Федеральное государственное автономное образовательное учреждение высшего образования «Национальный исследовательский университет ИТМО»

Факультет Программной Инженерии и Компьютерной Техники

Лабораторная работа №4

По Базам Данных

Вариант 34561

Выполнил:

Ларионов Владислав Васильевич

Группа P3109

Практик:

Воронина Дарья Сергеевна

Санкт-Петербург 2025

**Содержание**

[**Задание** 3](#_Toc199278777)

[**Выполнение задания** 4](#_Toc199278778)

[**Запрос 1** 4](#_Toc199278779)

[**1.1 Запрос 1 на языке PostgreSQL** 4](#_Toc199278780)

[**1.2 Индексы для запроса 1** 4](#_Toc199278781)

[**1.3 План выполнения без индексов для запроса 1** 5](#_Toc199278782)

[**1.4 EXPLAIN ANALYZE для запроса 1** 5](#_Toc199278783)

[**Запрос 2** 7](#_Toc199278784)

[**2.1 Запрос 2 на языке PostgreSQL** 7](#_Toc199278785)

[**2.2 Индексы для запроса 2** 7](#_Toc199278786)

[**2.3 План выполнения без индексов для запроса 2** 9](#_Toc199278787)

[**2.4 EXPLAIN ANALYZE для запроса 2** 11](#_Toc199278788)

[**Вывод:** 12](#_Toc199278789)

# **Задание**

Составить запросы на языке SQL (пункты 1-2).

Для каждого запроса предложить индексы, добавление которых уменьшит время выполнения запроса (указать таблицы/атрибуты, для которых нужно добавить индексы, написать тип индекса; объяснить, почему добавление индекса будет полезным для данного запроса).

Для запросов 1-2 необходимо составить возможные планы выполнения запросов. Планы составляются на основании предположения, что в таблицах отсутствуют индексы. Из составленных планов необходимо выбрать оптимальный и объяснить свой выбор.  
Изменятся ли планы при добавлении индекса и как?

Для запросов 1-2 необходимо добавить в отчет вывод команды EXPLAIN ANALYZE [запрос]

Подробные ответы на все вышеперечисленные вопросы должны присутствовать в отчете (планы выполнения запросов должны быть нарисованы, ответы на вопросы - представлены в текстовом виде).

1. Сделать запрос для получения атрибутов из указанных таблиц, применив фильтры по указанным условиям:  
   Н\_ТИПЫ\_ВЕДОМОСТЕЙ, Н\_ВЕДОМОСТИ.  
   Вывести атрибуты: Н\_ТИПЫ\_ВЕДОМОСТЕЙ.НАИМЕНОВАНИЕ, Н\_ВЕДОМОСТИ.ДАТА.  
   Фильтры (AND):  
   a) Н\_ТИПЫ\_ВЕДОМОСТЕЙ.ИД < 3.  
   b) Н\_ВЕДОМОСТИ.ИД = 1457443.  
   Вид соединения: LEFT JOIN.
2. Сделать запрос для получения атрибутов из указанных таблиц, применив фильтры по указанным условиям:  
   Таблицы: Н\_ЛЮДИ, Н\_ВЕДОМОСТИ, Н\_СЕССИЯ.  
   Вывести атрибуты: Н\_ЛЮДИ.ОТЧЕСТВО, Н\_ВЕДОМОСТИ.ЧЛВК\_ИД, Н\_СЕССИЯ.ДАТА.  
   Фильтры (AND):  
   a) Н\_ЛЮДИ.ИМЯ = Ярослав.  
   b) Н\_ВЕДОМОСТИ.ИД > 1250972.  
   c) Н\_СЕССИЯ.ИД < 27640.  
   Вид соединения: INNER JOIN.

# **Выполнение задания**

## **Запрос 1**

### **1.1 Запрос 1 на языке PostgreSQL**

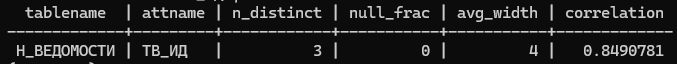
1. SELECT
2. Н\_ТИПЫ\_ВЕДОМОСТЕЙ.НАИМЕНОВАНИЕ,
3. Н\_ВЕДОМОСТИ.ДАТА
4. FROM
5. Н\_ТИПЫ\_ВЕДОМОСТЕЙ
6. LEFT JOIN
7. Н\_ВЕДОМОСТИ ON Н\_ТИПЫ\_ВЕДОМОСТЕЙ.ИД = Н\_ВЕДОМОСТИ.ТВ\_ИД
8. WHERE
9. Н\_ТИПЫ\_ВЕДОМОСТЕЙ.ИД < 3
10. AND Н\_ВЕДОМОСТИ.ИД = 1457443;

### **1.2 Индексы для запроса 1**

CREATE INDEX idx\_ведомости\_тв\_ид ON Н\_ВЕДОМОСТИ(ТВ\_ИД);

Тип: B-tree

Ускорит соединение с Н\_ВЕДОМОСТИ





Изображение выглядит как текст, снимок экрана, Шрифт, черный

Контент, сгенерированный ИИ, может содержать ошибки.

Как мы видим, всего 3 уникальных значения (на 222440 записей) и список хорошо отсортирован, то есть индекс не будет полезным (несмотря на это, такой индекс уже существует)

### **1.3 План выполнения без индексов для запроса 1**

Изображение выглядит как текст, линия, снимок экрана, диаграмма

Контент, сгенерированный ИИ, может содержать ошибки.

Как мы видим, первый план выполнения более оптимальный, так как сначала происходит отбор по условиям, а только потом объединение таблиц (то есть объединяется меньшее количество строк)

### **1.4 EXPLAIN ANALYZE для запроса 1**

Изображение выглядит как текст, снимок экрана, Шрифт

Контент, сгенерированный ИИ, может содержать ошибки.

Запрос соединяет таблицы "Н\_ТИПЫ\_ВЕДОМОСТЕЙ" и "Н\_ВЕДОМОСТИ" через Nested Loop.

Сначала выполняется полное сканирование "Н\_ТИПЫ\_ВЕДОМОСТЕЙ" с фильтром "ИД < 3", которое находит 2 строки и отбрасывает 1.

Затем для каждой из этих строк происходит поиск в "Н\_ВЕДОМОСТИ" по индексу первичного ключа "ВЕД\_PK" с условием "ИД = 1457443", каждый поиск занимает около 0.015 мс.

После этого применяется условие соединения ("Н\_ТИПЫ\_ВЕДОМОСТЕЙ"."ИД" = "Н\_ВЕДОМОСТИ"."ТВ\_ИД"), из двух возможных комбинаций остается одна строка, а вторая отбрасывается.

Общее время выполнения запроса — 0.122 мс, время планирования — 1.117 мс.

Изображение выглядит как текст, снимок экрана, диаграмма, линия

Контент, сгенерированный ИИ, может содержать ошибки.

## **Запрос 2**

### **2.1 Запрос 2 на языке PostgreSQL**

SELECT

    Н\_ЛЮДИ.ОТЧЕСТВО,

    Н\_ВЕДОМОСТИ.ЧЛВК\_ИД,

    Н\_СЕССИЯ.ДАТА

FROM

    Н\_ЛЮДИ

JOIN

    Н\_ВЕДОМОСТИ ON Н\_ЛЮДИ.ИД = Н\_ВЕДОМОСТИ.ЧЛВК\_ИД

JOIN

    Н\_СЕССИЯ ON Н\_ЛЮДИ.ИД = Н\_СЕССИЯ.ЧЛВК\_ИД

WHERE

    Н\_ЛЮДИ.ИМЯ = 'Ярослав'

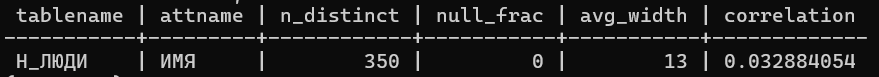
    AND Н\_ВЕДОМОСТИ.ИД > 1250972

    AND Н\_СЕССИЯ.ИД < 27640;

### **2.2 Индексы для запроса 2**

CREATE INDEX idx\_люди\_имя ON Н\_ЛЮДИ USING HASH (ИМЯ);

Тип: Hash (проверяется всего одно значение, для проверки равенства быстрее, чем B-tree)



Изображение выглядит как снимок экрана, черный, Шрифт, текст

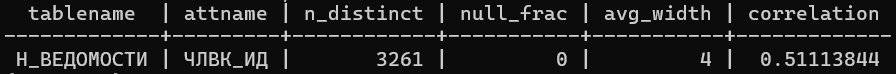
Контент, сгенерированный ИИ, может содержать ошибки.

Как мы видим, 350 уникальных значений (всего 5118), то есть примерно по 15 позиций на каждое имя, что не оправдывает использование индекса. Но, с другой стороны, значения плохо отсортированы, однако поддержание индекса будет дороже обходиться, чем использование seq scan на все имена. Поэтому индексом можно пренебречь.

CREATE INDEX idx\_ведомости\_члвк\_ид ON Н\_ВЕДОМОСТИ(ЧЛВК\_ИД);

Тип: B-tree

Ускоряет соединение с Н\_ЛЮДИ



Изображение выглядит как текст, снимок экрана, Шрифт, черный

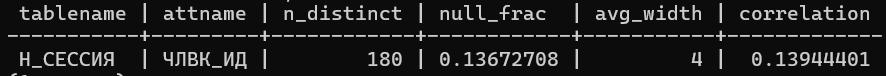
Контент, сгенерированный ИИ, может содержать ошибки.

Индекс уже существует, но если бы его не было, то его создание не было бы лишним – 3261 уникальных значений и отсортировано всего половина списка.

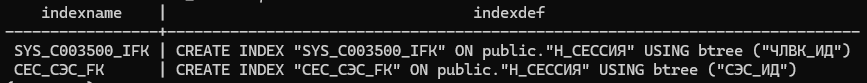
CREATE INDEX idx\_сессия\_члвк\_ид ON Н\_СЕССИЯ(ЧЛВК\_ИД);

Тип: B-tree

Ускорит соединение с Н\_ЛЮДИ







Индекс уже существует, но оценим, насколько он был бы необходим, если бы его не было. Всего 180 уникальных значений на 3752 (около 20 строк на одно значение), что не оправдывает использование индекса. Но, с другой стороны, значения плохо отсортированы, однако поддержание индекса будет дороже обходиться, чем использование seq scan на все значения. Поэтому индексом можно было бы и пренебречь.

### **2.3 План выполнения без индексов для запроса 2**

Изображение выглядит как текст, диаграмма, линия, График

Контент, сгенерированный ИИ, может содержать ошибки.

Изображение выглядит как текст, диаграмма, линия, График

Контент, сгенерированный ИИ, может содержать ошибки.

Как мы видим, первый план выполнения более оптимальный, так как используется левостороннее дерево, мы используем как можно раньше выборку после чего используем объединение таблиц.

Индексы сильно ускоряют запрос благодаря замене Seq Scan на Index Scan.

Для соединения таблиц Н\_ЛЮДИ и Н\_СЕССИЯ рекомендуется использовать Nested Loop Join. Это объясняется небольшим размером обеих таблиц (5,118 и 3,752 строк соответственно) и наличием эффективного индекса по полю соединения. Среднее количество сессий на одного человека составляет около 0.63, что делает такой тип соединения наиболее подходящим для данного случая. Время выполнения подобных запросов обычно не превышает 1 миллисекунды.

Для соединения таблиц Н\_ЛЮДИ и Н\_ВЕДОМОСТИ предпочтительнее применять Hash Join. Несмотря на небольшой размер таблицы Н\_ЛЮДИ (5,118 строк), таблица Н\_ВЕДОМОСТИ значительно больше (222,440 строк). Среднее количество ведомостей на одного человека составляет около 43.5, а система уже использует параллельное выполнение с двумя worker'ами. В таких условиях Hash Join показывает лучшую производительность со временем выполнения около 15-20 миллисекунд.

### **2.4 EXPLAIN ANALYZE для запроса 2**

Изображение выглядит как текст, снимок экрана, Шрифт

Контент, сгенерированный ИИ, может содержать ошибки.

Запрос использует Nested Loop для соединения таблиц "Н\_ЛЮДИ" и "Н\_ВЕДОМОСТИ" по условию "Н\_ЛЮДИ"."ИД" = "Н\_ВЕДОМОСТИ"."ЧЛВК\_ИД", но не находит подходящих строк (rows=0). Внутри этого соединения выполняется еще один Nested Loop между таблицами "Н\_СЕССИЯ" и "Н\_ЛЮДИ".

Сначала происходит Seq Scan таблицы "Н\_СЕССИЯ" с фильтром "ИД" < 27640, который обрабатывает 3460 строк и отбрасывает 292.

Для каждой из этих строк выполняется поиск в "Н\_ЛЮДИ" по индексу "ЧЛВК\_PK" с кешированием (Memoize), где проверяется условие "ИД" = "Н\_СЕССИЯ"."ЧЛВК\_ИД" и фильтр "ИМЯ" = 'Ярослав'.

Кеш срабатывает 3285 раз, а 175 раз происходит обращение к индексу, но ни одна строка не проходит фильтр.

Затем для каждой из 16 строк из первого соединения выполняется Index Scan по индексу "ВЕД\_ЧЛВК\_FK\_IFK" таблицы "Н\_ВЕДОМОСТИ" с условием "ЧЛВК\_ИД" = "Н\_СЕССИЯ"."ЧЛВК\_ИД" и фильтром "ИД" > 1250972, но также не находится подходящих строк.

Общее время выполнения запроса составляет 2.535 мс, время планирования — 1.126 мс.

Изображение выглядит как текст, диаграмма, линия, График

Контент, сгенерированный ИИ, может содержать ошибки.

# **Вывод:**

Во время выполнения данной лабораторной работы я разобрался, что такое индексы, научился их писать и применять. Понял, как можно и нужно оптимизировать sql-запросы.