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 II курса ИКТ
группы К3240
Кобелев Л.К.

Санкт-Петербург
2022

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

Задачи:

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 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.

Индивидуальное задание. Вариант 15.

Описание предметной области: БД образовательной организации содержит сведения об аудиториях и расписании проводимых в них занятий. Занятия проводятся на разных площадках. Время начала и окончания занятия по дням недели фиксировано. База данных используется для получения справок о наличии свободных аудиторий в указанное время, о месте и времени проведения определенных занятий.

БД должна содержать следующий минимальный набор сведений:

- Номер аудитории.
- Количество мест.
- Тип аудитории.
- Название площадки.
- Адрес площадки.
- Код дисциплины.
- Название дисциплины.
- Вид занятия.
- ФИО преподавателя.
- Должность преподавателя.
- Номер студенческой группы.
- Учебный год.
- Код направления.
- Название направления.
- Код подразделения.
- Название подразделения.
- Максимально возможное количество студентов для посещения занятия.
- Дата.
- День недели.
- Время начала занятия.
- Время окончания занятия.

Задание 2. Создать запросы:

- Вывести загрузку преподавателей в понедельник (в часах).
- Найти недельную нагрузку студентов каждой группы (в часах).
- Вывести список свободных лекционных аудиторий в заданное время.
- Вывести количество аудиторий каждого типа.
- Вывести еженедельное количество часов занятий для каждой группы.
- Найти номера аудиторий каждого типа, имеющих максимальное количество мест.
- Вывести фамилии преподавателей, которые всегда проводят практические занятия в одной и той же аудитории.

Задание 3. Создать представление:

- содержащее данные о расписании заданной группы на каждый день;
- средняя недельная аудиторная нагрузка по группам по каждому направлению.

Схема базы данных представлена на рисунке 1.

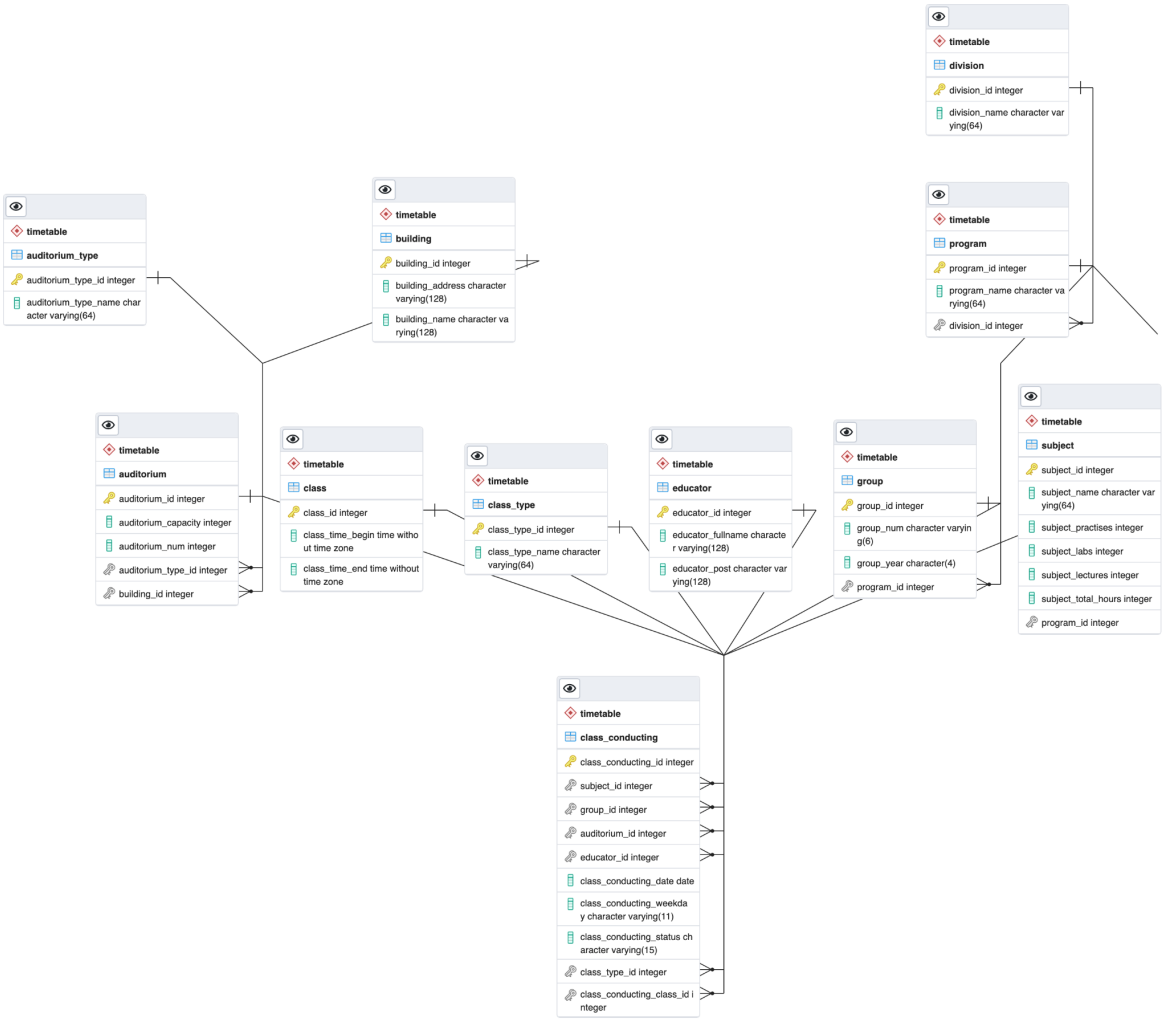


Рисунок 1 - схема базы данных timetable

Выполнение

Запросы к базе данных

1. **Вывести загрузку преподавателей в понедельник (в часах).**

```sql
SELECT e.educator_fullname, time '01:30' * COUNT(DISTINCT cc.class_conducting_class_id) as load
FROM timetable.class_conducting cc
JOIN timetable.educator e
ON cc.educator_id = e.educator_id
WHERE cc.class_conducting_weekday LIKE 'Понедельник'
GROUP BY e.educator_fullname
```

Data Output | Explain | Messages | Notifications

	educator_fullname character varying (128)	load interval
1	Гаусс Карл Фридрих	01:30:00
2	Демидович Борис Павлович	01:30:00
3	Лагранж Жозеф-Луи	03:00:00
4	Эйлер Леонард	01:30:00

2. **Найти недельную нагрузку студентов каждой группы (в часах).**

```
SELECT g.group_num, g.group_year, time '01:30' * COUNT(cc.class_conducting_id) as load
FROM timetable.class_conducting cc
JOIN timetable.group g
ON g.group_id = cc.group_id
GROUP BY g.group_num, g.group_year
```

Data Output | Explain | Messages | Notifications

	group_num character varying (6)	group_year character (4)	load interval
1	M1110	2022	22:30:00
2	M1130	2022	22:30:00
3	M1120	2022	22:30:00

3. **Вывести количество аудиторий каждого типа.**

```
SELECT at.auditorium_type_name, COUNT(a.auditorium_id)
FROM timetable.auditorium a
JOIN timetable.auditorium_type at
ON a.auditorium_type_id = at.auditorium_type_id
GROUP BY at.auditorium_type_name
```

Data Output | Explain | Messages | Notifications

	auditorium_type_name character varying (64)	count bigint
1	Семинарная	4
2	Лекционная	2

4. **Вывести список свободных лекционных аудиторий в заданное время.**

```
SELECT DISTINCT a.auditorium_num
FROM timetable.auditorium_type at
JOIN timetable.auditorium a
ON at.auditorium_type_id = a.auditorium_type_id
JOIN timetable.class_conducting cc
ON cc.auditorium_id = a.auditorium_id
JOIN timetable.class c
ON c.class_id = cc.class_conducting_class_id
WHERE at.auditorium_type_name LIKE 'Лекционная'
AND make_time(10, 0, 0) NOT BETWEEN c.class_time_begin AND c.class_time_end
AND make_date(2022, 4, 19) != cc.class_conducting_date
```

Data Output | Explain | Messages | Notifications

	auditorium_num integer
1	201
2	100

5. **Вывести еженедельное количество часов занятий для каждой группы.**

```
SELECT g.group_num, g.group_year, time '01:30' * COUNT(cc.class_conducting_id) as load
FROM timetable.class_conducting cc
JOIN timetable.group g
ON g.group_id = cc.group_id
GROUP BY g.group_num, g.group_year
```

Data Output | Explain | Messages | Notifications

	group_num character varying (6)	group_year character (4)	load interval
1	M1110	2022	22:30:00
2	M1130	2022	22:30:00
3	M1120	2022	22:30:00

6. **Найти номера аудиторий каждого типа, имеющих максимальное количество мест.**

```
SELECT at.auditorium_type_name, a.auditorium_num, a.auditorium_capacity
FROM timetable.auditorium a
JOIN timetable.auditorium_type at
ON a.auditorium_type_id = at.auditorium_type_id
WHERE (at.auditorium_type_name, a.auditorium_capacity) IN
    (
    SELECT at.auditorium_type_name, MAX(a.auditorium_capacity)
    FROM timetable.auditorium a
    JOIN timetable.auditorium_type at
    ON a.auditorium_type_id = at.auditorium_type_id
    GROUP BY at.auditorium_type_name
    )
```

Data Output | Explain | Messages | Notifications

	auditorium_type_name character varying (64)	auditorium_num integer	auditorium_capacity integer
1	Лекционная	100	110
2	Семинарная	202	40

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

```
SELECT educator_fullname
FROM (
    SELECT e.educator_fullname, COUNT(DISTINCT cc.auditorium_id)
    FROM timetable.educator e
    JOIN timetable.class_conducting cc
    ON e.educator_id = cc.educator_id
    WHERE cc.class_type_id = 3
    GROUP BY e.educator_fullname
) as count
WHERE count = 1
```

Data Output | Explain | Messages | Notifications

	educator_fullname character varying (128)
1	Гаусс Карл Фридрих
2	Демидович Борис Павлович
3	Золотарёв Егор Иванович
4	Колмогоров Андрей Николаевич
5	Лагранж Жозеф-Луи
6	Нэш Джон Форбс
7	Райгородский Андрей Михайлович
8	Сриниваса Рамануджа
9	Тьюринг Алан
10	Чебышев Пафнутий Львович
11	Эйлер Леонард

Представления

1. Содержащее данные о расписании заданной группы на каждый день

```sql
CREATE VIEW M1110_Schedule AS
    SELECT cc.class_conducting_date, cc.class_conducting_weekday, c.class_time_begin, c.class_time_end, s.subject_name
    FROM timetable.class_conducting cc
    JOIN timetable.group g
    ON cc.group_id = g.group_id
    JOIN timetable.subject s
    ON cc.subject_id = s.subject_id
    JOIN timetable.class c
    ON cc.class_conducting_class_id = c.class_id
    WHERE g.group_num LIKE 'M1110' AND g.group_year LIKE '2022'
```

Data Output | Explain | Messages | Notifications

```
CREATE VIEW

Query returned successfully in 35 msec.
```

```sql
SELECT *
FROM M1110_Schedule
```

Data Output | Explain | Messages | Notifications

	class_conducting_date date	class_conducting_weekday character varying (11)	class_time_begin time without time zone	class_time_end time without time zone	subject_name character varying (64)
1	2022-04-18	Понедельник	08:20:00	09:50:00	Мега Высшая Математика
2	2022-04-18	Понедельник	10:00:00	11:30:00	Мега Высшая Математика
3	2022-04-18	Понедельник	11:40:00	13:10:00	Мега Высшая Математика
4	2022-04-19	Вторник	10:00:00	11:30:00	Теория Круп
5	2022-04-19	Вторник	11:40:00	13:10:00	Теория Круп
6	2022-04-20	Среда	08:20:00	09:50:00	Теория Круп
7	2022-04-20	Среда	10:00:00	11:30:00	Мега Высшая Математика
8	2022-04-20	Среда	11:40:00	13:10:00	Теория Круп
9	2022-04-21	Четверг	11:40:00	13:10:00	Теория Цифр
10	2022-04-21	Четверг	13:30:00	15:00:00	Теория Цифр
11	2022-04-22	Пятница	10:00:00	11:30:00	Теория Вафли и Математическая Сосиска
12	2022-04-22	Пятница	11:40:00	13:10:00	Теория Вафли и Математическая Сосиска
13	2022-04-23	Суббота	10:00:00	11:30:00	Декретная Математика
14	2022-04-23	Суббота	11:40:00	13:10:00	Декретная Математика
15	2022-04-23	Суббота	13:30:00	15:00:00	Декретная Математика

2. Средняя недельная аудиторная нагрузка по группам по каждому направлению.

```sql
CREATE VIEW average_program_load AS
    SELECT program_name, AVG(weekly_time)
    FROM
    (
        SELECT p.program_name, g.group_num, time '01:30' * COUNT(cc.class_conducting_id) as weekly_time
        FROM timetable.program p
        JOIN timetable.group g
        ON p.program_id = g.program_id
        JOIN timetable.class_conducting cc
        ON g.group_id = cc.group_id
        GROUP BY p.program_name, g.group_num
    ) AS subquery
    GROUP BY program_name
```

Data Output | Explain | Messages | Notifications

```
CREATE VIEW

Query returned successfully in 28 msec.
```

```sql
SELECT *
FROM average_program_load
```

Data Output | Explain | Messages | Notifications

	program_name character varying (64)	avg interval
1	Прикладная Математика	22:30:00
2	Теоритечская Математика	22:30:00
3	Гуманитарная Математика	22:30:00

Запросы на модификацию данных

INSERT (Продублирована последняя пара группы M1110 23 апреля 2022)

```sql
INSERT INTO timetable.class_conducting
(class_conducting_id, subject_id, group_id, auditorium_id, educator_id,
class_conducting_date, class_conducting_weekday, class_conducting_status,
class_type_id, class_conducting_class_id)
SELECT cc.class_conducting_id + 100, cc.subject_id, cc.group_id, cc.auditorium_id, cc.educator_id,
cc.class_conducting_date, cc.class_conducting_weekday, cc.class_conducting_status,
cc.class_type_id, cc.class_conducting_class_id + 1
FROM timetable.class_conducting cc
JOIN timetable.group g
ON cc.group_id = g.group_id
WHERE g.group_num LIKE 'M1110' AND cc.class_conducting_date = '2022-04-23'
ORDER BY cc.class_conducting_class_id DESC
LIMIT 1
```

Data Output | Explain | Messages | Notifications

```
INSERT 0 1

Query returned successfully in 30 msec.
```

UPDATE (Обновлён преподаватель на свободного в этот момент с должностью “Junior Практик”)

```sql
UPDATE timetable.class_conducting
SET educator_id =
(
    SELECT e.educator_id
    FROM timetable.educator e
    WHERE e.educator_post LIKE 'Junior Практик' AND e.educator_id NOT IN
    (
        SELECT educator_id
        FROM timetable.class_conducting
        WHERE class_conducting_date = '2022-04-21' AND class_conducting_class_id = 4
    )
    LIMIT 1
)
WHERE class_conducting_id = 26
```

Data Output | Explain | Messages | Notifications

```
UPDATE 1

Query returned successfully in 32 msec.
```

DELETE (Удалены все пары, которые были более полугода назад от последней пары)

```
1  DELETE FROM timetable.class_conducting
2  WHERE class_conducting_date <
3  (
4      SELECT MAX(class_conducting_date) - interval '6 month'
5      FROM timetable.class_conducting
6  )
```

Data Output | Explain | Messages | Notifications

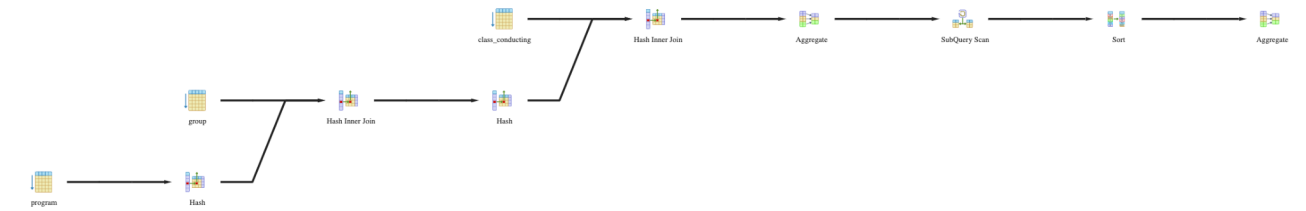
```
DELETE 0

Query returned successfully in 24 msec.
```

Создание индексов

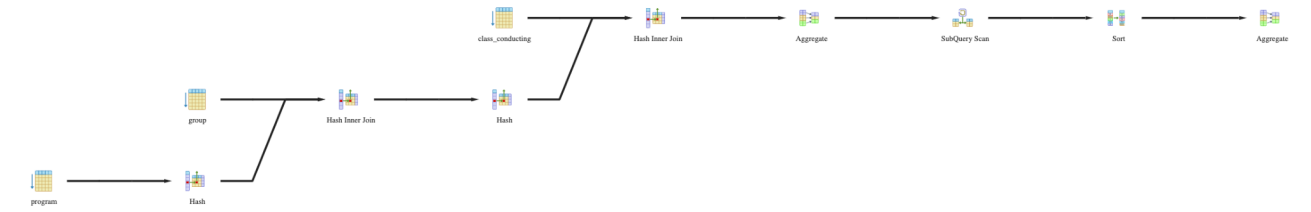
Простой индекс

Будем использовать VIEW average_program_load. Выполнение до создания индекса:



	#	Node	Timings		Rows	Loops
			Exclusive	Inclusive	Actual	
	1.	→ Aggregate (actual=1.235..1.244 rows=3 loops=1)	0.075 ms	1.244 ms	3	1
	2.	→ Sort (actual=1.164..1.17 rows=3 loops=1)	0.037 ms	1.17 ms	3	1
	3.	→ Subquery Scan (actual=1.121..1.133 rows=3 loops=1)	0.006 ms	1.133 ms	3	1
	4.	→ Aggregate (actual=1.117..1.128 rows=3 loops=1) Buckets: Batches: Memory Usage: 24 kB	1.009 ms	1.128 ms	3	1
	5.	→ Hash Inner Join (actual=0.098..0.119 rows=46 loops=1) Hash Cond: (cc.group_id = g.group_id)	0.03 ms	0.119 ms	46	1
	6.	→ Seq Scan on class_conducting as cc (actual=0.026..0.0...	0.03 ms	0.03 ms	46	1
	7.	→ Hash (actual=0.056..0.059 rows=3 loops=1) Buckets: 1024 Batches: 1 Memory Usage: 9 kB	0.005 ms	0.059 ms	3	1
	8.	→ Hash Inner Join (actual=0.05..0.054 rows=3 loops=... Hash Cond: (g.program_id = p.program_id)	0.015 ms	0.054 ms	3	1
	9.	→ Seq Scan on group as g (actual=0.01..0.01 row...	0.01 ms	0.01 ms	3	1
	10.	→ Hash (actual=0.028..0.029 rows=3 loops=1) Buckets: 1024 Batches: 1 Memory Usage: 9 kB	0.012 ms	0.029 ms	3	1
	11.	→ Seq Scan on program as p (actual=0.017....	0.018 ms	0.018 ms	3	1

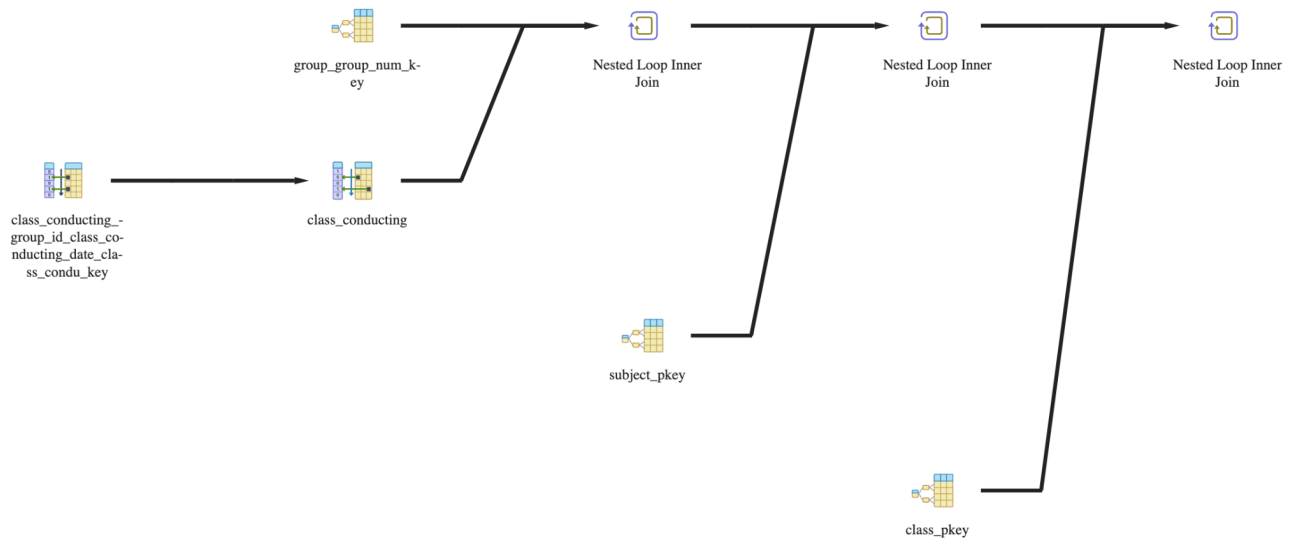
Создадим индекс: CREATE INDEX idx_program_name ON timetable.program(program_name). Выполним запрос повторно:



	#	Node	Timings		Rows	
			Exclusive	Inclusive	Actual	Loops
	1.	→ Aggregate (actual=0.107..0.111 rows=3 loops=1)	0.009 ms	0.111 ms	3	1
	2.	→ Sort (actual=0.101..0.103 rows=3 loops=1)	0.011 ms	0.103 ms	3	1
	3.	→ Subquery Scan (actual=0.088..0.092 rows=3 loops=1)	0.003 ms	0.092 ms	3	1
	4.	→ Aggregate (actual=0.088..0.09 rows=3 loops=1) Buckets: Batches: Memory Usage: 24 kB	0.023 ms	0.09 ms	3	1
	5.	→ Hash Inner Join (actual=0.049..0.067 rows=46 loops=1) Hash Cond: (cc.group_id = g.group_id)	0.02 ms	0.067 ms	46	1
	6.	→ Seq Scan on class_conducting as cc (actual=0.006..0.0...	0.01 ms	0.01 ms	46	1
	7.	→ Hash (actual=0.036..0.038 rows=3 loops=1) Buckets: 1024 Batches: 1 Memory Usage: 9 kB	0.005 ms	0.038 ms	3	1
	8.	→ Hash Inner Join (actual=0.03..0.033 rows=3 loops=... Hash Cond: (g.program_id = p.program_id)	0.01 ms	0.033 ms	3	1
	9.	→ Seq Scan on group as g (actual=0.002..0.002 r...	0.002 ms	0.002 ms	3	1
	10.	→ Hash (actual=0.02..0.021 rows=3 loops=1) Buckets: 1024 Batches: 1 Memory Usage: 9 kB	0.017 ms	0.021 ms	3	1
	11.	→ Seq Scan on program as p (actual=0.003....	0.004 ms	0.004 ms	3	1

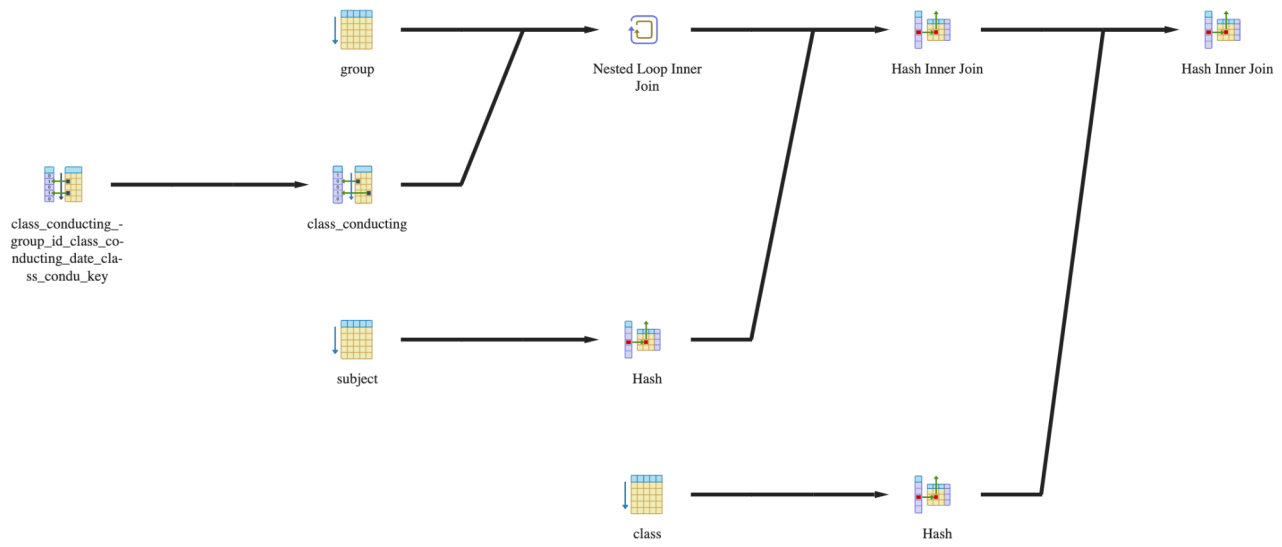
Составной индекс

Будем использовать VIEW M1110_Schedule . Выполнение до создания индекса:



#	Node	Timings		Rows	
		Exclusive	Inclusive	Actual	Loops
1.	→ Nested Loop Inner Join (actual=0.045..0.088 rows=16 loops=1)	0.024 ms	0.088 ms	16	1
2.	→ Nested Loop Inner Join (actual=0.038..0.063 rows=16 loops=1)	0.028 ms	0.063 ms	16	1
3.	→ Nested Loop Inner Join (actual=0.028..0.034 rows=16 loops=1)	0.008 ms	0.034 ms	16	1
4.	→ Index Scan using group_group_num_key on group as g (actual… **Filter**: ((((group_num)::text ~~ 'M1110'::text) AND (group_year ~~ '2022'::text)) **Index Cond**: ((group_num)::text = 'M1110'::text) **Rows Removed by Filter**: 0	0.015 ms	0.015 ms	1	1
5.	→ Bitmap Heap Scan on class_conducting as cc (actual=0.011..0… **Recheck Cond**: (group_id = g.group_id) **Heap Blocks**: exact=1	0.007 ms	0.012 ms	16	1
6.	→ Bitmap Index Scan using class_conducting_group_id_clas… **Index Cond**: (group_id = g.group_id)	0.005 ms	0.005 ms	16	1
7.	→ Index Scan using subject_pkey on subject as s (actual=0.001..0.001… **Index Cond**: (subject_id = cc.subject_id)	0.001 ms	0.001 ms	1	16
8.	→ Index Scan using class_pkey on class as c (actual=0.001..0.001 rows=1 l… **Index Cond**: (class_id = cc.class_conducting_class_id)	0.001 ms	0.001 ms	1	16

Создадим индекс: CREATE INDEX idx_group_num_year ON timetable.group(group_num, group_year). Выполним запрос повторно:



#	Node	Timings		Rows	
		Exclusive	Inclusive	Actual	Loops
1.	→ Hash Inner Join (actual=0.112..0.129 rows=16 loops=1) Hash Cond: (cc.class_conducting_class_id = c.class_id)	0.016 ms	0.129 ms	16	1
2.	→ Hash Inner Join (actual=0.07..0.082 rows=16 loops=1) Hash Cond: (cc.subject_id = s.subject_id)	0.016 ms	0.082 ms	16	1
3.	→ Nested Loop Inner Join (actual=0.025..0.031 rows=16 loops=1)	0.007 ms	0.031 ms	16	1
4.	→ Seq Scan on group as g (actual=0.006..0.007 rows=1 loops=1) Filter: (((group_num)::text ~~ 'M1110'::text) AND (group_year ~~ '2022'::text)) Rows Removed by Filter: 2	0.007 ms	0.007 ms	1	1
5.	→ Bitmap Heap Scan on class_conducting as cc (actual=0.015..... Recheck Cond: (group_id = g.group_id) Heap Blocks: exact=1	0.011 ms	0.017 ms	16	1
6.	→ Bitmap Index Scan using class_conducting_group_id_cla... Index Cond: (group_id = g.group_id)	0.007 ms	0.007 ms	16	1
7.	→ Hash (actual=0.036..0.036 rows=7 loops=1) Buckets: 1024 Batches: 1 Memory Usage: 9 kB	0.026 ms	0.036 ms	7	1
8.	→ Seq Scan on subject as s (actual=0.009..0.01 rows=7 loops=1)	0.01 ms	0.01 ms	7	1
9.	→ Hash (actual=0.031..0.031 rows=5 loops=1) Buckets: 1024 Batches: 1 Memory Usage: 9 kB	0.019 ms	0.031 ms	5	1
10.	→ Seq Scan on class as c (actual=0.011..0.013 rows=5 loops=1)	0.013 ms	0.013 ms	5	1

Удаление индексов

DROP INDEX timetable.idx_program_name, timetable.idx_group_num_year

Выводы

Созданы запросы и представления на выборку данных к базе данных PostgreSQL, составлены запросы на модификацию данных (INSERT, UPDATE, DELETE) с использованием подзапросов, изучено графическое представление запросов, созданы простой и составной индексы для двух произвольных запросов, проведено сравнение времени выполнения запросов без индексов и с индексами.