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 по

Базам данных №4

Вариант 3

Работу выполнила:

Касьяненко В. М.

Группа:

P3120

Преподаватель:

Королева Ю. А.

Санкт-Петербург,

2023

**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.

1. Сделать запрос для получения атрибутов из указанных таблиц, применив фильтры по указанным условиям:

Таблицы: Н\_ЛЮДИ, Н\_ВЕДОМОСТИ.

Вывести атрибуты: Н\_ЛЮДИ.ИМЯ, Н\_ВЕДОМОСТИ.ИД.

Фильтры (AND):

a) Н\_ЛЮДИ.ФАМИЛИЯ < Иванов.

b) Н\_ВЕДОМОСТИ.ИД = 1250981.

c) Н\_ВЕДОМОСТИ.ИД = 1250972.

Вид соединения: RIGHT JOIN.

1. 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.ДАТА, Н\_СЕССИЯ.ИД.

Фильтры (AND):

a) Н\_ЛЮДИ.ИД < 100012.

b) Н\_ВЕДОМОСТИ.ИД = 1457443.

Вид соединения: RIGHT JOIN.

**Запросы**

SELECT "Н\_ЛЮДИ"."ИМЯ","Н\_ЛЮДИ"."ФАМИЛИЯ", "Н\_ВЕДОМОСТИ"."ИД"

FROM "Н\_ЛЮДИ"

RIGHT JOIN "Н\_ВЕДОМОСТИ" ON "Н\_ЛЮДИ"."ИД" = "Н\_ВЕДОМОСТИ"."ЧЛВК\_ИД"

WHERE "Н\_ЛЮДИ"."ФАМИЛИЯ" < 'Иванов'

AND "Н\_ВЕДОМОСТИ"."ИД" IN (1250981, 1250972);

SELECT "Н\_ЛЮДИ"."ОТЧЕСТВО", "Н\_ВЕДОМОСТИ"."ДАТА", "Н\_СЕССИЯ"."ИД"

FROM "Н\_ЛЮДИ"

RIGHT JOIN "Н\_ВЕДОМОСТИ" ON "Н\_ЛЮДИ"."ИД" = "Н\_ВЕДОМОСТИ"."ЧЛВК\_ИД"

RIGHT JOIN "Н\_СЕССИЯ" ON "Н\_ВЕДОМОСТИ"."СЭС\_ИД" = "Н\_СЕССИЯ"."СЭС\_ИД"

WHERE "Н\_ЛЮДИ"."ИД" < 100012

AND "Н\_ВЕДОМОСТИ"."ИД" = 1457443

**Добавление индексов**

Для первого запроса можно добавить индексы на следующие атрибуты и таблицы:

* Н\_ЛЮДИ.ФАМИЛИЯ - индекс типа B-Tree
* Н\_ВЕДОМОСТИ.ИД - индекс типа B-Tree
* Н\_ВЕДОМОСТИ.ЧЛВК\_ИД - индекс типа B-Tree

1. Индекс на атрибут "ФАМИЛИЯ" в таблице "Н\_ЛЮДИ" позволит ускорить выполнение операции WHERE. Без индекса, для выполнения данного условия WHERE придется просканировать все строки таблицы "Н\_ЛЮДИ". С индексом, системе удастся быстро найти нужные строки, что существенно ускорит выполнение запроса.
2. Индекс на атрибут "ИД" в таблице "Н\_ВЕДОМОСТИ" также позволит ускорить выполнение запроса. Без индекса придется просканировать всю таблицу, чтобы найти строки с нужными идентификаторами. С индексом система быстро найдет нужные строки, что ускорит выполнение запроса.
3. Индекс на атрибут "ЧЛВК\_ИД" в таблице "Н\_ВЕДОМОСТИ" поможет оптимизировать операцию объединения таблиц (JOIN). Без индекса системе придется выполнить вложенный цикл, просматривая каждую строку таблицы "Н\_ВЕДОМОСТИ" и для каждой строки искать соответствующую строку в таблице "Н\_ЛЮДИ". С индексом по атрибуту "ЧЛВК\_ИД" система сможет быстро найти нужные строки в таблице "Н\_ЛЮДИ", что существенно ускорит выполнение запроса.

Для второго запроса можно добавить B-Tree индексы на следующие атрибуты:

1. "Н\_ЛЮДИ"."ИД" - для таблицы "Н\_ЛЮДИ"
2. "Н\_ВЕДОМОСТИ"."ЧЛВК\_ИД" - для таблицы "Н\_ВЕДОМОСТИ"
3. "Н\_ВЕДОМОСТИ"."СЭС\_ИД" - для таблицы "Н\_ВЕДОМОСТИ"

* Первый индекс на "Н\_ЛЮДИ"."ИД" позволит ускорить выборку строк из таблицы "Н\_ЛЮДИ", соответствующих условию "Н\_ЛЮДИ"."ИД" < 100012.
* Второй индекс на "Н\_ВЕДОМОСТИ"."ЧЛВК\_ИД" поможет оптимизировать выполнение JOIN между таблицами "Н\_ЛЮДИ" и "Н\_ВЕДОМОСТИ", поскольку этот индекс будет использоваться для объединения строк в этих таблицах.
* Третий индекс на "Н\_ВЕДОМОСТИ"."СЭС\_ИД" ускорит выполнение JOIN между таблицами "Н\_ВЕДОМОСТИ" и "Н\_СЕССИЯ", так как этот индекс будет использоваться для объединения строк в этих таблицах.

**Планы выполнения запросов**

Планы выполнения запросов без индексов для первого запроса:

1. Производится полный скан таблицы "Н\_ЛЮДИ" с применением фильтра "ФАМИЛИЯ" < ‘Иванов’, затем производится полный скан таблицы "Н\_ВЕДОМОСТИ" с фильтром "ИД" IN (1250981, 1250972), затем производится объединение результатов с помощью операции RIGHT JOIN.
2. Производится полный скан таблицы "Н\_ВЕДОМОСТИ" с применением фильтра "ИД" IN (1250981, 1250972), затем производится полный скан таблицы "Н\_ЛЮДИ" с фильтром "ФАМИЛИЯ" < ‘Иванов’, затем производится объединение результатов с помощью операции RIGHT JOIN.

Оптимальным планом выполнения данного запроса является первый вариант, так как фильтр по фамилии применяется к таблице "Н\_ЛЮДИ" до выполнения объединения, что позволяет уменьшить количество строк, обрабатываемых в запросе.

При добавлении индекса на столбец "ФАМИЛИЯ" таблицы "Н\_ЛЮДИ" план выполнения запроса изменится следующим образом:

1. Производится индексный поиск в таблице "Н\_ЛЮДИ" по фильтру "ФАМИЛИЯ" < 'Иванов', затем производится полный скан таблицы "Н\_ВЕДОМОСТИ" с фильтром "ИД" IN (1250981, 1250972), затем производится объединение результатов с помощью операции RIGHT JOIN.
2. Производится полный скан таблицы "Н\_ВЕДОМОСТИ" с применением фильтра "ИД" IN (1250981, 1250972), затем производится индексный поиск в таблице "Н\_ЛЮДИ" по фильтру "ФАМИЛИЯ" < 'Иванов', затем производится объединение результатов с помощью операции RIGHT JOIN.

При добавлении индекса на столбец "ФАМИЛИЯ" таблицы "Н\_ЛЮДИ" план выполнения запроса станет оптимальным, так как будет использован индексный поиск вместо полного скана таблицы "Н\_ЛЮДИ", что уменьшит время выполнения запроса.

Для второго запроса можно составить несколько возможных планов выполнения:

1. План с использованием полного сканирования таблиц. В этом случае для выполнения запроса будут просмотрены все строки таблиц "Н\_ЛЮДИ", "Н\_ВЕДОМОСТИ" и "Н\_СЕССИЯ", а затем произойдет сортировка и объединение результатов.
2. План с использованием индекса только на таблице "Н\_ВЕДОМОСТИ". В этом случае будут просмотрены только те строки таблицы "Н\_ВЕДОМОСТИ", которые удовлетворяют условию "Н\_ВЕДОМОСТИ"."ИД" = 1457443, затем произойдет объединение с таблицами "Н\_ЛЮДИ" и "Н\_СЕССИЯ" при помощи соединений по полям "Н\_ЛЮДИ"."ИД" и "Н\_ВЕДОМОСТИ"."СЭС\_ИД" соответственно.
3. План с использованием индексов на таблицах "Н\_ЛЮДИ" и "Н\_ВЕДОМОСТИ". В этом случае будут просмотрены только те строки таблиц "Н\_ЛЮДИ" и "Н\_ВЕДОМОСТИ", которые удовлетворяют условиям "Н\_ЛЮДИ"."ИД" < 100012 и "Н\_ВЕДОМОСТИ"."ИД" = 1457443 соответственно, затем произойдет объединение с таблицей "Н\_СЕССИЯ" при помощи соединения по полю "Н\_ВЕДОМОСТИ"."СЭС\_ИД". Также возможна сортировка результата.

Оптимальным планом выполнения запроса является план №2, который использует индекс на таблице "Н\_ВЕДОМОСТИ" и позволяет сократить количество просматриваемых строк.

При добавлении индекса на таблицу "Н\_ЛЮДИ" планы могут измениться, так как запрос может начать использовать индекс на этой таблице в сочетании с индексом на таблице "Н\_ВЕДОМОСТИ", что также может привести к сокращению количества просматриваемых строк. Однако, при добавлении индекса на таблицу "Н\_СЕССИЯ", планы измениться не должны, так как в данном запросе не используется условие для этой таблицы в качестве фильтра.

**EXPLAIN ANALYZE**

Начало формы

Конец формы

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Автоматически созданное описание

Из результатов можно увидеть, что для выполнения запроса был выбран план выполнения с использованием двух вложенных циклов (Nested Loop). Первый вложенный цикл соединяет таблицы "Н\_ЛЮДИ" и "Н\_ВЕДОМОСТИ" с использованием условия соединения по полю "ЧЛВК\_ИД". Второй вложенный цикл не используется, так как в запросе нет условий соединения таблицы "Н\_ВЕДОМОСТИ" с таблицей "Н\_СЕССИЯ".

Также можно увидеть стоимость плана выполнения (cost), которая равна 0.70, и количество строк, которые были ожидаемо обработаны (rows), равное 1. В данном случае запрос был выполнен очень быстро (Execution Time: 0.065 ms), что свидетельствует о том, что он не нагружает базу данных.

Из описания индексных сканов можно сделать вывод, что для оптимизации запроса были использованы индексы, что позволяет ускорить выполнение запроса.

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Автоматически созданное описание

Из результатов можно увидеть, что для выполнения запроса был выбран план выполнения с использованием двух вложенных циклов (Nested Loop). Первый вложенный цикл соединяет таблицы "Н\_ЛЮДИ" и "Н\_ВЕДОМОСТИ" с использованием условия соединения по полю "ЧЛВК\_ИД". Второй вложенный цикл соединяет таблицы "Н\_ВЕДОМОСТИ" и "Н\_СЕССИЯ" с использованием условия соединения по полю "СЭС\_ИД".

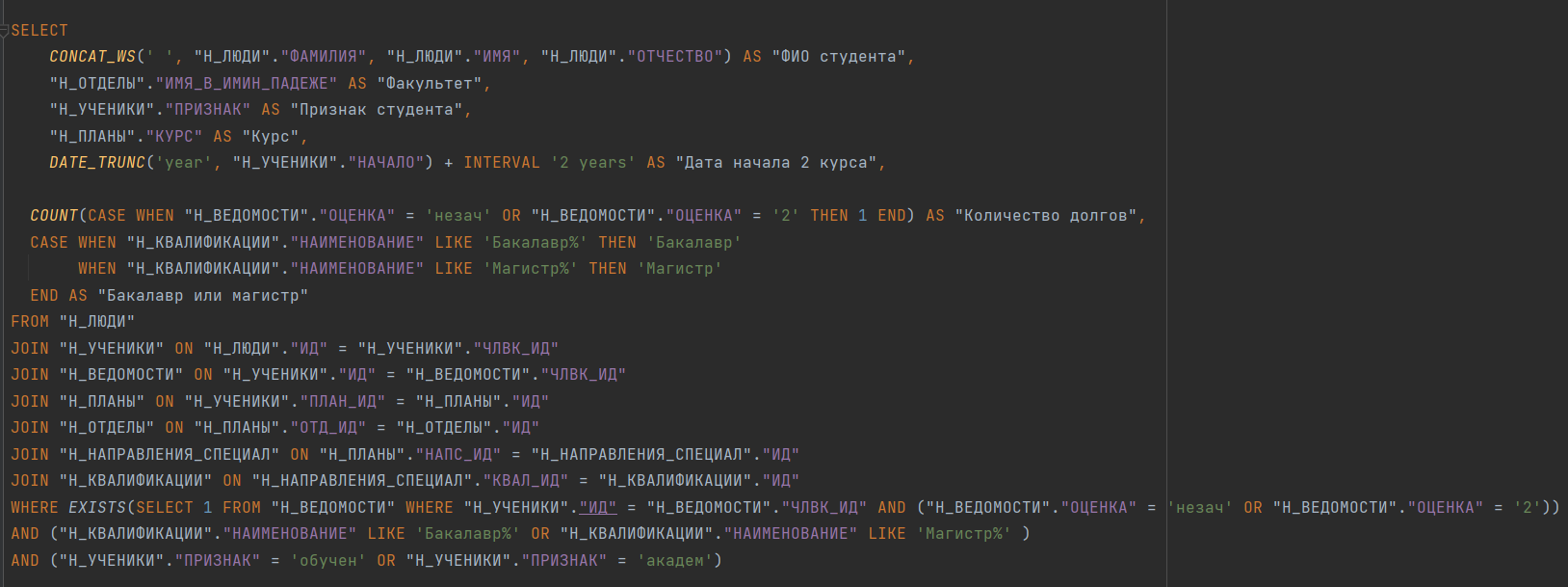
Также можно увидеть стоимость плана выполнения (cost), количество строк, которые были ожидаемо обработаны (rows), и фактическое время выполнения (actual time).

Из результатов также видно, что были использованы индексные сканы для таблиц "Н\_ЛЮДИ", "Н\_ВЕДОМОСТИ" и "Н\_СЕССИЯ", что может ускорить выполнение запроса.

В данном случае запрос был выполнен очень быстро (Execution Time: 0.058 ms), что свидетельствует о том, что он не нагружает базу данных.

**Дополнительные задания**

Задание 1: создать запрос для вывода ФИО человека, его факультет, бакалавриат или магистрант, курс, количество долгов, дата, когда он был на 2 курсе, а также его признак не должен быть равен «отчислен», после чего проанализировать его EXPLAIN ANALYZE.



Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

HashAggregate (cost=2241.03..2277.16 rows=1606 width=523) (actual time=18.605..18.627 rows=27 loops=1)

Group Key: concat\_ws(' '::text, «Н\_ЛЮДИ».«ФАМИЛИЯ», «Н\_ЛЮДИ».«ИМЯ», «Н\_ЛЮДИ».«ОТЧЕСТВО»), «Н\_ОТДЕЛЫ».«ИМЯ\_В\_ИМИН\_ПАДЕЖЕ», «Н\_УЧЕНИКИ».«ПРИЗНАК», «Н\_ПЛАНЫ».«КУРС», «Н\_УЧЕНИКИ».«НАЧАЛО», CASE WHEN ((«Н\_КВАЛИФИКАЦИИ».«НАИМЕНОВАНИЕ»)::text ~~ 'Бакалавр%'::text) THEN 'Бакалавр'::text WHEN ((«Н\_КВАЛИФИКАЦИИ».«НАИМЕНОВАНИЕ»)::text ~~ 'Магистр%'::text) THEN 'Магистр'::text ELSE NULL::text END

* Это операция агрегации по хэшу, используемая для группировки данных.
* "Group Key" указывает столбцы, по которым происходит группировка.
* В данном случае, группировка происходит по конкатенации значений столбцов "Н\_ЛЮДИ"."ФАМИЛИЯ", "Н\_ЛЮДИ"."ИМЯ", "Н\_ЛЮДИ"."ОТЧЕСТВО", "Н\_ОТДЕЛЫ"."ИМЯ\_В\_ИМИН\_ПАДЕЖЕ", "Н\_УЧЕНИКИ"."ПРИЗНАК", "Н\_ПЛАНЫ"."КУРС", "Н\_УЧЕНИКИ"."НАЧАЛО" и выражения CASE.

Batches: 1 Memory Usage: 89kB

* Это информация о выполнении операции.
* "Batches" указывает количество пакетов данных, использованных при выполнении операции.
* "Memory Usage" указывает количество используемой памяти.

-> Nested Loop (cost=2.71..2204.89 rows=1606 width=513) (actual time=4.203..17.281 rows=1924 loops=1)

* Это операция вложенного цикла.
* (cost=2.71..2204.89): Это оценка стоимости выполнения операции. Значение "2.71" представляет оценку стоимости выполнения операции в самом оптимистичном сценарии, а "2204.89 " - в наихудшем сценарии. Стоимость измеряется в условных единицах и используется оптимизатором запросов для выбора наиболее эффективного плана выполнения.
* (actual time=4.203..17.281): Это фактическое время выполнения операции. Значение "4.203" указывает на минимальное время выполнения, а "17.281 " - на максимальное время выполнения операции.

-> Nested Loop (cost=2.42..1679.47 rows=168 width=918) (actual time=4.189..14.928 rows=27 loops=1)

-> Nested Loop Semi Join (cost=2.13..1625.00 rows=168 width=873) (actual time=4.180..14.834 rows=27 loops=1)

* Это полусоединение вложенных циклов.
* Оно используется для объединения двух наборов данных, где только соответствующие строки из внешнего набора остаются в результирующем наборе данных.

-> Nested Loop (cost=1.84..219.08 rows=1196 width=869) (actual time=0.090..6.253 rows=6694 loops=1)

-> Nested Loop (cost=1.55..15.65 rows=41 width=844) (actual time=0.078..0.644 rows=333 loops=1)

-> Nested Loop (cost=1.40..13.91 rows=41 width=430) (actual time=0.070..0.439 rows=333 loops=1)

-> Hash Join (cost=1.25..6.26 rows=9 width=422) (actual time=0.054..0.122 rows=64 loops=1)

Hash Cond: («Н\_НАПРАВЛЕНИЯ\_СПЕЦИАЛ».«КВАЛ\_ИД» = "Н\_КВАЛИФИКАЦИИ».«ИД»)

* Это операция соединения по хэшу.
* Она используется для объединения двух наборов данных на основе значения хэша.
* В данном случае соединение происходит между таблицами "Н\_НАПРАВЛЕНИЯ\_СПЕЦИАЛ" и "Н\_КВАЛИФИКАЦИИ" по столбцам "КВАЛ\_ИД" и "ИД" соответственно.

-> Seq Scan on «Н\_НАПРАВЛЕНИЯ\_СПЕЦИАЛ» (cost=0.00..4.51 rows=151 width=8) (actual time=0.018..0.038 rows=151 loops=1)

* + Это операция последовательного сканирования таблицы.
  + Она просматривает все строки таблицы последовательно для выполнения операции.

-> Hash (cost=1.24..1.24 rows=1 width=422) (actual time=0.022..0.022 rows=9 loops=1)

Buckets: 1024 Batches: 1 Memory Usage: 9kB

* + Это операция хэширования.
  + Она используется для создания хэш-таблицы на основе входных данных.
  + В данном случае создается хэш-таблица для выполнения операции соединения.

-> Seq Scan on «Н\_КВАЛИФИКАЦИИ» (cost=0.00..1.24 rows=1 width=422) (actual time=0.011..0.015 rows=9 loops=1)"

Filter: (((«НАИМЕНОВАНИЕ»)::text ~~ 'Бакалавр%'::text) OR ((«НАИМЕНОВАНИЕ»)::text ~~ 'Магистр%'::text))

Rows Removed by Filter: 7

* Это операция последовательного сканирования таблицы.
* Она просматривает все строки таблицы последовательно для выполнения операции.

-> Index Scan using «ПЛАН\_НАПС\_FK\_I» on «Н\_ПЛАНЫ» (cost=0.15..0.74 rows=11 width=16) (actual time=0.001..0.004 rows=5 loops=64)

Index Cond: («НАПС\_ИД» = «Н\_НАПРАВЛЕНИЯ\_СПЕЦИАЛ».«ИД»)

* Это операция сканирования индекса.
* Она используется для поиска данных с использованием индекса.
* В данном случае используется индекс "ПЛАН\_НАПС\_FK\_I" на таблице "Н\_ПЛАНЫ".

-> Memoize (cost=0.15..0.20 rows=1 width=422) (actual time=0.000..0.000 rows=1 loops=333)

Cache Key: «Н\_ПЛАНЫ».«ОТД\_ИД»

Cache Mode: logical

Hits: 331 Misses: 2 Evictions: 0 Overflows: 0 Memory Usage: 1kB

* Это операция кэширования.
* Она используется для сохранения промежуточных результатов в памяти для повторного использования.
* В данном случае кэшируются данные из таблицы "Н\_ОТДЕЛЫ".

-> Index Scan using «ОТД\_PK» on «Н\_ОТДЕЛЫ» (cost=0.14..0.19 rows=1 width=422) (actual time=0.003..0.003 rows=1 loops=2)

Index Cond: («ИД» = «Н\_ПЛАНЫ».«ОТД\_ИД»)

-> Index Scan using «УЧЕН\_ПЛАН\_FK\_I» on «Н\_УЧЕНИКИ» (cost=0.29..4.62 rows=34 width=33) (actual time=0.002..0.013 rows=20 loops=333)

Index Cond: («ПЛАН\_ИД» = «Н\_ПЛАНЫ».«ИД»)

Filter: (((«ПРИЗНАК»)::text = 'обучен'::text) OR ((«ПРИЗНАК»)::text = 'академ'::text))

Rows Removed by Filter: 5

-> Index Scan using «ВЕД\_ЧЛВК\_FK\_IFK» on «Н\_ВЕДОМОСТИ» «Н\_ВЕДОМОСТИ\_1» (cost=0.29..2.72 rows=3 width=4) (actual time=0.001..0.001 rows=0 loops=6694)

Index Cond: («ЧЛВК\_ИД» = «Н\_УЧЕНИКИ».«ИД»)

Filter: (((«ОЦЕНКА»)::text = 'незач'::text) OR ((«ОЦЕНКА»)::text = '2'::text))

Rows Removed by Filter: 0

-> Index Scan using «ЧЛВК\_PK» on «Н\_ЛЮДИ» (cost=0.28..0.32 rows=1 width=53) (actual time=0.003..0.003 rows=1 loops=27)

Index Cond: («ИД» = «Н\_УЧЕНИКИ».«ЧЛВК\_ИД»)

-> Index Scan using «ВЕД\_ЧЛВК\_FK\_IFK» on «Н\_ВЕДОМОСТИ» (cost=0.29..2.38 rows=68 width=10) (actual time=0.002..0.045 rows=71 loops=27)

Index Cond: («ЧЛВК\_ИД» = «Н\_УЧЕНИКИ».«ИД»)

Planning Time: 4.313 ms

Execution Time: 18.783 ms

Задание 2: составить запрос, написать его план выполнения, после чего сверить с EXPLAIN ANALYZE.

Запрос:

EXPLAIN ANALYSE SELECT "Н\_ЛЮДИ"."ФАМИЛИЯ", "Н\_ВЕДОМОСТИ"."ДАТА"  
FROM "Н\_ЛЮДИ"  
RIGHT JOIN "Н\_ВЕДОМОСТИ" ON "Н\_ЛЮДИ"."ИД" = "Н\_ВЕДОМОСТИ"."ЧЛВК\_ИД"  
WHERE "Н\_ЛЮДИ"."ИД" > 100012

План выполнения:

* Seq Scan "Н\_ЛЮДИ": производится полное сканирование таблицы для получения всех строк.
* Seq Scan "Н\_ВЕДОМОСТИ": производится полное сканирование таблицы "Н\_ВЕДОМОСТИ" для получения всех строк.
* Nested Loop: для каждой строки из первого сканирования (таблицы "Н\_ЛЮДИ") производится поиск соответствующей строки во втором сканировании (таблице "Н\_ВЕДОМОСТИ") по условию "Н\_ЛЮДИ"."ИД" = "Н\_ВЕДОМОСТИ"."ЧЛВК\_ИД". Если соответствующие строки найдены, то выбираются столбцы "Н\_ЛЮДИ"."ФАМИЛИЯ" и "Н\_ВЕДОМОСТИ"."ДАТА".
* Filter: После объединения результатов производится фильтрация по условию "Н\_ЛЮДИ"."ИД" > 100012.

EXPLAIN ANALYZE:

Nested Loop (cost=0.58..216.65 rows=43 width=28) (actual time=0.004..0.005 rows=0 loops=1)

-> Index Scan using "ЧЛВК\_PK" on "Н\_ЛЮДИ" (cost=0.28..8.29 rows=1 width=24) (actual time=0.004..0.004 rows=0 loops=1)"

Index Cond: ("ИД" < 100012)"

-> Index Scan using "ВЕД\_ЧЛВК\_FK\_IFK" on "Н\_ВЕДОМОСТИ" (cost=0.29..207.67 rows=68 width=12) (never executed)"

Index Cond: ("ЧЛВК\_ИД" = "Н\_ЛЮДИ"."ИД")

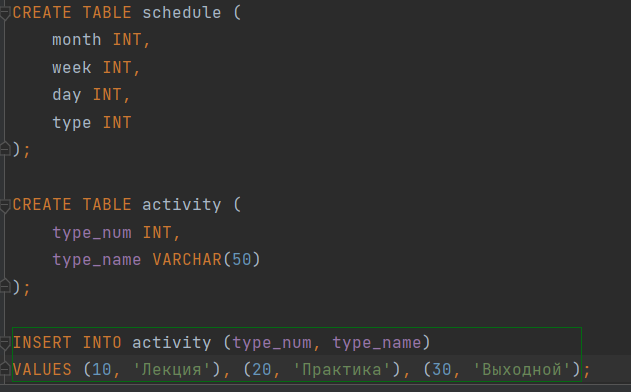
Planning Time: 0.315 ms

Execution Time: 0.037 ms

Этот запрос выполняет вложенный цикл со следующими шагами:

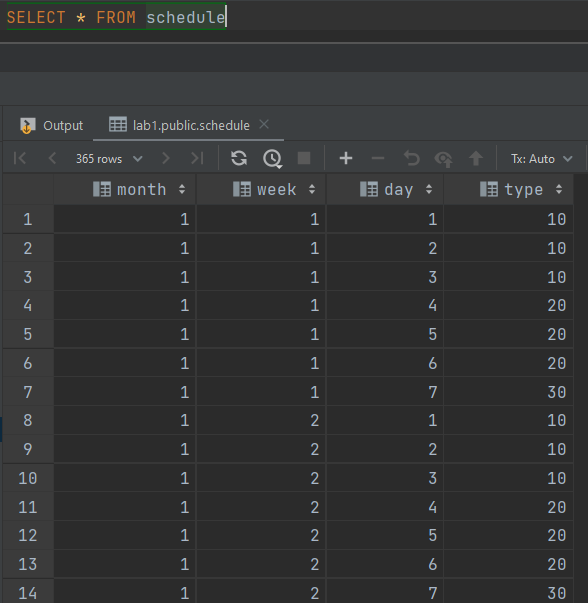
* Сначала выполняется сканирование индекса "ЧЛВК\_PK" на таблице "Н\_ЛЮДИ". Условие для сканирования индекса - значение столбца "ИД" < 100012.
* Затем выполняется сканирование индекса "ВЕД\_ЧЛВК\_FK\_IFK" на таблице "Н\_ВЕДОМОСТИ", но этот шаг фактически не выполняется (never executed), поскольку предыдущий шаг не вернул ни одной строки. Это связано с условием соединения двух таблиц, которое требует совпадения значений столбца "ЧЛВК\_ИД" из таблицы "Н\_ВЕДОМОСТИ" со значением столбца "ИД" из таблицы "Н\_ЛЮДИ".

Задание 3: создать таблицу на год, используя for, с месяцами, номерами недель и днями в неделе, а также с типом занятия, где 1-3 день - лекции, 4-6 - практики, 7 – выходной.



Изображение выглядит как текст, снимок экрана

Автоматически созданное описание

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Задание 4: создать представление, которое будет находить таблицу с наибольшим количество атрибутов и выводить их.

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Автоматически созданное описание

Задание 5: создать процедуру, куда передаётся имя таблицы и эта таблица создается с полями id и name, создать процедуру, куда передается имя таблицы, id и name и создается запись, а также создать функцию, куда передаётся имя таблицы и выводятся все записи.

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Изображение выглядит как текст, снимок экрана, программное обеспечение

Автоматически созданное описание

Изображение выглядит как текст, снимок экрана, Мультимедийное программное обеспечение, программное обеспечение

Автоматически созданное описание

**Вывод**

При выполнении данной лабораторной работы я изучила различные виды индексов и узнала, как использовать их для оптимизации скорости выполнения запросов.