

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**»

Автор: Кахикало К.Р.

Факультет: ИКТ

Группа: K3240

Преподаватель: Говорова М.М.



Санкт-Петербург 2024

Оглавление

Практическое задание:.....	3
Схема базы данных:.....	5
Ход работы:.....	6

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

Оборудование: компьютерный класс.

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.

Вариант 7. БД «Курсы»

Описание предметной области: Сеть учебных подразделений НОУ ДПО 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.

Необходимо хранить информацию по аттестации обучающихся.

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.

Задание 1.1 (ЛР 1 БД). Выполните инфологическое моделирование базы данных системы. (Ограничения задать самостоятельно.)

Задание 1.2. Создайте логическую модель БД, используя ИЛМ (задание 1.1). Используйте необходимые средства поддержки целостности данных в СУБД.

Задание 2. Создать запросы:

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

- Вывести список преподавателей с указанием количества программ, где они преподавали за истекший год.
- Вывести список преподавателей, которые не проводят занятия на третьей паре ни в один из дней недели.
- Вывести список свободных лекционных аудиторий на ближайший понедельник.
- Вычислить общее количество обучающихся по каждой программе за последний год.
- Вычислить среднюю загруженность компьютерных классов в неделю за последний месяц (в часах).
- Найти самые популярные программы за последние 3 года.

Задание 3. Создать представление:

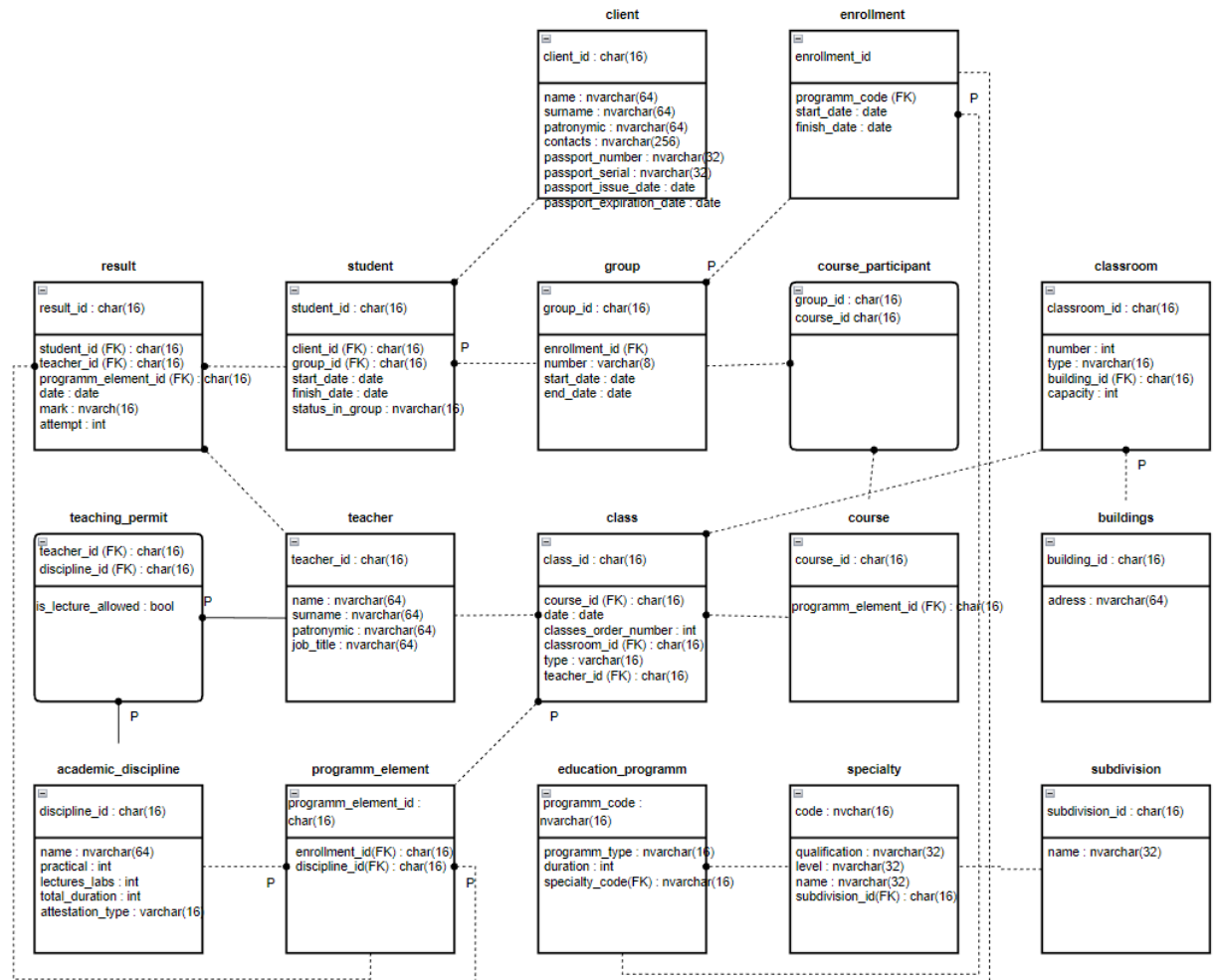
- для потенциальных слушателей, содержащее перечень специальностей, изучаемых на них дисциплин и количество часов;
- общих доход по каждой программе за последний год.

Задание 4. Создать хранимые процедуры:

- Для получения расписания занятий для групп на определенный день недели.
- Записи на курс слушателя.
- Получения перечня свободных лекционных аудиторий на любой день недели. Если свободных аудиторий не имеется, то выдать соответствующее сообщение.

Задание 5. Создать необходимые триггеры.

Схема базы данных:



Ход работы:

Я начал выполнять задания по порядку. Мне нужно было получить программы у которых количество слушателей меньше 10. Самая первая версия была с тремя JOIN, а потом WHERE с subquery, но, предполагаю, что это было непроизводительно, так как subquery коррелированный. Поэтому я совершил преждевременную оптимизацию и перешёл к следующему варианту. Я попробовал три JOIN, а потом GROUP BY, но это требовало кучи колонок в GROUP BY, непроизводительно (много колонок в GROUP BY и лишние JOIN). Этот вариант я решил немного упростить. Если бы оптимизатор в Postgres был умнее, то он бы понял, что между group_id и enrollment_id и programm_code есть функциональная зависимость, и он бы смог оптимизировать запрос, тогда я бы смог не использовать subquery, а использовал бы три JOIN и GROUP BY только с одним столбцом.

```
SELECT number, education_programm.programm_code
FROM (SELECT "group".group_id, number, "group".enrollment_id as enrollment_id_short
      FROM "group"
      JOIN student ON "group".group_id = student.group_id
      GROUP BY "group".group_id
      HAVING COUNT(student.student_id) < 10
     )
JOIN enrollment ON enrollment_id_short = enrollment.enrollment_id
JOIN      education_programm      ON      enrollment.programm_code      =
education_programm.programm_code;
```

	number character varying (8) 🔒	programm_code character varying (16) 🔒
1	K3242	PROG002
2	K3241	PROG001


Для того чтобы вывести список преподавателей с указанием количества программ, где они преподавали за истекший год. Так как нужно было выводить даже учителей, у которых не было программ, составил такой запрос с LEFT JOIN, вместо INNER JOIN:

```
SELECT teacher.teacher_id, COUNT(class)
FROM teacher
      LEFT JOIN class ON teacher.teacher_id = class.teacher_id AND extract(year from
class.date) = extract(year from CURRENT_TIMESTAMP) - 1
      LEFT JOIN course ON class.course_id = course.course_id
      LEFT JOIN programm_element ON course.programm_element_id =
programm_element.programm_element_id
      LEFT JOIN enrollment ON programm_element.enrollment_id =
enrollment.enrollment_id
GROUP BY enrollment.programm_code, teacher.teacher_id;
```

	teacher_id [PK] uuid	count bigint
1	11914ea8-82fb-46c8-bef5-2e376ac2df97	0
2	04645e63-c4a3-4da4-9ce6-b8ed4be50b...	0
3	b6b06ff0-eba6-46f5-8263-8a6e1b6b6bb0	0
4	17a5c7a6-fb30-4ff9-b4d5-63fd6c20183e	1
5	72a13add-efca-49f0-bc53-fbb7e7be2d8c	0
6	6c0bbc51-93f8-479e-a828-2c71e08de423	0
7	42cea071-d15a-4a49-a0dd-c84d9c19e5...	0
8	bc611abc-039c-4eea-b95f-3f6545d67c3f	0
9	e89c4be9-4cbe-46f1-8333-870d697b400b	0
10	064f78ca-718e-4c20-9bf8-3abee87114ca	1
11	000331af-1293-42eb-9dab-f7e000711a9d	0

Вывести преподавателей, которые никогда не проводили и не планируют проводить занятия на 3 паре легко:

```
SELECT DISTINCT (class.teacher_id)
FROM class
WHERE classes_order_number <> 3;
```

	teacher_id uuid 
1	17a5c7a6-fb30-4ff9-b4d5-63fd6c20183e
2	064f78ca-718e-4c20-9bf8-3abee87114ca



Для вывода свободных лекционных аудиторий на ближайший понедельник, пришлось много работать с датами. 7 - EXTRACT(DOW FROM CURRENT_DATE) - дней до конца недели. +1 чтобы сдвинуть на понедельник. || day позволяет превратить numeric в string с day, чтобы потом скастить в interval.

```
SELECT classroom.classroom_id
FROM classroom
WHERE NOT EXISTS(SELECT * from class WHERE classroom.classroom_id =
class.classroom_id AND class.date = CURRENT_DATE + ((7 - EXTRACT(DOW FROM
CURRENT_DATE) + 1) || ' day')::interval) AND type = 'лекционная';
```

	classroom_id [PK] uuid	
1	4bd4ffe8-f48f-4375-b99c-191ee70e0bcd	
2	5c086288-5405-44a2-8d9e-d77a3eccd0...	
3	9df8d337-ec5a-4e49-872e-0e3cc0a0c01b	
4	978e4cd8-5e50-4ea2-b2ca-fc5a7d7052d8	


Вычислить общее количество обучающихся по каждой программе за последний год.

```
SELECT education_programm.programm_code, COUNT(student)
FROM education_programm
LEFT JOIN enrollment ON education_programm.programm_code =
enrollment.programm_code
LEFT JOIN "group" ON enrollment.enrollment_id = "group".enrollment_id AND
extract(year from CURRENT_TIMESTAMP) IN (extract(year from "group".start_date),
extract(year from "group".end_date))
LEFT JOIN student ON "group".group_id = student.group_id
GROUP BY education_programm.programm_code;
```

	programm_code [PK] character varying (16) 	count bigint 
1	PROG002	3
2	PROG003	0
3	PROG005	0
4	PROG001	9
5	PROG004	0


Вычислить среднюю загруженность компьютерных классов в неделю за последний месяц (в часах). Так как у меня нет такого типа аудитории как компьютерный класс, заменил на лекционную аудиторию. Чтобы получить количество часов из количества среднего количества пар, домножил на длительность академического часа.

```
SELECT AVG(load) * 1.5 as average_load
FROM (SELECT COUNT(class.classroom_id) as load
      FROM classroom
      LEFT JOIN class ON classroom.classroom_id = class.classroom_id AND extract(month
from CURRENT_TIMESTAMP) = extract(month from class.date) AND extract(year from
CURRENT_TIMESTAMP) = extract(year from class.date)
      WHERE classroom.type = 'лекционная'
      GROUP BY classroom.classroom_id);
```

	average_load numeric	
1	0.000000000000000000000000	

Для вывода самых популярных программ за последние 3 года отсортировал программы по общему количеству слушателей за 3 года и вывел 3 самые популярные.

```
SELECT education_programm.programm_code, COUNT(student) as students_count
FROM education_programm
LEFT JOIN enrollment ON education_programm.programm_code =
enrollment.programm_code
LEFT JOIN "group" ON enrollment.enrollment_id = "group".enrollment_id AND
extract(year from CURRENT_TIMESTAMP) >= extract(year from "group".start_date) AND
(extract(year from CURRENT_TIMESTAMP) - 2) <= extract(year from "group".end_date)
LEFT JOIN student ON "group".group_id = student.group_id
GROUP BY education_programm.programm_code
ORDER BY students_count DESC
LIMIT 3;
```

	programm_code [PK] character varying (16) 	students_count bigint 
1	PROG001	9
2	PROG002	3
3	PROG003	0

Дальше я создал View для потенциальных слушателей, содержащее перечень специальностей, изучаемых на них дисциплин и количество часов. Я реализовал то что запрашивалось в задании, но считаю что это некорректно и стоило выводить данные не для специальностей, а для enrollment, так как программа может меняться каждый код и, соответственно, каждый год в ней будут разные дисциплины.

```
CREATE VIEW programm_disciplines AS
SELECT education_programm.programm_code, academic_discipline.name as discipline_name,
academic_discipline.total_duration
FROM education_programm
      JOIN enrollment ON education_programm.programm_code =
enrollment.programm_code
      JOIN programm_element ON enrollment.enrollment_id =
programm_element.enrollment_id
      JOIN academic_discipline ON programm_element.discipline_id =
academic_discipline.discipline_id
ORDER BY education_programm.programm_code;
```

	programm_code character varying (16) 🔒	discipline_name character varying (32) 🔒	total_duration integer 🔒
1	PROG001	Информатика	100
2	PROG002	Математический анализ	60
3	PROG001	Математический анализ	60
4	PROG002	Химия	50

Общий доход я считать не могу, так как в системе нет никаких финансовых данных. Вместо этого я считаю количество студентов отдельно для каждой программы за последний код, а дальше потребитель этих данных, сам мог бы вычислить общий доход, умножив на стоимость обучения, при условии что она фиксирована.

```
CREATE VIEW students_per_programm AS
SELECT education_programm.programm_code, COUNT(student) as students_count
FROM education_programm
LEFT JOIN enrollment ON education_programm.programm_code =
enrollment.programm_code
LEFT JOIN "group" ON enrollment.enrollment_id = "group".enrollment_id AND
extract(year from CURRENT_TIMESTAMP) IN (extract(year from "group".start_date),
extract(year from "group".end_date))
LEFT JOIN student ON "group".group_id = student.group_id
GROUP BY education_programm.programm_code;
```

	programm_code character varying (16) 🔒	students_count bigint 🔒
1	PROG002	3
2	PROG003	0
3	PROG005	0
4	PROG001	9
5	PROG004	0

Дальше 3 запроса на модификацию данных. Первый из них создаёт новую попытку сдачи экзаменов/сессии для каждой существующей, если это возможно. Расширение используется для генерации новых UUID.

CREATE EXTENSION IF NOT EXISTS "uuid-ossf";

```
INSERT INTO result (result_id, student_id, teacher_id, programm_element_id, attempt)
SELECT uuid_generate_v4(), student_id, teacher_id, programm_element_id, attempt + 1
FROM result
WHERE attempt < 2 AND NOT EXISTS(SELECT * FROM result inner_result WHERE
result.student_id = inner_result.student_id AND result.programm_element_id =
inner_result.programm_element_id AND result.attempt + 1 = inner_result.attempt);
```

Было:

	result_id [PK] uuid	student_id uuid	teacher_id uuid	programm_element_id uuid	date date	mark mark	attempt integer
1	faa8f566-3918-47fa-a4ac-ef8d85c17da2	9101da2f...	064f78ca...	002f0e3b-5821-4bdf-82b1-443d63a49245	2024-01-01	2	0
2	faa8f566-3918-47fa-a4ac-ef8d85c17da3	9101da2f...	064f78ca...	002f0e3b-5821-4bdf-82b1-443d63a49245	2024-01-01	4	1

Стало:

	result_id [PK] uuid	student_id uuid	teacher_id uuid	programm_element_id uuid	date date	mark mark	attempt integer
1	d26f836f-dbad-4965-b1da-62f273f8d04a	9101da2f...	064f78ca...	002f0e3b-5821-4bdf-82b1-443d63a49245	[null]	[null]	2
2	faa8f566-3918-47fa-a4ac-ef8d85c17da2	9101da2f...	064f78ca...	002f0e3b-5821-4bdf-82b1-443d63a49245	2024-01-01	2	0
3	faa8f566-3918-47fa-a4ac-ef8d85c17da3	9101da2f...	064f78ca...	002f0e3b-5821-4bdf-82b1-443d63a49245	2024-01-01	4	1

Следующий запрос переводит всех студентов у которых есть хотя один несданных зачёт или экзамен в состояние “отчислен”.

```
UPDATE student
SET status_in_group = 'отчислен'
WHERE EXISTS(SELECT * FROM result WHERE result.student_id = student.student_id
AND mark IS NULL);
```

Было:

	student_id [PK] uuid	client_id uuid	group_id uuid	start_date date	finish_date date	status_in_group student_status
1	00495bd0-99de-436b-8456-6fbbb70248...	3b25f691...	4f0ee28b...	2023-09-01	2024-07-01	учится
2	0456f9f0-b753-4ad1-9cae-de8d7fbc9490	5c6ee5d3...	4f0ee28b...	2023-09-01	2024-07-01	учится
3	0ae4485d-faf1-4475-bb04-ad480075a2...	18ab5bd9...	4f0ee28b...	2023-09-01	2024-07-01	отчислен
4	3b8aaf62-d4aa-48f2-9cfd-f1a9c90c0ad1	132259da...	4f0ee28b...	2023-09-01	2024-07-01	отчислен
5	405e00b3-37e4-49e0-a191-929de9d72a...	7e8f59c1-...	4f0ee28b...	2023-09-01	2024-07-01	учится
6	6a15d442-c3fe-41cc-95ec-76f7c6583f87	19bc6de0...	4f0ee28b...	2023-09-01	2024-07-01	отчислен
7	9101da2f-9b03-4f3c-a569-f5708fa8c6c5	24fd3f29-...	4f0ee28b...	2023-09-01	2024-07-01	учится
8	a4b34641-1125-4ba5-8c8f-c6bebe87b4...	4acca545...	4f0ee28b...	2023-09-01	2024-07-01	учится
9	befeb349-8417-4759-912f-aeb8397174...	8bb9d041...	8a472103...	2023-09-01	2024-07-01	учится
10	c1c6b124-4291-44b9-9246-4e3781a8b4...	20cd7ef1-...	8a472103...	2023-09-01	2024-07-01	учится
11	d5d7cd03-d086-4a4e-9c2b-b1c14e2154...	6d7f68b0...	4f0ee28b...	2023-09-01	2024-07-01	учится
12	dc25462f-a669-4c72-88eb-c38ff1d3c6e4	9cca6112...	8a472103...	2023-09-01	2024-07-01	учится

Стало:

	student_id [PK] uuid	client_id uuid	group_id uuid	start_date date	finish_date date	status_in_group student_status
1	00495bd0-99de-436b-8456-6fbbb70248...	3b25f691...	4f0ee28b...	2023-09-01	2024-07-01	учится
2	0456f9f0-b753-4ad1-9cae-de8d7fbc9490	5c6ee5d3...	4f0ee28b...	2023-09-01	2024-07-01	учится
3	0ae4485d-faf1-4475-bb04-ad480075a2...	18ab5bd9...	4f0ee28b...	2023-09-01	2024-07-01	отчислен
4	3b8aaf62-d4aa-48f2-9cfd-f1a9c90c0ad1	132259da...	4f0ee28b...	2023-09-01	2024-07-01	отчислен
5	405e00b3-37e4-49e0-a191-929de9d72a...	7e8f59c1-...	4f0ee28b...	2023-09-01	2024-07-01	учится
6	6a15d442-c3fe-41cc-95ec-76f7c6583f87	19bc6de0...	4f0ee28b...	2023-09-01	2024-07-01	отчислен
7	9101da2f-9b03-4f3c-a569-f5708fa8c6c5	24fd3f29-...	4f0ee28b...	2023-09-01	2024-07-01	отчислен
8	a4b34641-1125-4ba5-8c8f-c6bebe87b4...	4acca545...	4f0ee28b...	2023-09-01	2024-07-01	учится
9	befeb349-8417-4759-912f-aeb8397174...	8bb9d041...	8a472103...	2023-09-01	2024-07-01	учится
10	c1c6b124-4291-44b9-9246-4e3781a8b4...	20cd7ef1-...	8a472103...	2023-09-01	2024-07-01	учится
11	d5d7cd03-d086-4a4e-9c2b-b1c14e2154...	6d7f68b0...	4f0ee28b...	2023-09-01	2024-07-01	учится
12	dc25462f-a669-4c72-88eb-c38ff1d3c6e4	9cca6112...	8a472103...	2023-09-01	2024-07-01	учится

Этим запросом удаляются все клиенты, которые не являются студентами.

```
DELETE FROM client
WHERE NOT EXISTS(SELECT * FROM student WHERE student.client_id = client.client_id);
```

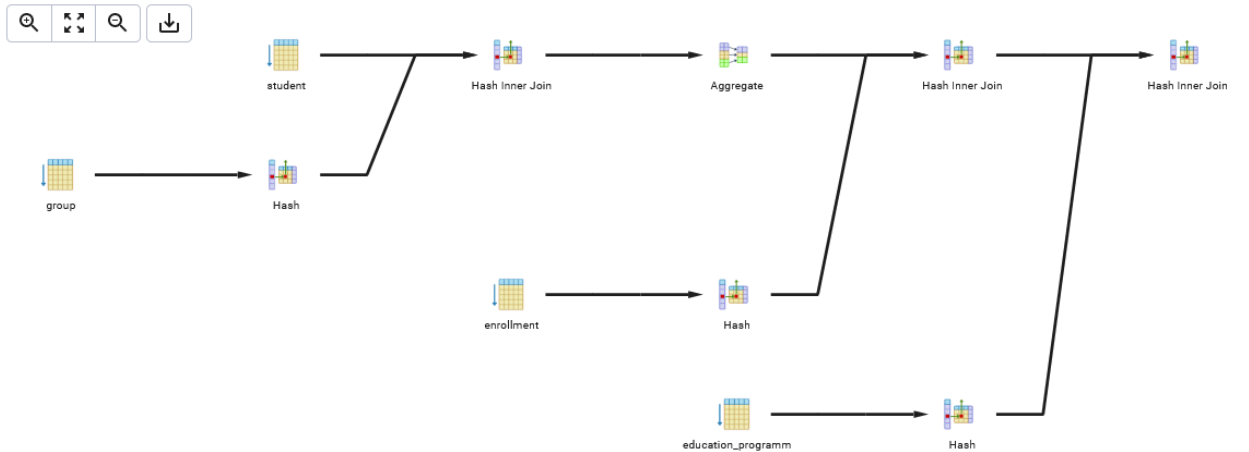
Было:

	client_id [PK] uuid	name character varying (64)	surname character varying (64)	patronymic character varying (64)	contacts character varying (256)
1	132259da-3ddb-4bae-8d01-39414d808...	Василий	Василевский	Васильевич	vasyapupkin@mail.ru +7894595495409
2	18ab5bd9-9b31-4d3b-9d24-11e25cc9fe...	Сергей	Сергеев	Сергеевич	sergeys@example.com +79044567890
3	19bc6de0-ac42-4fb5-a5b5-21f26dc0fetc	Ольга	Ольгина	Олеговна	olgao@example.com +79055678901
4	20cd7ef1-bd53-4fc6-b6b6-31f37ec1f0fd	Александр	Александров	Александрович	alexandera@example.com +79066789012
5	24fd3f29-ec14-4f51-bf76-2f9bf9181b67	Анна	Андреевна	Анатолевна	annushka@example.com +71234567890
6	3b25f691-ac4e-4d86-9ba3-f0b9fc2d61...	Игорь	Игнатов	Игоревич	igorek123@inbox.ru +79876543210
7	4acca545-ecb6-4aaf-8c34-bf3b1a3d38...	Елена	Еленова	Евгеньевна	elenka@somemail.com +73451234567
8	5c6ee5d3-ee1a-4e9b-9e69-1a5e4c9b9e...	Мария	Марьева	Михайловна	mariam@example.com +79011234567
9	6d7f68b0-5d3a-49f6-8eba-1a9e4c1c9f2e	Дмитрий	Дмитриев	Дмитриевич	dmitryd@example.com +79022345678
10	7e8f59c1-6f4b-48b9-9cba-2a0e5c2d9f3f	Ирина	Иванова	Игоревна	irinai@example.com +79033456789
11	8bb9d041-2d55-40d4-afee-1e8e5d2d9e...	Надежда	Николаева	Николаевна	nadyan@example.com +79077890123
12	9cca6112-3d66-4e8d-bf7f-2e9f6e3c9f90	Константин	Константинов	Константинович	konstantink@example.com +79088901234
13	8bb9d041-2d55-40d4-afee-1e8e5d2d9e...	Василий	Пупкин	Петрович	konstantink@example.com +79088901234

Стало:

	client_id [PK] uuid	name character varying (64)	surname character varying (64)	patronymic character varying (64)	contacts character varying (256)
1	132259da-3ddb-4bae-8d01-39414d808...	Василий	Василевский	Васильевич	vasyapupkin@mail.ru +7894595495409
2	18ab5bd9-9b31-4d3b-9d24-11e25cc9fe...	Сергей	Сергеев	Сергеевич	sergeys@example.com +79044567890
3	19bc6de0-ac42-4fb5-a5b5-21f26dc0fetc	Ольга	Ольгина	Олеговна	olgao@example.com +79055678901
4	20cd7ef1-bd53-4fc6-b6b6-31f37ec1f0fd	Александр	Александров	Александрович	alexandera@example.com +79066789012
5	24fd3f29-ec14-4f51-bf76-2f9bf9181b67	Анна	Андреевна	Анатолевна	annushka@example.com +71234567890
6	3b25f691-ac4e-4d86-9ba3-f0b9fc2d61...	Игорь	Игнатов	Игоревич	igorek123@inbox.ru +79876543210
7	4acca545-ecb6-4aaf-8c34-bf3b1a3d38...	Елена	Еленова	Евгеньевна	elenka@somemail.com +73451234567
8	5c6ee5d3-ee1a-4e9b-9e69-1a5e4c9b9e...	Мария	Марьева	Михайловна	mariam@example.com +79011234567
9	6d7f68b0-5d3a-49f6-8eba-1a9e4c1c9f2e	Дмитрий	Дмитриев	Дмитриевич	dmitryd@example.com +79022345678
10	7e8f59c1-6f4b-48b9-9cba-2a0e5c2d9f3f	Ирина	Иванова	Игоревна	irinai@example.com +79033456789
11	8bb9d041-2d55-40d4-afee-1e8e5d2d9e...	Надежда	Николаева	Николаевна	nadyan@example.com +79077890123
12	9cca6112-3d66-4e8d-bf7f-2e9f6e3c9f90	Константин	Константинов	Константинович	konstantink@example.com +79088901234

Explain самого первого запроса, так как он выглядит самым большим.



#	Node	Actual	Loops
1.	→ Nested Loop Inner Join (rows=2 loops=1)	2	1
2.	→ Hash Inner Join (rows=2 loops=1) Hash Cond: (enrollment.enrollment_id = unnamed_subquery.enrollment_id_short)	2	1
3.	→ Seq Scan on enrollment as enrollment (rows=2 loops=1)	2	1
4.	→ Hash (rows=2 loops=1) Buckets: 1024 Batches: 1 Memory Usage: 9 kB	2	1
5.	→ Subquery Scan (rows=2 loops=1)	2	1
6.	→ Aggregate (rows=2 loops=1) Filter: (count(student.student_id) < 10) Rows Removed by Filter: 0 Buckets: Batches: Memory Usage: 24 kB	2	1
7.	→ Hash Inner Join (rows=12 loops=1) Hash Cond: ("group".group_id = student.group_id)	12	1
8.	→ Seq Scan on group as group (rows=2 loops=1)	2	1
9.	→ Hash (rows=12 loops=1) Buckets: 1024 Batches: 1 Memory Usage: 9 kB	12	1
10.	→ Seq Scan on student as student (rows=12 loops=1)	12	1
11.	Index Only Scan using education_programm_pkey on education_programm as education_pr... Index Cond: (programm_code = (enrollment.programm_code)::text)	1	2

История запросов:

Query
Query History

Show queries generated internally by pgAdmin?
☒

Remove Remove All

Today - 18.02.2024

```

E SELECT number, education_programm.progra
09:49:35
▶ SELECT number, education_programm.progra
09:49:29
▶ DELETE FROM client WHERE NOT EXISTS(SE
09:48:02
▶ UPDATE student SET status_in_group = '...
09:47:08
▶ UPDATE student SET status_in_group = '...
09:47:01
▶ INSERT INTO result (result_id, student...
09:45:18
▶ INSERT INTO result (result_id, student...
09:45:02
COMMIT;
09:44:49

```

18.02.2024 09:49:35
Date

1
Rows affected

110 msec
Duration

Copy Copy to Query Editor

```

SELECT number, education_programm.progra
FROM (SELECT "group".group_id, number, "
FROM "group"
JOIN student ON "group".g
GROUP BY "group".group_id
HAVING COUNT(student.student_id) <
)
JOIN enrollment ON enrollment_id_short =
JOIN education_programm ON enrollment.pr

```

Messages

Successfully run. Total query runtime: 110 msec. 1 rows affected.

Без индексов время выполнения около 100мс, но погрешность очень большая.

✓ Successfully run. Total query runtime: 73 msec. 2 rows affected. ✕

С индексом

```
CREATE INDEX group_id_index ON student (group_id);
```

✓ Successfully run. Total query runtime: 84 msec. 2 rows affected. ✕

✓ Successfully run. Total query runtime: 75 msec. 2 rows affected. ✕

✓ Successfully run. Total query runtime: 88 msec. 2 rows affected. ✕

✓ Successfully run. Total query runtime: 88 msec. 2 rows affected. ✕

Кажется, что время выполнения в среднем уменьшилось, но я по настоящему не считал среднее на большом количестве попыток и, скорее всего, ничего не изменилось, потому что данных очень мало. План запроса остался таким же, как и без индекса.

Дальше я создал составной индекс для запроса на вычисление загрузки компьютерных классов. Я сделал составной индекс для class по атрибутам classroom_id и date.

```
CREATE INDEX class_some_fields_index ON class (classroom_id, date);
```

До добавления индексов время выполнения было около 90мс, после добавления осталось таким же, как и в предыдущем случае, из-за того что данных слишком мало. Explain без индекса и с индексом одинаковые:

Вывод:

Овладел практическими навыками создания представлений и запросов на выборку данных к базе данных PostgreSQL, использования подзапросов при модификации данных и индексов. Также научился применять JOIN запросы, разобрался в отличиях между ними.