МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное

образовательное учреждение высшего образования

«Национальный исследовательский университет ИТМО»

**ФАКУЛЬТЕТ ПРОГРАММНОЙ ИНЖЕНЕРИИ И КОМПЬЮТЕРНОЙ ТЕХНИКИ**

**ЛАБОРАТОРНАЯ РАБОТА № 4**

по дисциплине

«Базы данных»

Вариант № 5827

**Выполнил:**

студент группы Р3117

Агей Михаил  
  
**Преподаватель:**

Инячина Диана

Санкт-Петербург

2023

[**Текст задания 3**](#_5lc2vvb6szbi)

[**Запрос 1 4**](#_932o8tf8td55)

[Варианты построения плана 4](#_se1s3wu14pk)

[Анализ соединений 4](#_bp3hguhh58wj)

[Оптимизация 4](#_obxctwq5pqku)

[Анализ оптимизации 5](#_n1iznjrepqm6)

[**Запрос 2 7**](#_sduz2y1qg04q)

[**Общая концепция влияния индексов на работу запросов 8**](#_y266wj1gz07b)

[**Вывод 8**](#_a8o0lnn708tg)

# 

# Текст задания

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).

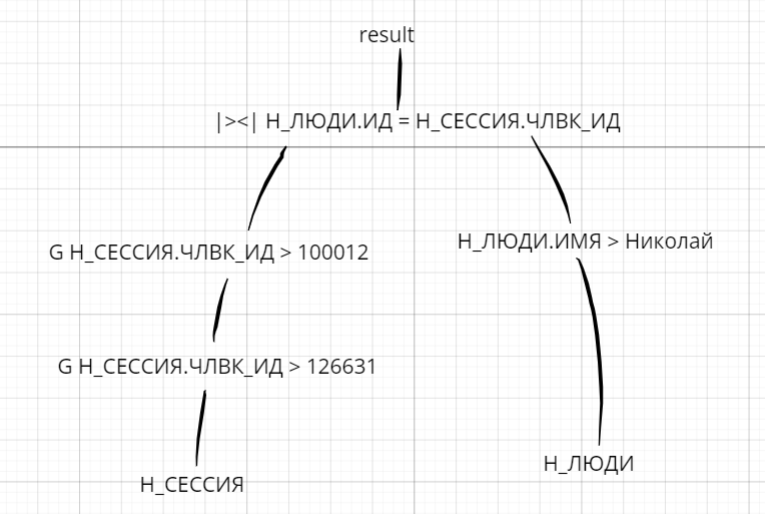
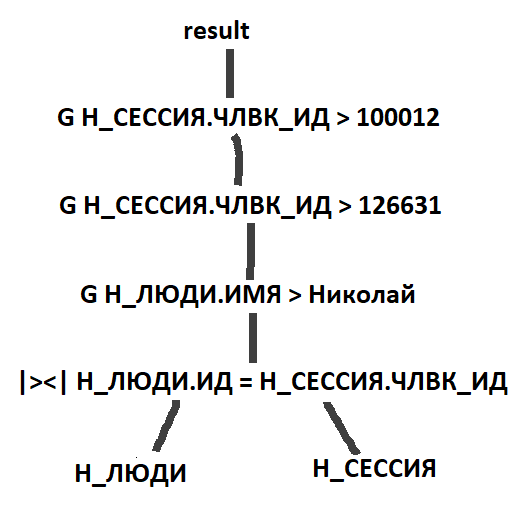
1. Сделать запрос для получения атрибутов из указанных таблиц, применив фильтры по указанным условиям:  
   Таблицы: Н\_ЛЮДИ, Н\_СЕССИЯ.  
   Вывести атрибуты: Н\_ЛЮДИ.ИД, Н\_СЕССИЯ.ДАТА.  
   Фильтры (AND):  
   a) Н\_ЛЮДИ.ИМЯ > Николай.  
   b) Н\_СЕССИЯ.ЧЛВК\_ИД > 126631.  
   c) Н\_СЕССИЯ.ЧЛВК\_ИД > 100012.  
   Вид соединения: INNER 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.ФАМИЛИЯ, Н\_ВЕДОМОСТИ.ИД, Н\_СЕССИЯ.ЧЛВК\_ИД.  
   Фильтры (AND):  
   a) Н\_ЛЮДИ.ОТЧЕСТВО < Владимирович.  
   b) Н\_ВЕДОМОСТИ.ИД = 1457443.  
   Вид соединения: RIGHT JOIN.

# 

# Запрос 1

SELECT Н\_ЛЮДИ.ИД, Н\_СЕССИЯ.ДАТА  
FROM Н\_ЛЮДИ   
INNER JOIN Н\_СЕССИЯ  
ON Н\_ЛЮДИ.ИД = Н\_СЕССИЯ.ЧЛВК\_ИД  
WHERE Н\_ЛЮДИ.ИМЯ > 'Николай'   
AND Н\_СЕССИЯ.ЧЛВК\_ИД > 126631  
AND Н\_СЕССИЯ.ЧЛВК\_ИД > 100012

## Варианты построения плана



## Анализ соединений

1 вариант - соединение первым шагом, 2 вариант - последним. Существуют также их комбинации. Например, соединить таблицу после 1 или 2 операций фильтрации.

Самым оптимальным считаю план 2, потому что в нем дополнительное соединение производится на последнем шаге, а выборки как можно раньше. Что уменьшает время работы промежуточных операций

## Оптимизация

Очевидная оптимизация - убрать условие Н\_СЕССИЯ.ЧЛВК\_ИД > 100012, так как оно является избыточным при наличии условия Н\_СЕССИЯ.ЧЛВК\_ИД > 126631.  
Вторая оптимизация: для таблицы Н\_СЕССИЯ можно создать индекс для столбца ЧЛВК\_ИД:

*CREATE INDEX ЧЛВК\_ИД\_ИНДЕКС*

*ON Н\_СЕССИЯ*

*USING btree(ЧЛВК\_ИД);*

Благодаря данному индексу выборка (фильтрация) Н\_СЕССИЯ.ЧЛВК\_ИД > 126631 может производиться быстрее. B-tree индекс будет оптимальным в данном случае, потому что он отлично подходит для ускорения выборки с числовым неравенством

Третья оптимизация: для таблицы Н\_ЛЮДИ можно создать индекс для столбца ИМЯ:

*CREATE INDEX ИМЯ\_ИНДЕКС*

*ON Н\_ЛЮДИ*

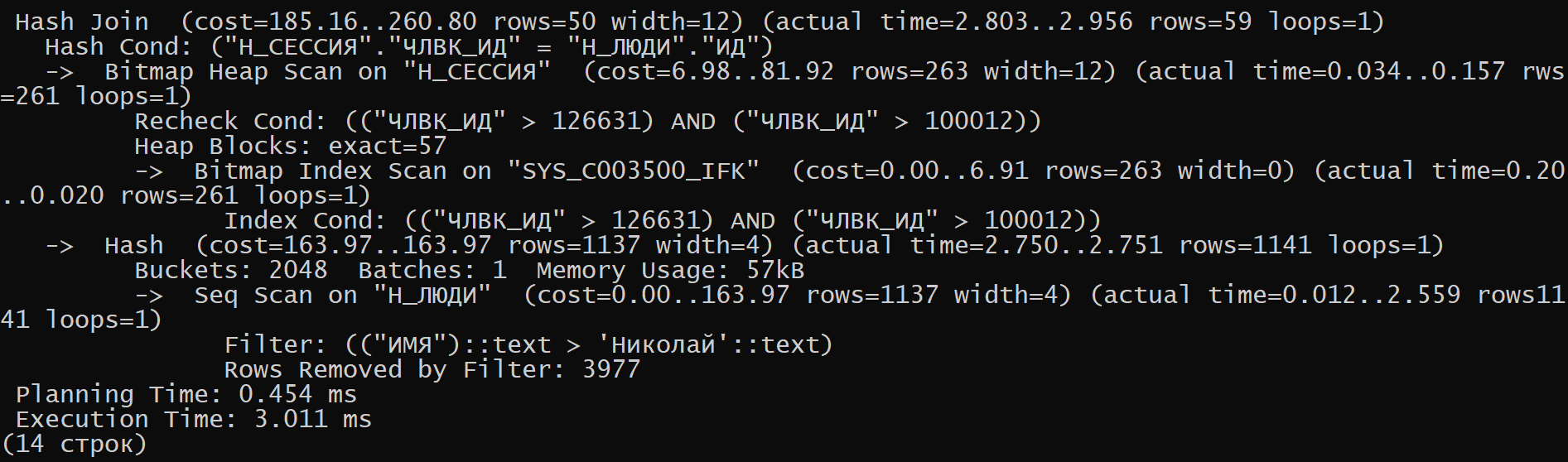
*USING btree(ИМЯ);*

Благодаря данному индексу выборка (фильтрация) Н\_ЛЮДИ.ИМЯ > 'Николай' может производиться быстрее. B-tree индекс будет оптимальным в данном случае, потому что он отлично подходит для ускорения выборки с неравенством при сравнении строк

## Анализ плана выполнения запроса

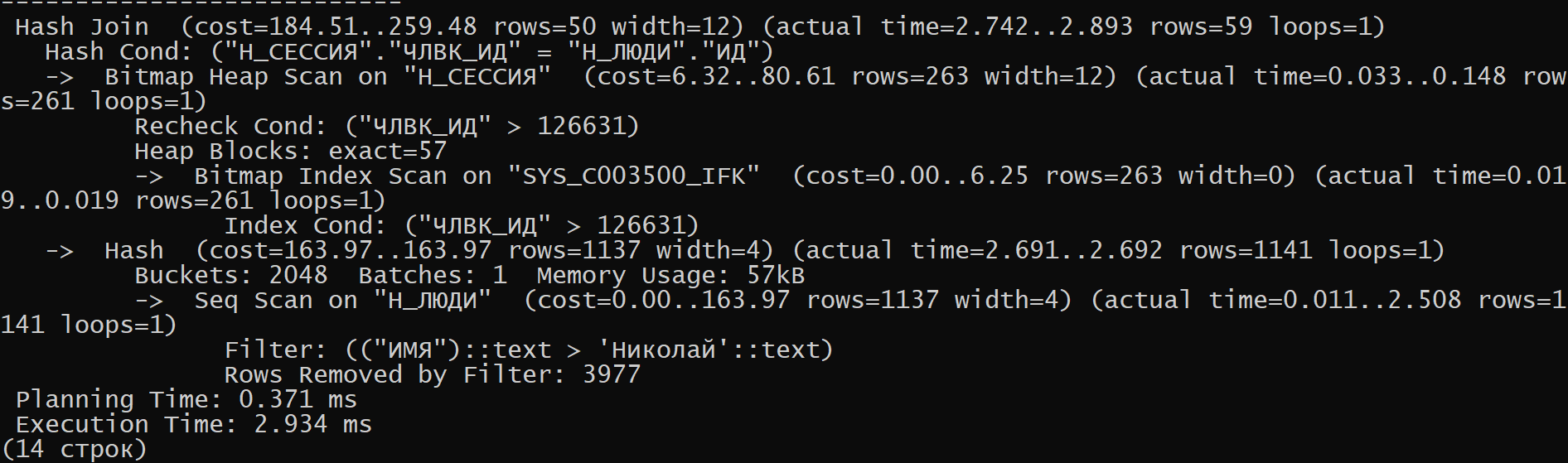
Для сокращения записи в примерах основной запрос обозначим переменной **QUERY** => изначально и после каждой оптимизации будем выполнять команду EXPLAIN ANALYZE **QUERY**

*До оптимизаций:*



Planning Time: **0.454** ms

Execution Time: **3.011** ms

*После удаления последнего избыточного условия поиска*  


Planning Time: **0.371** ms

Execution Time: **2.934** ms

# Запрос 2

SELECT Н\_ЛЮДИ.ФАМИЛИЯ, Н\_ВЕДОМОСТИ.ИД, Н\_СЕССИЯ.ЧЛВК\_ИД  
FROM Н\_ЛЮДИ   
RIGHT JOIN Н\_СЕССИЯ  
ON Н\_ЛЮДИ.ИД = Н\_СЕССИЯ.ЧЛВК\_ИД  
RIGHT JOIN Н\_ВЕДОМОСТИ  
ON Н\_ЛЮДИ.ИД = Н\_ВЕДОМОСТИ.ЧЛВК\_ИД  
WHERE Н\_ЛЮДИ.ОТЧЕСТВО < 'Владимирович'  
AND Н\_ВЕДОМОСТИ.ИД = 1457443

## Варианты построения плана

# 

## Анализ соединений

1 вариант: сначала происходит выборка в 2 таблицах, после они соединяются друг с другом и в конце с Н\_СЕССИИ.

2 вариант: выборка происходит сначала только по условию Н\_ВЕДОМОСТИ.ИД = 1457443. После обе таблицы, на записи которых налагаются условия соединяются, и только после этого происходит выборка по условию таблицы Н\_ЛЮДИ. В конце таблицы соединяются с Н\_СЕССИИ.

Существуют также другие менее рациональные комбинации. Например, соединить сразу 3 таблицы, а после этого делать выборки по условиям.

Самым оптимальным считаю план 2. Благодаря условию Н\_ВЕДОМОСТИ.ИД = 1457443, после выборки в этой таблице останется не более 1 записи (т к ИД - первичный ключ). Соединение - RIGHT JOIN и связь Н\_ВЕДОМОСТИ - Н\_ЛЮДИ - многие к 1. То есть после соединения таблиц останется не более 1 записи. Это многократно ускорит выборку по условию Н\_ЛЮДИ.ОТЧЕСТВО < 'Владимирович'.  
В варианте 1 пришлось бы перебирать все записи таблицы Н\_ЛЮДИ. В этом главное преимущество варианта 2 перед вариантом 1.   
В данном запросе нет условий выборки для таблицы Н\_СЕССИИ, поэтому присоединяем ее в конце, чтобы уменьшить количество промежуточных связей.

## Оптимизация

Уже осуществленная оптимизация:Hash-индекс для Н\_ВЕДОМОСТИ.ИД. Hash-индекс потому что в запросе используется условие с равенством: Н\_ВЕДОМОСТИ.ИД = 1457443, а данный вид индексов как раз отлично подходит для оптимизации запросов с прямым сравнением. Оптимизация уже осуществлена, потому что Н\_ВЕДОМОСТИ.ИД это первичный ключ. А при создании первичного ключа для него автоматически создается индекс.

Предлагаемая оптимизация: для таблицы Н\_ЛЮДИ можно создать индекс для столбца ОТЧЕСТВО:

*CREATE INDEX ОТЧЕСТВО\_ИНДЕКС ON Н\_ЛЮДИ USING btree(ОТЧЕСТВО);*

Благодаря данному индексу выборка (фильтрация) Н\_ЛЮДИ.ОТЧЕСТВО > 'Владимирович' может производиться быстрее. B-tree индекс будет оптимальным в данном случае, потому что он отлично подходит для ускорения выборки с неравенством при сравнении строк

## Анализ плана выполнения запроса

# 

Planning Time: **0.783** ms

Execution Time: **0.076** ms

Время выполнения запроса более чем в 10 раз меньше времени определения оптимального плана выполнения запроса. Это произошло благодаря тому, что был выбран оптимальный план выполнения запроса (вариант 2), что позволило избежать нескольких полных переборов таблиц, получив не более чем 1 запись после первой выборки.

# Общая концепция влияния индексов на работу запросов

При добавлении индексов (с учетом того, что содержание и количество индексов выбраны правильно) время выполнения запроса может значительно уменьшиться. С другой стороны возрастает время выбора рационального плана выполнения, а также появляются дополнительные издержки на обновление индексов при изменения таблицы, и затрачивается место в памяти на хранения индексов. Соответственно при слишком большом количестве индексов и непродуманном их создании время работы запроса может увеличиться.

# Вывод

Выполнив лабораторную работу №4 по Базам данных, я узнал, что такое индексы в СУБД, на какие типы они делятся и как их создавать. Проанализировал 2 запроса из базы данных, предложив действия по оптимизации данных запросов путем создания индексов.