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»

Выполнила:
студентка II курса ИКТ
группы К3241
Кормановская Д.

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.

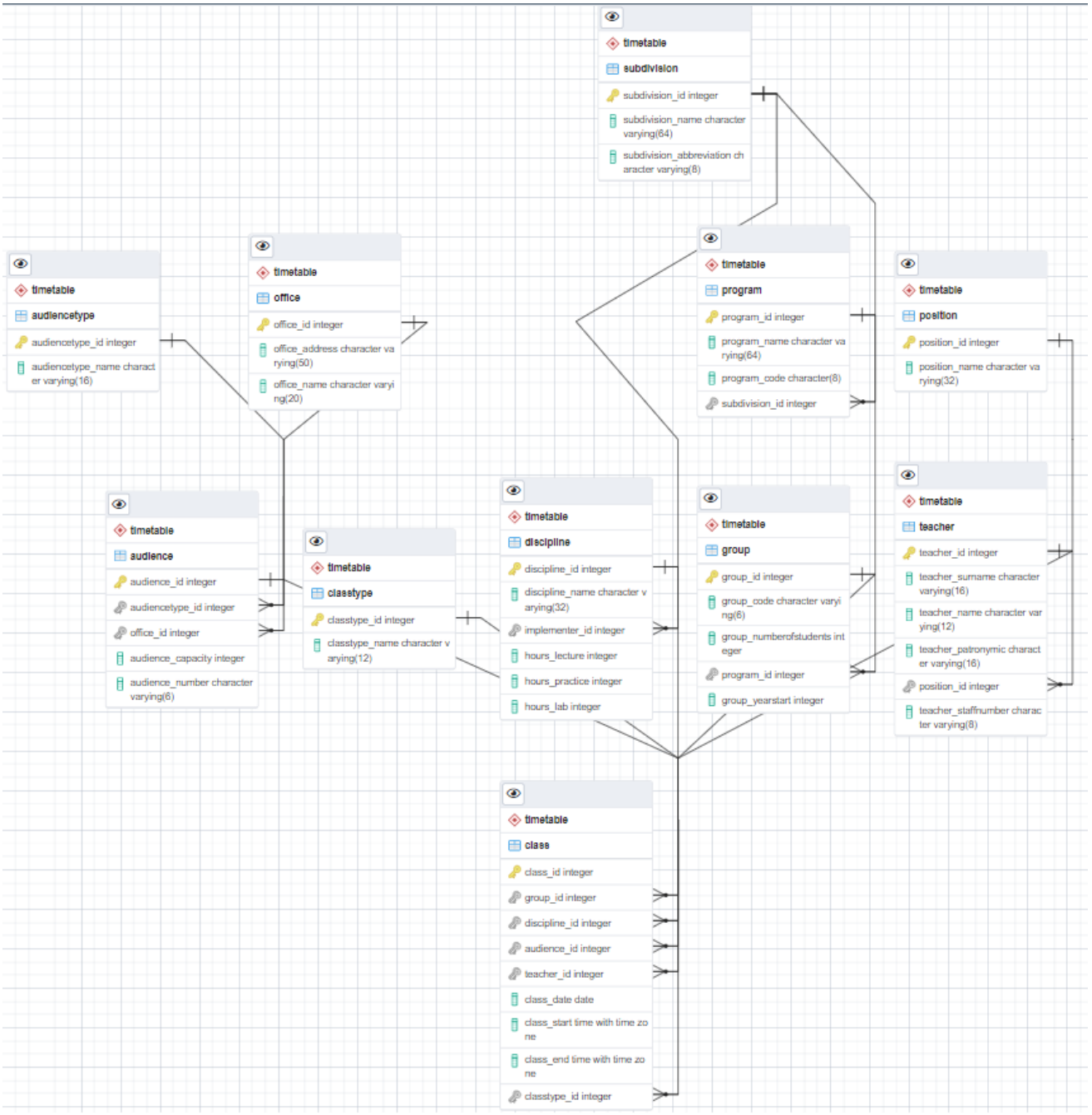


Рисунок 1 - схема базы данных timetable

Выполнение

Запросы к базе данных

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

```
SELECT teacher_name, SUM(duration)
FROM (
	SELECT concat(t.teacher_surname, ' ', t.teacher_name) as teacher_name,
	(c.class_end - c.class_start) as duration, date_part('isodow', c.class_date) as weekday
	FROM timetable."class" c
	JOIN timetable.teacher t ON t.teacher_id = c.teacher_id
	) as subquery
WHERE weekday = 1
GROUP BY teacher_name
```

```sql
SELECT teacher_name, SUM(duration)
FROM (
    SELECT concat(t.teacher_surname, ' ', t.teacher_name) as teacher_name,
    (c.class_end - c.class_start) as duration, date_part('isodow', c.class_date) as weekday
    FROM timetable."class" c
    JOIN timetable.teacher t ON t.teacher_id = c.teacher_id
    ) as subquery
WHERE weekday = 1
GROUP BY teacher_name
```

Data Output | Explain | Messages | Notifications

	teacher_name text	sum interval
1	Верди Джузеппе	01:30:00
2	Смирнова Анна	01:30:00

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

```
SELECT group_code, SUM(duration) as load
FROM (
        SELECT g.group_code, (c.class_end - c.class_start) as duration
        FROM timetable."class" c
        JOIN timetable."group" g
        ON c.group_id = g.group_id
        ) as subquery
GROUP BY group_code
```

```sql
SELECT group_code, SUM(duration) as load
FROM (
    SELECT g.group_code, (c.class_end - c.class_start) as duration
    FROM timetable."class" c
    JOIN timetable."group" g
    ON c.group_id = g.group_id
    ) as subquery
GROUP BY group_code
```

Data Output | Explain | Messages | Notifications

	group_code character varying (6)	load interval
1	sefs56	04:30:00
2	lklk22	10:10:00
3	lklk43	05:00:00

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

```
SELECT at.audiencetype_name, COUNT(*)
FROM timetable.audiencetype at
JOIN timetable.audience a
ON a.audiencetype_id = at.audiencetype_id
GROUP BY at.audiencetype_name
```

```
1  SELECT at.audiencetype_name, COUNT(*)
2  FROM timetable.audiencetype at
3  JOIN timetable.audience a
4  ON a.audiencetype_id = at.audiencetype_id
5  GROUP BY at.audiencetype_name
```

Data Output | Explain | Messages | Notifications

	audiencetype_name character varying (16)	count bigint
1	Актовый зал	1
2	Кабинет	4
3	Коворкинг	1

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

```
SELECT DISTINCT a.audience_id, a.audience_capacity, a.audience_number, a.office_id
FROM timetable.audience a
JOIN timetable."class" c
ON a.audience_id = c.audience_id
WHERE make_time(9,0,0) NOT BETWEEN c.class_start AND c.class_end
AND make_date(2022,3,21) != c.class_date
ORDER BY a.audience_id
```

```
1  SELECT DISTINCT a.audience_id, a.audience_capacity, a.audience_number, a.office_id
2  FROM timetable.audience a
3  JOIN timetable."class" c
4  ON a.audience_id = c.audience_id
5  WHERE make_time(9,0,0) NOT BETWEEN c.class_start AND c.class_end
6  AND make_date(2022,3,21) != c.class_date
7  ORDER BY a.audience_id
```

Data Output | Explain | Messages | Notifications

	audience_id [PK] integer	audience_capacity integer	audience_number character varying (6)	office_id integer
1	2	300	93n234	1
2	3	40	24234	1
3	5	35	223	2

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

```
SELECT group_code, SUM(duration) as load
FROM (
	SELECT g.group_code, (c.class_end - c.class_start) as duration
	FROM timetable."class" c
	JOIN timetable."group" g
	ON c.group_id = g.group_id
	) as subquery
GROUP BY group_code
```

```sql
SELECT group_code, SUM(duration) as load
FROM (
    SELECT g.group_code, (c.class_end - c.class_start) as duration
    FROM timetable."class" c
    JOIN timetable."group" g
    ON c.group_id = g.group_id
    ) as subquery
GROUP BY group_code
```

Data Output | Explain | Messages | Notifications

	group_code character varying (6)	load interval
1	sefs56	04:30:00
2	lklk22	10:10:00
3	lklk43	05:00:00

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

```sql
SELECT a.audience_number, a.audience_capacity, at.audiencetype_name
FROM timetable.audience a
JOIN timetable.audiencetype at
ON a.audiencetype_id = at.audiencetype_id
WHERE CONCAT(a.audiencetype_id, ' ', a.audience_capacity) IN (
    SELECT CONCAT(a.audiencetype_id, ' ', MAX(a.audience_capacity))
    FROM timetable.audience a
    GROUP BY a.audiencetype_id
    )
```

```sql
SELECT a.audience_number, a.audience_capacity, at.audiencetype_name
FROM timetable.audience a
JOIN timetable.audiencetype at
ON a.audiencetype_id = at.audiencetype_id
WHERE CONCAT(a.audiencetype_id, ' ', a.audience_capacity) IN (
    SELECT CONCAT(a.audiencetype_id, ' ', MAX(a.audience_capacity))
    FROM timetable.audience a
    GROUP BY a.audiencetype_id
    )
```

Data Output | Explain | Messages | Notifications

	audience_number character varying (6)	audience_capacity integer	audiencetype_name character varying (16)
1	93n234	300	Актовый зал
2	24349	42	Кабинет
3	32-54	30	Коворкинг

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

```sql
SELECT teacher_surname
FROM (
    SELECT t.teacher_surname, COUNT(DISTINCT c.audience_id) as different_audience
```

FROM timetable.teacher t
JOIN timetable."class" c
ON t.teacher_id = c.teacher_id
GROUP BY t.teacher_surname
) AS subquery
WHERE different_audience = 1

```sql
SELECT teacher_surname
FROM (
    SELECT t.teacher_surname, COUNT(DISTINCT c.audience_id) as different_audience
    FROM timetable.teacher t
    JOIN timetable."class" c
    ON t.teacher_id = c.teacher_id
    GROUP BY t.teacher_surname
    ) AS subquery
WHERE different_audience = 1
```

Data Output | Explain | Messages | Notifications

	teacher_surname character varying (16)
1	Верди
2	Иванов
3	Смирнова

Представления

1. **Содержащее данные о расписании заданной группы на каждый день**

CREATE VIEW lklk43_timetable AS
SELECT g.group_code, c.class_date, c.class_start, c.class_end, d.discipline_name
FROM timetable."class" c
JOIN timetable."group" g
ON c.group_id = g.group_id
JOIN timetable.discipline d
ON d.discipline_id = c.discipline_id
WHERE g.group_code LIKE 'lklk43'
ORDER BY c.class_date, c.class_start

```sql
CREATE VIEW lklk43_timetable AS
    SELECT g.group_code, c.class_date, c.class_start, c.class_end, d.discipline_name
    FROM timetable."class" c
    JOIN timetable."group" g
    ON c.group_id = g.group_id
    JOIN timetable.discipline d
    ON d.discipline_id = c.discipline_id
    WHERE g.group_code LIKE 'lklk43'
    ORDER BY c.class_date, c.class_start
```

Data Output | Explain | Messages | Notifications

```
CREATE VIEW

Query returned successfully in 171 msec.
```

```
SELECT *
FROM lklk43_timetable
```

Data Output | Explain | Messages | Notifications

	group_code character varying (6)	class_date date	class_start time without time zone	class_end time without time zone	discipline_name character varying (32)
1	lklk43	2022-03-21	13:30:00	15:00:00	Основы ООП
2	lklk43	2022-03-22	08:20:00	09:50:00	Философия
3	lklk43	2022-03-22	10:00:00	10:30:00	Машинное обучение
4	lklk43	2022-03-25	15:20:00	16:50:00	Основы ООП

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

```
CREATE VIEW avg_load_by_program AS
    SELECT program_name, AVG(load) as avg_load
    FROM (
    SELECT p.program_name, g.group_code, (SUM(c.class_end) - SUM(c.class_start)) as load
        FROM timetable."class" c
        JOIN timetable."group" g
        ON c.group_id = g.group_id
        JOIN timetable."program" p
        ON p.program_id = g.program_id
        GROUP BY p.program_name, g.group_code
        ) as subquery
    GROUP BY program_name
```

```
CREATE VIEW avg_load_by_program AS
    SELECT program_name, AVG(load) as avg_load
    FROM (
        SELECT p.program_name, g.group_code, (SUM(c.class_end) - SUM(c.class_start)) as load
        FROM timetable."class" c
        JOIN timetable."group" g
        ON c.group_id = g.group_id
        JOIN timetable."program" p
        ON p.program_id = g.program_id
        GROUP BY p.program_name, g.group_code
        ) as subquery
    GROUP BY program_name
```

Data Output | Explain | Messages | Notifications

```
CREATE VIEW

Query returned successfully in 98 msec.
```

```
SELECT * FROM avg_load_by_program
```

Data Output | Explain | Messages | Notifications

	program_name character varying (64)	avg_load interval
1	Нейротехнологии	04:30:00
2	Прикладная информатика	07:35:00

Запросы на модификацию данных

INSERT

INSERT INTO timetable."class"
(class_id, group_id, discipline_id, audience_id, teacher_id,
class_date, class_start, class_end, classtype_id)
SELECT c.class_id + 100, c.group_id, c.discipline_id, c.audience_id,
c.teacher_id, c.class_date + interval '1 week',
c.class_start, c.class_end, c.classtype_id
FROM timetable."class" c
JOIN timetable."group" g
ON g.group_id = c.group_id
WHERE group_code LIKE 'lklk43'
AND c.class_date = '2022-03-22'

```
1  INSERT INTO timetable."class"
2  (class_id, group_id, discipline_id, audience_id, teacher_id,
3  class_date, class_start, class_end, classtype_id)
4  SELECT c.class_id + 100, c.group_id, c.discipline_id, c.audience_id,
5  c.teacher_id, c.class_date + interval '1 week',
6  c.class_start, c.class_end, c.classtype_id
7  FROM timetable."class" c
8  JOIN timetable."group" g
9  ON g.group_id = c.group_id
10 WHERE group_code LIKE 'lklk43'
11 AND c.class_date = '2022-03-22'
```

Data Output | Explain | Messages | Notifications

```
INSERT 0 2

Query returned successfully in 44 msec.
```

UPDATE

```
UPDATE timetable."class"
SET audience_id = (
                    SELECT audience_id
                    FROM timetable.audience
                    ORDER BY audience_capacity ASC
                    LIMIT 1
)
WHERE group_id in (
                    SELECT group_id
                    FROM timetable."group"
                    WHERE group_numberofstudents < 30
)
```

```sql
UPDATE timetable."class"
SET audience_id = (
    SELECT audience_id
    FROM timetable.audience
    ORDER BY audience_capacity ASC
    LIMIT 1
)
WHERE group_id in (
    SELECT group_id
    FROM timetable."group"
    WHERE group_numberofstudents < 30
)
```

Messages

```
UPDATE 3

Query returned successfully in 36 msec.
```

DELETE

```sql
DELETE FROM timetable.teacher
WHERE teacher_id NOT IN (
                SELECT DISTINCT teacher_id
                FROM timetable."class"
                WHERE class_date > '2022-01-01'
)
```

```sql
DELETE FROM timetable.teacher
WHERE teacher_id NOT IN (
    SELECT DISTINCT teacher_id
    FROM timetable."class"
    WHERE class_date > '2022-01-01'
)
```

Messages Data Output

```
DELETE 0

Query returned successfully in 64 msec.
```

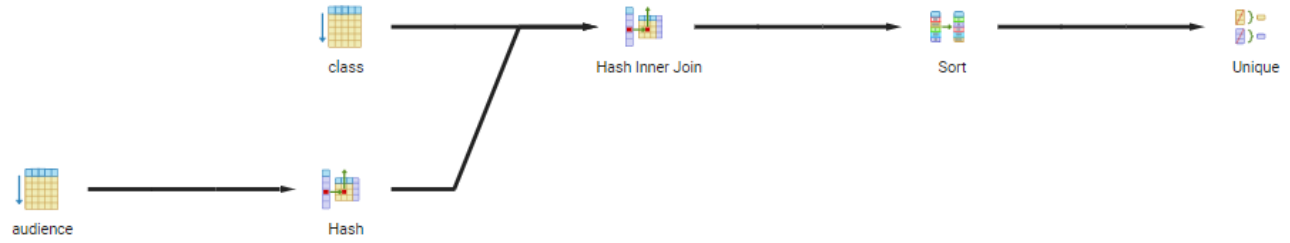
Создание индексов

Запрос:

```sql
SELECT DISTINCT a.audience_id, a.audience_capacity, a.audience_number, a.office_id
FROM timetable.audience a
JOIN timetable."class" c
ON a.audience_id = c.audience_id
WHERE make_time(9,0,0) NOT BETWEEN c.class_start AND c.class_end
AND make_date(2022,3,21) != c.class_date
```

ORDER BY a.audience_id

```
Successfully run. Total query runtime: 40 msec.
```



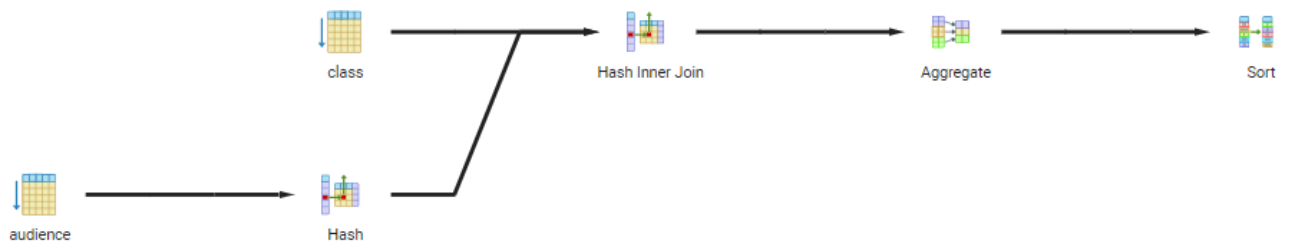
#	Node	Timings		Rows	
		Exclusive	Inclusive	Actual	Loops
1.	→ Unique (actual=0.064..0.068 rows=3 loops=1)	0.005 ms	0.068 ms	3	1
2.	→ Sort (actual=0.063..0.064 rows=4 loops=1)	0.014 ms	0.064 ms	4	1
3.	→ Hash Inner Join (actual=0.045..0.05 rows=4 loops=1) **Hash Cond**: (c.audience_id = a.audience_id)	0.015 ms	0.05 ms	4	1
4.	→ Seq Scan on class as c (actual=0.016..0.019 rows=... **Filter**: (('2022-03-21'::date <> class_date) AND (('09:00:0 0'::time without time zone < class_start) OR ('09:00:00'::ti me without time zone > class_end))) **Rows Removed by Filter**: 6	0.019 ms	0.019 ms	4	1
5.	→ Hash (actual=0.017..0.017 rows=7 loops=1) **Buckets**: 1024 **Batches**: 1 **Memory Usage**: 9 kB	0.01 ms	0.017 ms	7	1
6.	→ Seq Scan on audience as a (actual=0.006..0.00...	0.008 ms	0.008 ms	7	1

Простой индекс:

CREATE INDEX idx_classdate on timetable."class"(class_date);

Запрос после создания индекса:

#	Node	Timings		Rows	
		Exclusive	Inclusive	Actual	Loops
1.	→ Sort (actual=0.043..0.044 rows=3 loops=1)	0.006 ms	0.044 ms	3	1
2.	→ Aggregate (actual=0.037..0.038 rows=3 loops=1) **Buckets**: **Batches**: **Memory Usage**: 32 kB	0.009 ms	0.038 ms	3	1
3.	→ Hash Inner Join (actual=0.026..0.029 rows=4 loops=1) **Hash Cond**: (c.audience_id = a.audience_id)	0.008 ms	0.029 ms	4	1
4.	→ Seq Scan on class as c (actual=0.01..0.012 rows=... **Filter**: (('2022-03-21'::date <> class_date) AND (('09:00:0 0'::time without time zone < class_start) OR ('09:00:00'::t ime without time zone > class_end))) **Rows Removed by Filter**: 6	0.012 ms	0.012 ms	4	1
5.	→ Hash (actual=0.01..0.01 rows=7 loops=1) **Buckets**: 1024 **Batches**: 1 **Memory Usage**: 9 kB	0.004 ms	0.01 ms	7	1
6.	→ Seq Scan on audience as a (actual=0.005..0.0...	0.006 ms	0.006 ms	7	1

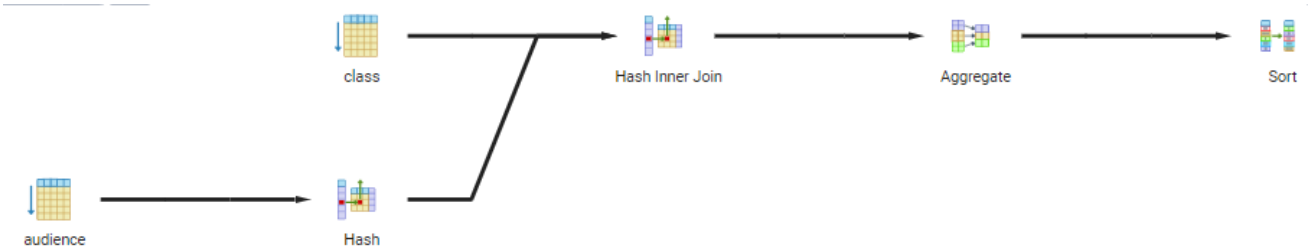


Составной индекс:

CREATE INDEX idx_classtime on timetable."class"(class_start, class_end);

Запрос после создания индекса:

	#	Node	Timings		Rows	
			Exclusive	Inclusive	Actual	Loops
	1.	→ Sort (actual=0.033..0.034 rows=3 loops=1)	0.006 ms	0.034 ms	3	1
	2.	→ Aggregate (actual=0.028..0.029 rows=3 loops=1) Buckets: Batches: Memory Usage: 32 kB	0.007 ms	0.029 ms	3	1
	3.	→ Hash Inner Join (actual=0.021..0.023 rows=4 loops=1) Hash Cond: (c.audience_id = a.audience_id)	0.005 ms	0.023 ms	4	1
	4.	→ Seq Scan on class as c (actual=0.008..0.01 rows=... Filter: (('2022-03-21'::date <> class_date) AND (('09:00:00'::time without time zone < class_start) OR ('09:00:00'::time without time zone > class_end))) Rows Removed by Filter: 6	0.01 ms	0.01 ms	4	1
	5.	→ Hash (actual=0.008..0.008 rows=7 loops=1) Buckets: 1024 Batches: 1 Memory Usage: 9 kB	0.003 ms	0.008 ms	7	1
	6.	→ Seq Scan on audience as a (actual=0.004..0.0...	0.005 ms	0.005 ms	7	1



Удаление индексов:
DROP INDEX timetable.idx_classtime, timetable.idx_classdate

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.