Факультет Программной Инженерии и Компьютерной техники

**Лабораторная работа №4**

**Вариант №37646275**

Выполнила:

Брель Мария Владимировна

Группа Р3107

Преподаватели:

Байрамова Хумай

Николаев Владимир Вячеславович

**Оглавление**

[Задание: 3](#__RefHeading___Toc1941_1931527120)

[Запрос №1: 4](#__RefHeading___Toc3985_2680889073)

[Индексы: 4](#__RefHeading___Toc2795_1612188114)

[Результат EXPLAIN ANALYSE: 5](#__RefHeading___Toc2797_1612188114)

[Запрос №2: 6](#__RefHeading___Toc3985_26808890731)

[Индексы: 6](#__RefHeading___Toc2795_16121881141)

[Результат EXPLAIN ANALYSE: 7](#__RefHeading___Toc2797_16121881141)

[Вывод: 8](#__RefHeading___Toc973_1545597831)

# **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?

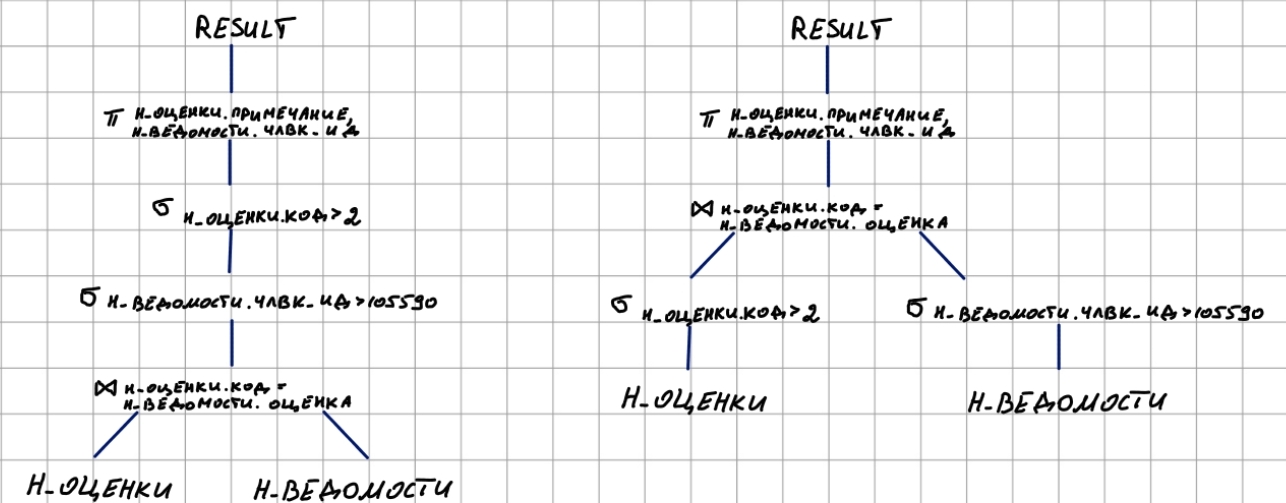
Для запросов 1-2 необходимо добавить в отчет вывод команды EXPLAIN ANALYZE [запрос]

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:  
   Таблицы: Н\_ОЦЕНКИ, Н\_ВЕДОМОСТИ.  
   Вывести атрибуты: Н\_ОЦЕНКИ.ПРИМЕЧАНИЕ, Н\_ВЕДОМОСТИ.ЧЛВК\_ИД.  
   Фильтры (AND):  
   a) Н\_ОЦЕНКИ.КОД > 2.  
   b) Н\_ВЕДОМОСТИ.ЧЛВК\_ИД > 105590.  
   Вид соединения: INNER JOIN.
2. Сделать запрос для получения атрибутов из указанных таблиц, применив фильтры по указанным условиям:  
   Таблицы: Н\_ЛЮДИ, Н\_ОБУЧЕНИЯ, Н\_УЧЕНИКИ.  
   Вывести атрибуты: Н\_ЛЮДИ.ИМЯ, Н\_ОБУЧЕНИЯ.НЗК, Н\_УЧЕНИКИ.НАЧАЛО.  
   Фильтры: (AND)  
   a) Н\_ЛЮДИ.ОТЧЕСТВО = Георгиевич.  
   b) Н\_ОБУЧЕНИЯ.НЗК > 001000.  
   c) Н\_УЧЕНИКИ.НАЧАЛО = 1996-09-01.  
   Вид соединения: INNER JOIN.

# **Запрос №1:** Таблицы: Н\_ОЦЕНКИ, Н\_ВЕДОМОСТИ. Вывести атрибуты: Н\_ОЦЕНКИ.ПРИМЕЧАНИЕ, Н\_ВЕДОМОСТИ.ЧЛВК\_ИД. Фильтры (AND): a) Н\_ОЦЕНКИ.КОД > 2. b) Н\_ВЕДОМОСТИ.ЧЛВК\_ИД > 105590. Вид соединения: INNER JOIN.

|  |
| --- |
| SELECT "Н\_ОЦЕНКИ"."ПРИМЕЧАНИЕ", "Н\_ВЕДОМОСТИ"."ЧЛВК\_ИД" FROM "Н\_ВЕДОМОСТИ" JOIN "Н\_ОЦЕНКИ" ON "Н\_ОЦЕНКИ"."КОД" = "Н\_ВЕДОМОСТИ"."ОЦЕНКА" WHERE "Н\_ОЦЕНКИ"."КОД" IN ('3','4','5') AND "Н\_ВЕДОМОСТИ"."ЧЛВК\_ИД">105590; |

Оптимальным является план №2, так как он производит объединение таблиц по ранее выбранным атрибутам, а не по таблицам целиком.

## **Индексы:**

|  |
| --- |
| CREATE INDEX "ИНДЕКС\_ОЦЕНКИ\_КОД" ON "Н\_ОЦЕНКИ" USING btree("КОД"); CREATE INDEX "ИНДЕКС\_ВЕДОМОСТИ\_ЧЛВК\_ИД" ON "Н\_ВЕДОМОСТИ" USING btree("ЧЛВК\_ИД"); CREATE INDEX "ИНДЕКС\_ВЕДОМОСТИ\_ОЦЕНКА" ON "Н\_ВЕДОМОСТИ" USING hash("ОЦЕНКА"); |

Добавление этих индексов должно ускорить выполнение запросов, так как по перечисленным полям происходит выборка с использованием оператора сравнения. Так же быстрее будет происходить соединение таблиц. В первых двух случаях используются операторы сравнения „>“ и „<“, так что эффективнее использовать btree. В третьем используется прямое сравнение, так что эффективнее использовать хэш-индекс.

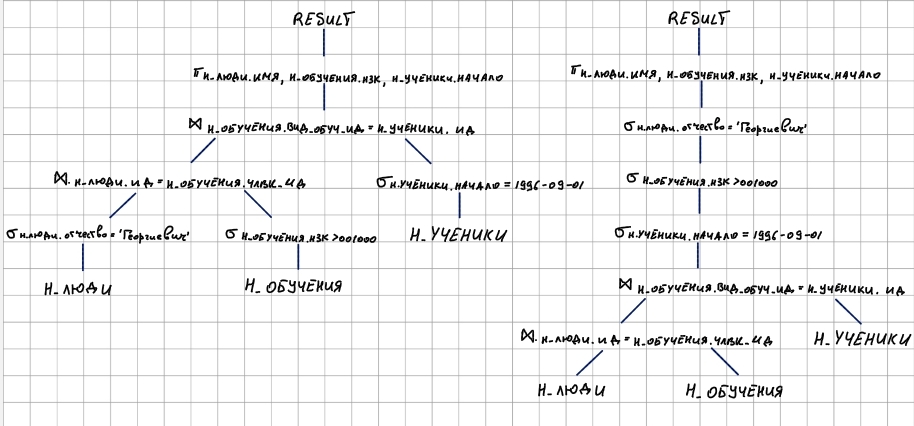
При добавлении индексов планы выполнения запросов изменятся, так как будет происходить индексный скан и Hash Join станет быстрее благодаря индексам.

## **Результат EXPLAIN ANALYSE:**

|  |
| --- |
| Hash Join (cost=1.16..7706.27 rows=74125 width=26) (actual time=0.080..143.217 rows=107434 loops=1)  Hash Cond: (("Н\_ВЕДОМОСТИ"."ОЦЕНКА")::text = ("Н\_ОЦЕНКИ"."КОД")::text)  -> Seq Scan on "Н\_ВЕДОМОСТИ" (cost=0.00..6846.50 rows=222375 width=10) (actual time=0.023..64.523 rows=222413 loops=1)  Filter: ("ЧЛВК\_ИД" > 105590)  Rows Removed by Filter: 27  -> Hash (cost=1.12..1.12 rows=3 width=27) (actual time=0.030..0.032 rows=3 loops=1)  Buckets: 1024 Batches: 1 Memory Usage: 9kB  -> Seq Scan on "Н\_ОЦЕНКИ" (cost=0.00..1.12 rows=3 width=27) (actual time=0.009..0.011 rows=3 loops=1)  Filter: (("КОД")::text = ANY ('{3,4,5}'::text[]))  Rows Removed by Filter: 6  Planning Time: 2.091 ms  Execution Time: 152.891 ms |

# **Запрос №2:** Таблицы: Н\_ЛЮДИ, Н\_ОБУЧЕНИЯ, Н\_УЧЕНИКИ. Вывести атрибуты: Н\_ЛЮДИ.ИМЯ, Н\_ОБУЧЕНИЯ.НЗК, Н\_УЧЕНИКИ.НАЧАЛО. Фильтры: (AND) a) Н\_ЛЮДИ.ОТЧЕСТВО = Георгиевич. b) Н\_ОБУЧЕНИЯ.НЗК > 001000. c) Н\_УЧЕНИКИ.НАЧАЛО = 1996-09-01. Вид соединения: INNER JOIN.

|  |
| --- |
| SELECT "Н\_ЛЮДИ"."ИМЯ", "Н\_ОБУЧЕНИЯ"."НЗК", "Н\_УЧЕНИКИ"."НАЧАЛО" FROM "Н\_ЛЮДИ" JOIN "Н\_ОБУЧЕНИЯ" ON "Н\_ЛЮДИ"."ИД" = "Н\_ОБУЧЕНИЯ"."ЧЛВК\_ИД" JOIN "Н\_УЧЕНИКИ" ON "Н\_ОБУЧЕНИЯ"."ВИД\_ОБУЧ\_ИД" = "Н\_УЧЕНИКИ"."ИД" WHERE "Н\_ЛЮДИ"."ОТЧЕСТВО" = 'Георгиевич' AND "Н\_ОБУЧЕНИЯ"."НЗК" > '001000' AND "Н\_УЧЕНИКИ"."НАЧАЛО" = '1996-09-01'; |

1. Оптимальным является план №1, так как он производит объединение таблиц по ранее выбранным атрибутам, а не по таблицам целиком.

## **Индексы:**

|  |
| --- |
| CREATE INDEX "ИНДЕКС\_ЛЮДИ\_ОТЧЕСТВО" ON "Н\_ЛЮДИ" USING hash("ОТЧЕСТВО");  CREATE INDEX "ИНДЕКС\_ОБУЧЕНИЯ\_ЧЛВК\_ИД" ON "Н\_ОБУЧЕНИЯ" USING btree("ЧЛВК\_ИД"); CREATE INDEX "ИНДЕКС\_ОБУЧЕНИЯ\_ВИД\_ОБУЧЕНИЯ\_ИД" ON "Н\_ОБУЧЕНИЯ" USING btree("ВИД\_ОБУЧ\_ИД"); CREATE INDEX "ИНДЕКС\_ОБУЧЕНИЯ\_НЗК" ON "Н\_ОБУЧЕНИЯ" USING btree("НЗК");  CREATE INDEX "ИНДЕКС\_УЧЕНИКИ\_ИД" ON "Н\_УЧЕНИКИ" USING btree("ИД"); CREATE INDEX "ИНДЕКС\_УЧЕНИКИ\_НАЧАЛО" ON "Н\_УЧЕНИКИ" USING hash("НАЧАЛО"); |

Добавление этих индексов должно ускорить выполнение запросов, так как по перечисленным полям происходит выборка с использованием оператора сравнения. Так же быстрее будет происходить соединение таблиц. В первом и последнем случаях происходит прямое сравнение, так что эффективнее использовать хэш-индексы. В остальных случаях используются операторы сравнения „>“ и „<“, так что эффективнее использовать btree.

При добавлении индексов планы выполнения запросов изменятся, так как будет происходить индексный скан и Nested Loop Join станет быстрее благодаря индексам.

## **Результат EXPLAIN ANALYSE:**

|  |
| --- |
| Nested Loop (cost=0.85..257.57 rows=1 width=27) (actual time=0.161..13.270 rows=21 loops=1)  -> Merge Join (cost=0.57..199.22 rows=75 width=18) (actual time=0.045..3.922 rows=4387 loops=1)  Merge Cond: ("Н\_ОБУЧЕНИЯ"."ВИД\_ОБУЧ\_ИД" = "Н\_УЧЕНИКИ"."ИД")  -> Index Scan using "ОБУЧ\_ВО\_FK\_I" on "Н\_ОБУЧЕНИЯ" (cost=0.28..180.40 rows=4387 width=14) (actual time=0.032..2.812 rows=4387 loops=1)  Filter: (("НЗК")::text > '001000'::text)  Rows Removed by Filter: 634  -> Index Scan using "УЧЕН\_PK" on "Н\_УЧЕНИКИ" (cost=0.29..2722.68 rows=400 width=12) (actual time=0.009..0.011 rows=2 loops=1)  Filter: ("НАЧАЛО" = '1996-09-01 00:00:00'::timestamp without time zone)  -> Index Scan using "ЧЛВК\_PK" on "Н\_ЛЮДИ" (cost=0.28..0.78 rows=1 width=17) (actual time=0.002..0.002 rows=0 loops=4387)  Index Cond: ("ИД" = "Н\_ОБУЧЕНИЯ"."ЧЛВК\_ИД")  Filter: (("ОТЧЕСТВО")::text = 'Георгиевич'::text)  Rows Removed by Filter: 1  Planning Time: 1.757 ms  Execution Time: 13.347 ms |

# **Вывод:** в ходе выполнения данной лабораторной работы я научилась оптимизировать запросы, продумывать более выгодные планы выполнения запросов и использовать для этого разные виды индексов.