      | 1
Программасупер | 1
(3 rows)
```

Представления:

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

```
databases-lab-1=# SELECT p.id_program, d.id_discipline, d.volume
FROM lesson._program p
JOIN lesson.discipline_op dp ON p.id_program = dp.id_program
JOIN lesson.discipline d ON dp.id_discipline = d.id_discipline;
```

id_program	id_discipline	volume
123	1	42.5
1	1	42.5
2	2	100.2
3	3	70.8

(4 rows)

Запросы на модификацию данных:

1. Осуществить перевод студента с одного направления на другое

```
databases-lab-1=# SELECT s.id_student, s.group_number, s.student_status, p.id_program
FROM lesson.student s
INNER JOIN lesson._group g ON s.group_number = g.group_number
INNER JOIN lesson._set se ON g.id_set = se.id_set
INNER JOIN lesson._program p ON se.id_program = p.id_program
WHERE s.id_student = 3;
```

id_student	group_number	student_status	id_program
3	K32392	Зачислен	3

(1 row)

```

databases-lab-1=# UPDATE lesson.student
SET group_number = (
    SELECT group_number
    FROM lesson._group
    WHERE id_set = (
        SELECT id_set
        FROM lesson._set
        WHERE id_program = 2
        LIMIT 1
    )
)
WHERE id_student = 3;
UPDATE 1
databases-lab-1=# SELECT s.id_student, s.group_number, s.student_status, p.id_program
FROM lesson.student s
INNER JOIN lesson._group g ON s.group_number = g.group_number
INNER JOIN lesson._set se ON g.id_set = se.id_set
INNER JOIN lesson._program p ON se.id_program = p.id_program
WHERE s.id_student = 3;
 id_student | group_number | student_status | id_program
-----+-----+-----+-----
          3 | L23450      | Зачислен      |          2
(1 row)

```

2. Отчислить студента

```

databases-lab-1=# SELECT * FROM lesson.student;
 id_student | id_client | group_number | student_status
-----+-----+-----+-----
          23 |          42 | K32391      | Зачислен
           2 |           2 | L23450      | Отчислен
           3 |           3 | L23450      | Отчислен
(3 rows)

```

```

databases-lab-1=# SELECT * FROM lesson._group;
 group_number | date_from | date_by | group_size | id_set
-----+-----+-----+-----+-----
K32391       | 2023-06-29 | 2022-09-01 |          40 |          1
L23450       | 2023-07-31 | 2023-02-01 |          15 |          2
K32392       | 2023-08-31 | 2023-03-01 |          25 |          3
J32391       | 2023-06-30 | 2023-01-01 |          18 |          4
(4 rows)

```

После удаления студента и изменения статуса студента:

```

databases-lab-1=# UPDATE lesson.student
SET student_status = 'Отчислен'
WHERE id_student = 23;
UPDATE 1
databases-lab-1=# SELECT * FROM lesson.student;
 id_student | id_client | group_number | student_status
-----+-----+-----+-----
           2 |           2 | L23450      | Отчислен
           3 |           3 | L23450      | Отчислен
          23 |          42 | K32391      | Отчислен
(3 rows)

```



```

databases-lab-1=# UPDATE lesson._group
SET group_size = group_size - 1
WHERE group_number = 'K32391';
UPDATE 1
databases-lab-1=# SELECT * FROM lesson._group;
 group_number | date_from | date_by | group_size | id_set
-----+-----+-----+-----+-----
L23450       | 2023-07-31 | 2023-02-01 | 15 | 2
K32392       | 2023-08-31 | 2023-03-01 | 25 | 3
J32391       | 2023-06-30 | 2023-01-01 | 18 | 4
K32391       | 2023-06-29 | 2022-09-01 | 39 | 1
(4 rows)

```

Создание индексов:

Создание индекса на столбце group_number в таблице lesson.student:

Query Query History

```

1 CREATE INDEX idx_student_group_number ON lesson.student (group_number);
2 |
3

```

Выполнение запроса без использования индекса и создание плана запроса:

Query Query History

```

1
2 EXPLAIN ANALYZE SELECT * FROM lesson.student WHERE group_number = 'J32391'
3

```

Data Output Messages Notifications

	<div> <div>☰+</div> <div>📄</div> <div>▼</div> <div>📋</div> <div>🗑️</div> <div>🗄️</div> <div>⬇️</div> <div>📈</div> </div>
	<div> <div>QUERY PLAN</div> <div>text</div> <div>🔒</div> </div>
1	Seq Scan on student (cost=0.00..1.04 rows=1 width=80) (actual time=0.017..0.017 rows=0 loops=...
2	Filter: ((group_number)::text = 'J32391'::text)
3	Rows Removed by Filter: 3
4	Planning Time: 0.920 ms
5	Execution Time: 0.034 ms

Удаление индекса

```
1
2 DROP INDEX idx_student_group_number;
3
```

Создание индекса на столбце group_number в таблице lesson.student:

Query Query History

1 CREATE INDEX idx_student_group_number1 ON lesson.student (group_number);
2

Data Output Messages Notifications

CREATE INDEX









Query returned successfully in 37 msec.


Выполнение запроса с использованием индекса и создание плана запроса:

Query Query History

1 EXPLAIN ANALYZE SELECT * FROM lesson.student WHERE group_number = 'J32391';
2

Data Output Messages Notifications

	QUERY PLAN	
	text	
1	Seq Scan on student (cost=0.00..1.04 rows=1 width=80) (actual time=0.017..0.018 rows=0 loops=...	
2	Filter: ((group_number)::text = 'J32391'::text)	
3	Rows Removed by Filter: 3	
4	Planning Time: 1.509 ms	
5	Execution Time: 0.036 ms	

Удаление индекса:

Query Query History

1 DROP INDEX idx_student_group_number1;
2

Выводы:

В данной лабораторной работе при выполнении варианта 7 я овладел практическими навыками создания представлений и запросов на выборку данных к базе данных PostgreSQL, использования подзапросов при модификации данных и индексов.