

**Министерство науки и высшего образования Российской Федерации**  
федеральное государственное автономное образовательное учреждение  
высшего образования  
**«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»**

## **ОТЧЕТ**

по лабораторной работе «Запросы на выборку и модификацию данных,  
представления и индексы в PostgreSQL»

по дисциплине «Проектирование и реализация баз данных»

Отчет выполнил:

Студент: Никоноров Максим

Группа: K32422

Факультет: ИКТ

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.

## Практическое задание

Ссылка на папку со скриншотами:

<https://drive.google.com/drive/u/0/folders/16rxlN6NILQHk16INGNR7L5rqynyT2AAy>

### Часть I

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

```
SELECT tutor_isu_no,  
       sum(end_time - begin_time)  
FROM structure.schedule  
WHERE day_of_week_no = '1' /* Дополнительно можно  
уточнить year и week_no */  
GROUP BY tutor_isu_no
```

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

```
SELECT group_no,  
       sum(end_time - begin_time)  
FROM structure.schedule  
WHERE week_no = '45'  
       AND YEAR = 2022 /* year и week_no можно изменить */  
GROUP BY group_no
```

Запрос 3. Вывести список свободных лекционных аудиторий в заданное время.

```
SELECT auditory_no
FROM structure.auditory
WHERE auditory.auditory_type = 'lecture'
      AND auditory.auditory_no NOT IN
      (SELECT auditory_no
       FROM structure.schedule
       WHERE YEAR = 2022
            AND week_no = 47
            AND day_of_week_no = '1')
```

Запрос 4. Вывести количество аудиторий каждого типа.

```
SELECT auditory_type,
       COUNT(auditory_type)
FROM structure.auditory
GROUP BY auditory_type
```

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

```
SELECT group_no,
       discipline_code,
       SUM(end_time - begin_time)
FROM structure.schedule
WHERE week_no = '46'
      AND YEAR = '2022'
GROUP BY group_no,
       discipline_code
```

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

```
SELECT auditory_type,  
       auditory_no,  
       value_of_seats  
FROM structure.auditory  
WHERE auditory_type = 'lecture'  
   AND value_of_seats =  
      (SELECT MAX(value_of_seats)  
       FROM structure.auditory  
       WHERE auditory_type = 'lecture'  
       GROUP BY auditory_type)  
OR auditory_type = 'laboratory'  
   AND value_of_seats =  
      (SELECT MAX(value_of_seats)  
       FROM structure.auditory  
       WHERE auditory_type = 'laboratory'  
       GROUP BY auditory_type)
```

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

```
SELECT name  
FROM structure.tutor  
WHERE tutor_isu_no IN  
      (SELECT subquery.tutor_isu_no  
       FROM  
        (SELECT tutor_isu_no,  
                 COUNT(DISTINCT auditory_no) AS nunique  
         FROM structure.schedule  
         WHERE lesson_type = 'practice'  
         GROUP BY tutor_isu_no) subquery  
       WHERE subquery.nunique = 1)
```

## Часть II

Запрос 1. Создать представление, содержащее данные о расписании заданной группы на каждый день.

```
CREATE VIEW GroupSchedule AS
SELECT discipline_code,
       begin_time,
       day_of_week_no,
       lesson_type
FROM structure.schedule
WHERE group_no = 'K32422'
   AND YEAR = 2022
   AND season = 'fall'
GROUP BY discipline_code,
       begin_time,
       day_of_week_no,
       lesson_type
ORDER BY day_of_week_no,
       begin_time
```

Запрос 2. Создать представление, содержащее среднюю недельную аудиторную нагрузку по группам по каждому направлению.

```
CREATE VIEW WeekLoad AS
SELECT auditory_no,
       CAST(COUNT(schedule.group_no) AS DEC(12, 4))
/ 16,
       direction.direction_code
FROM structure.direction,
       structure.edu_program,
       structure.edu_plan,
       structure.group,
       structure.schedule
WHERE schedule.group_no = structure.group.group_no
      AND structure.group.edu_plan_no =
edu_plan.edu_plan_no
      AND edu_plan.edu_program_no =
edu_program.edu_program_no
      AND edu_program.direction_code =
direction.direction_code
      AND YEAR = 2022
      AND season = 'fall'
GROUP BY direction.direction_code,
          auditory_no
```

### Часть III

Запрос 1. Вставка нового поля в расписание по имени преподавателя и предмета.

```
INSERT INTO structure.Schedule (group_no,  
discipline_code, auditory_no, YEAR, season,  
tutor_isu_no, begin_time, end_time, fixed_day,  
lesson_type)  
VALUES ('K32422', (SELECT discipline_code FROM  
structure.discipline WHERE name = 'Физическая  
культура'), 12306, 2022, 'fall', (SELECT  
tutor_isu_no FROM structure.tutor WHERE name =  
'Тимошенко Жанна Викторовна'), '10:00:00',  
'11:30:00', '2022-11-24', 'practice')
```

Запрос 2. Перевести всех менторов в онлайн на время 41–43 недель 2022 года.

```
UPDATE structure.schedule  
SET auditory_no = NULL  
WHERE tutor_isu_no IN  
  (SELECT tutor_isu_no  
   FROM structure.tutor  
   WHERE POSITION = 'mentor')  
AND YEAR = 2022  
AND season = 'fall'  
AND week_no IN (41,  
                42,  
                43);
```

Запрос 3. Удаление студентов из группы, которые были отчислены.

**DELETE**

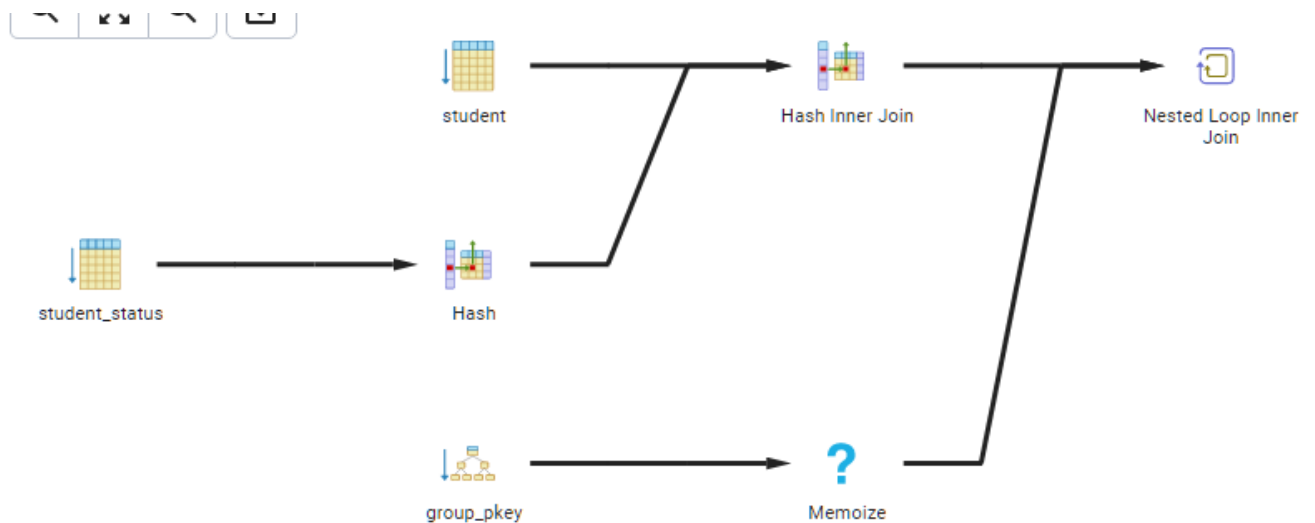
```
FROM structure.student_status USING structure.student
WHERE student_status.student_isu_no =
student.student_isu_no
AND student_status.group_no = 'R3143'
AND student_status.status = 'debt';
```

## Часть IV

Запрос 1. Использование одиночного индекса

Для начала я создал join-запрос, который выводит информацию о каждом студенте, его группе и статусе:

```
SELECT student.name,
        structure.group.group_no,
        student_status.status
FROM structure.student
INNER JOIN structure.student_status ON
student.student_isu_no =
student_status.student_isu_no
INNER JOIN structure.group ON student_status.group_no
= structure.group.group_no;
```





Сначала запускаю запрос без использования индексов:

```
1 SELECT student.name, structure.group.group_no, student_status.status
2 FROM structure.student
3 INNER JOIN structure.student_status ON student.student_isu_no = student_status.student_isu_no
4 INNER JOIN structure.group ON student_status.group_no = structure.group.group_no;
```

Total rows: 103 of 103    Query complete 00:00:00.144

Затем создаю индекс для ускорения работы join-a:

```
CREATE INDEX ind_all_students ON
structure.student_status (student_isu_no, group_no);
```

```
CREATE INDEX ind_all_students ON structure.student_status (student_isu_no, group_no);
```

Messages

CREATE INDEX

Query returned successfully in 171 msec.

И повторно запускаю программу:

```
3 SELECT student.name, structure.group.group_no, student_status.status
4 FROM structure.student
5 INNER JOIN structure.student_status ON student.student_isu_no = student_status.student_isu_no
6 INNER JOIN structure.group ON student_status.group_no = structure.group.group_no;
```

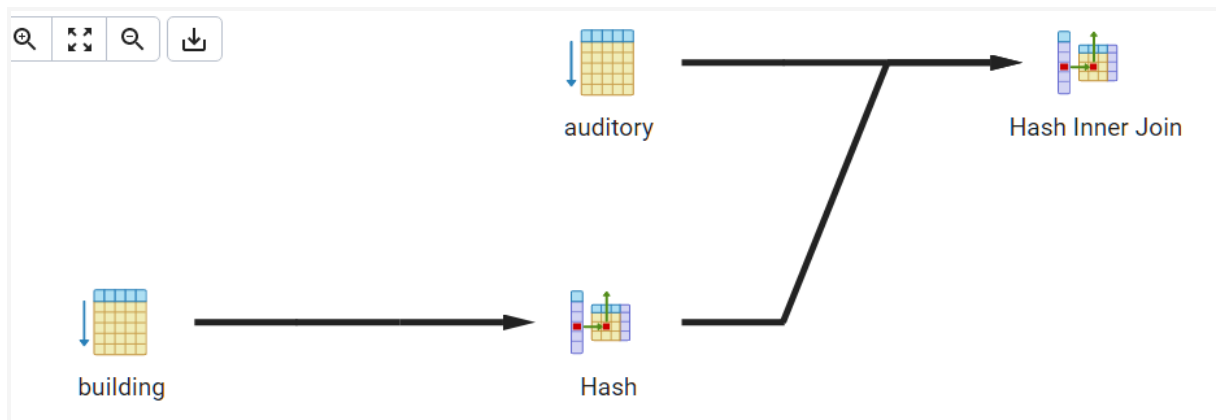
Total rows: 103 of 103    Query complete 00:00:00.084

Результат до–после: 144–84 мс.

## Запрос 2. Использование составного индекса

Создаю запрос, который выводит все московские аудитории, в которых проводятся лекции, с общим количеством мест больше 90

```
SELECT auditory_no,  
       value_of_seats  
FROM structure.auditory  
WHERE auditory_type = 'lecture'  
       AND building_no in  
       (SELECT building_no  
        FROM structure.building  
        WHERE city = 'Москва')  
       AND value_of_seats > 90;
```



Сначала запускаю запрос без использования индексов:

```
Messages Query History Query Explain × Notifications Data Output
1 SELECT auditory_no, value_of_seats
2 FROM structure.auditory
3 WHERE auditory_type = 'lecture'
4     AND building_no in (SELECT building_no FROM structure.building WHERE city = 'Москва')
5     AND value_of_seats > 90;

Total rows: 8 of 8 Query complete 00:00:00.157
```

Затем создаю два отдельных индекса для поиска города и кол-ва мест:

```
CREATE INDEX cit_h ON structure.building USING hash
(city);
```

```
CREATE INDEX seat_h ON structure.auditory USING hash
(value_of_seats)
```

```
1 CREATE INDEX cit_h ON structure.building USING hash (city);
2 CREATE INDEX seat_h ON structure.auditory USING hash (value_of_seats)
3 |
Messages
CREATE INDEX
Query returned successfully in 127 msec.
```

И повторно запускаю программу:

```
SELECT auditory_no, value_of_seats
FROM structure.auditory
WHERE auditory_type = 'lecture'|
  AND building_no in (SELECT building_no FROM structure.building WHERE city = 'Москва')
  AND value_of_seats > 90;
```

Output Messages Explain × Notifications

| auditory_no<br>[PK] integer | value_of_seats<br>integer |
|-----------------------------|---------------------------|
| 71119                       | 105                       |
| 71226                       | 120                       |
| 73209                       | 105                       |
| 74325                       | 120                       |
| 84302                       | 105                       |
| 84423                       | 105                       |
| 84427                       | 105                       |
| 83318                       | 135                       |

all rows: 8 of 8    Query complete 00:00:00.093

Результат до–после: 157–93 мс.

## **Вывод**

В ходе выполнения лабораторной работы я отточил навыки работы с базами данных PostgreSQL, закрепил структуры запросов выбора, представления, индексирования, вставки, обновления и удаления данных.

Начал лучше разбираться в среде работы с базами данных pgAdmin 4, получил знания о том, что вставка индексов помогает ускорить работу запросов.