МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ

Федеральное государственное автономное
образовательное учреждение высшего образования
«Санкт-Петербургский национальный исследовательский университет
информационных технологий, механики и оптики»

ФАКУЛЬТЕТ ПРОГРАММНОЙ ИНЖЕНЕРИИ И КОМПЬЮТЕРНОЙ ТЕХНИКИ

ЛАБОРАТОРНАЯ РАБОТА № 4

по дисциплине

‘Базы данных’

Вариант №13

Выполнил:
Студент группы P3132
Юксель Хамза

Преподаватель:
Егошин Алексей
Васильевич

УНИВЕРСИТЕТ ИТМО

Санкт-Петербург, 2025

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.ЧЛВК_ИД.
   Фильтры (AND):
   a) Н_ТИПЫ_ВЕДОМОСТЕЙ.НАИМЕНОВАНИЕ < Экзаменационный лист.
   b) Н_ВЕДОМОСТИ.ДАТА < 1998-01-05.
   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.ЧЛВК_ИД, Н_УЧЕНИКИ.НАЧАЛО.
   Фильтры: (AND)
   a) Н_ЛЮДИ.ОТЧЕСТВО < Сергеевич.
   b) Н_ОБУЧЕНИЯ.НЗК > 999080.
   c) Н_УЧЕНИКИ.ГРУППА = 3100.
   Вид соединения: RIGHT JOIN.

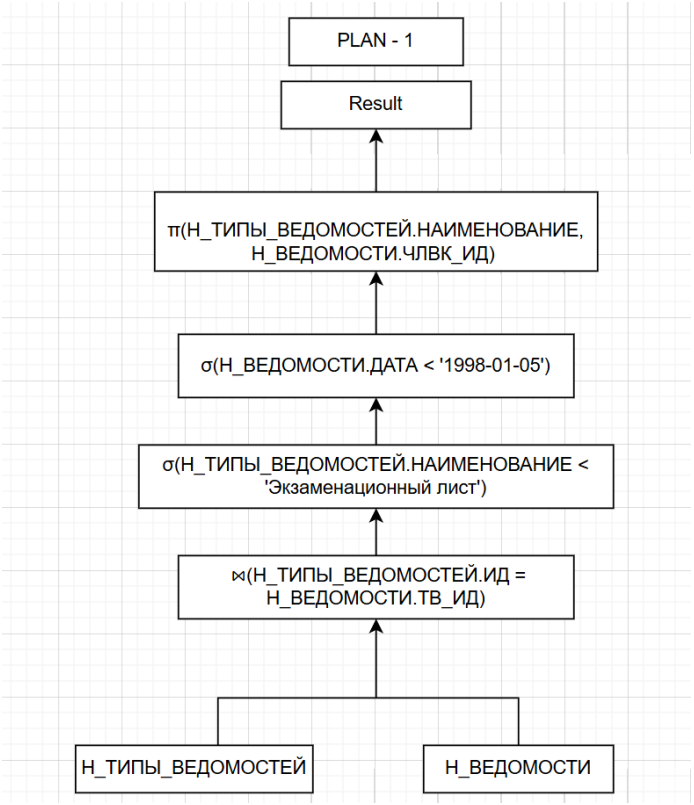
Запрос №1

```
-- 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.ЧЛВК_ИД.
   -- Фильтры (AND):
   -- a) Н_ТИПЫ_ВЕДОМОСТЕЙ.НАИМЕНОВАНИЕ < Экзаменационный лист.
   -- b) Н_ВЕДОМОСТИ.ДАТА < 1998-01-05
   -- Вид соединения: INNER JOIN.

   SELECT
      tv.НАИМЕНОВАНИЕ,
      v.ЧЛВК_ИД
   FROM
      Н_ТИПЫ_ВЕДОМОСТЕЙ tv
   INNER JOIN
           Н_ВЕДОМОСТИ v ON tv.ИД = v.ТВ_ИД
   WHERE tv.НАИМЕНОВАНИЕ < 'Экзаменационный лист'
      AND v.ДАТА < '1998-01-05';
```

```
ucheb=> SELECT
    TV.НАИМЕНОВАНИЕ,
    V.ЧЛВК_ИД
FROM
    Н_ТИПЫ_ВЕДОМОСТЕЙ TV
INNER JOIN
    Н_ВЕДОМОСТИ V ON TV.ИД = V.ТВ_ИД
WHERE
    TV.НАИМЕНОВАНИЕ < 'Экзаменационный лист'
    AND V.ДАТА < '1998-01-05';
 НАИМЕНОВАНИЕ | ЧЛВК_ИД
--------------+---------
(0 строк)
```

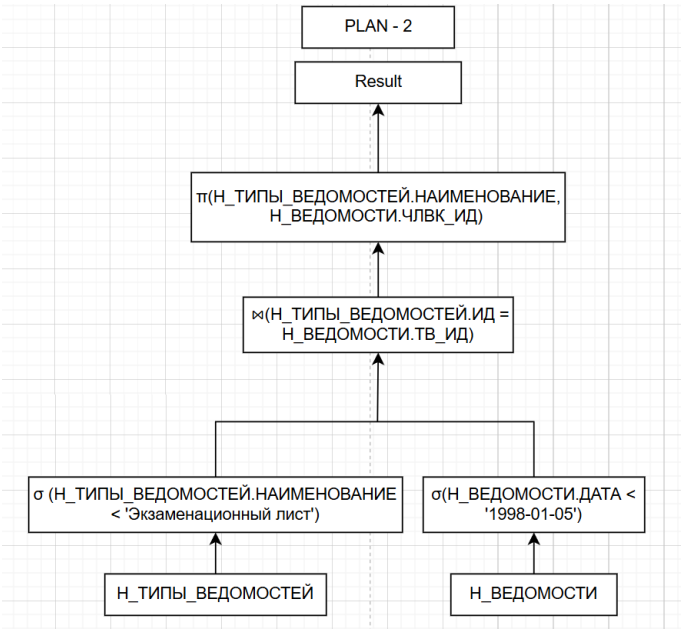
Планы выполнения

PLAN-1 for Query-1



Сначала таблицы Н_ТИПЫ_ВЕДОМОСТЕЙ и Н_ВЕДОМОСТИ полностью соединяются по ключам ИД и ТВ_ИД. Затем к большому промежуточному результату применяются фильтры по НАИМЕНОВАНИЕ и ДАТА. Этот план неэффективен из-за большого объема данных на этапе соединения.

PLAN-2 for Query-1



Сначала происходит фильтрация таблицы Н_ТИПЫ_ВЕДОМОСТЕЙ по условию НАИМЕНОВАНИЕ < 'Экзаменационный лист'. Параллельно фильтруется таблица Н_ВЕДОМОСТИ по условию ДАТА < '1998-01-05'. Затем отфильтрованные (значительно меньшие) наборы данных соединяются.

Оптимальный план (без новых индексов):

Оптимальным является **План №2**. Фильтрация данных до их соединения (pushing predicates down) уменьшает количество строк, участвующих в операции соединения. Это снижает нагрузку на систему, уменьшает объем промежуточных данных и ускоряет выполнение запроса.

Индексы

1. **Для таблицы Н_ТИПЫ_ВЕДОМОСТЕЙ:**
   a. **По столбцу НАИМЕНОВАНИЕ:**

   CREATE INDEX IDX_TB_НАИМЕНОВАНИЕ **ON** Н_ТИПЫ_ВЕДОМОСТЕЙ **USING** BTREE (НАИМЕНОВАНИЕ);

      i. **Тип индекса**: BTREE.
      ii. **Обоснование**: Этот индекс необходим для эффективной фильтрации по условию TV.НАИМЕНОВАНИЕ < 'Экзаменационный лист'. B-tree индексы хорошо подходят для операций сравнения по диапазону (<, >, <=, >=).

   b. **По столбцу ИД**: (Первичный ключ TB_PK уже существует и является B-tree индексом)
      i. **Обоснование**: Используется в условии JOIN.

2. **Для таблицы Н_ВЕДОМОСТИ:**
   a. **По столбцу ДАТА**: (Индекс ВЕД_ДАТА_I уже существует)

   -- **CREATE** INDEX IDX_ВЕД_ДАТА **ON** Н_ВЕДОМОСТИ USING **BTREE** (ДАТА); -- Уже существует как ВЕД_ДАТА_I

      i. **Тип индекса:** BTREE.
      ii. **Обоснование:** Необходим для быстрой фильтрации по условию V.ДАТА < '1998-01-05'.

   b. **По столбцу ТВ_ИД:** (Индекс ВЕД_ТВ_FK_I уже существует)
   -- CREATE INDEX IDX_ВЕД_ТВ_ИД ON Н_ВЕДОМОСТИ USING BTREE (ТВ_ИД); -- Уже существует как ВЕД_ТВ_FK_I
      i. **Тип индекса:** BTREE.

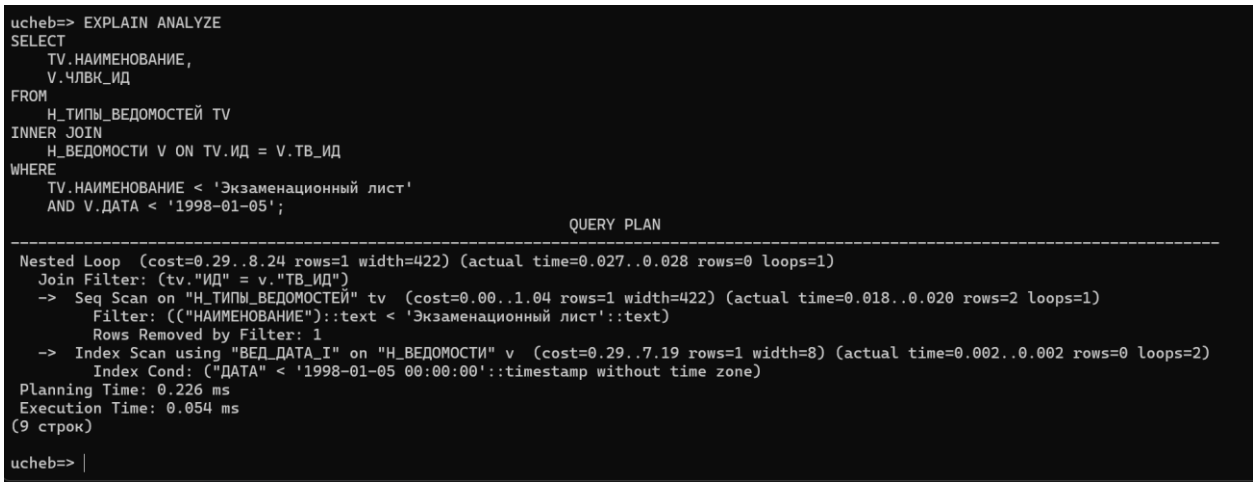
ii. **Обоснование:** Используется в условии JOIN для связи с таблицей Н_ТИПЫ_ВЕДОМОСТЕЙ.

Изменение планов при добавлении индекса и как:
При добавлении/использовании указанных индексов, План №2 будет выполняться значительно эффективнее:

- Вместо полного сканирования (Sequential Scan) таблицы Н_ТИПЫ_ВЕДОМОСТЕЙ для условия TV.НАИМЕНОВАНИЕ < 'Экзаменационный лист' будет использоваться **Index Scan** по индексу IDX_TB_НАИМЕНОВАНИЕ.
- Для условия V.ДАТА < '1998-01-05' в таблице Н_ВЕДОМОСТИ будет использоваться **Index Scan** по существующему индексу ВЕД_ДАТА_I.
- Операция INNER JOIN сможет эффективно использовать существующие индексы TB_PK на Н_ТИПЫ_ВЕДОМОСТЕЙ.ИД и ВЕД_TB_FK_I на Н_ВЕДОМОСТИ.ТВ_ИД. Это может привести к выбору более быстрого алгоритма соединения, такого как **Index Nested Loop Join** (если одна из отфильтрованных таблиц мала) или **Merge Join** (если результаты из Index Scan поступают в отсортированном виде).

В результате, СУБД будет напрямую обращаться к нужным данным через индексы, минуя чтение и обработку большей части таблиц, что кардинально сократит время выполнения запроса.

EXPLAIN ANALYZE



```
ucheb=> EXPLAIN ANALYZE
SELECT
    TV.НАИМЕНОВАНИЕ,
    V.ЧЛВК_ИД
FROM
    Н_ТИПЫ_ВЕДОМОСТЕЙ TV
INNER JOIN
    Н_ВЕДОМОСТИ V ON TV.ИД = V.ТВ_ИД
WHERE
    TV.НАИМЕНОВАНИЕ < 'Экзаменационный лист'
    AND V.ДАТА < '1998-01-05';
                                                                  QUERY PLAN
-----------------------------------------------------------------------------------------------------------------------------------------------------------
 Nested Loop  (cost=0.29..8.24 rows=1 width=422) (actual time=0.027..0.028 rows=0 loops=1)
   Join Filter: (tv."ИД" = v."ТВ_ИД")
   ->  Seq Scan on "Н_ТИПЫ_ВЕДОМОСТЕЙ" tv  (cost=0.00..1.04 rows=1 width=422) (actual time=0.018..0.020 rows=2 loops=1)
         Filter: (("НАИМЕНОВАНИЕ")::text < 'Экзаменационный лист'::text)
         Rows Removed by Filter: 1
   ->  Index Scan using "ВЕД_ДАТА_I" on "Н_ВЕДОМОСТИ" v  (cost=0.29..7.19 rows=1 width=8) (actual time=0.002..0.002 rows=0 loops=2)
         Index Cond: ("ДАТА" < '1998-01-05 00:00:00'::timestamp without time zone)
 Planning Time: 0.226 ms
 Execution Time: 0.054 ms
(9 строк)

ucheb=>
```

QUERY PLAN

```
 Nested Loop  (cost=0.29..8.24 rows=1 width=422) (actual time=0.027..0.028 rows=0 loops=1)
   Join Filter: (tv."ИД" = v."ТВ_ИД")
   ->  Seq Scan on "Н_ТИПЫ_ВЕДОМОСТЕЙ" tv  (cost=0.00..1.04 rows=1 width=422) (actual
time=0.018..0.020 rows=2 loops=1)
         Filter: (("НАИМЕНОВАНИЕ")::text < 'Экзаменационный лист'::text)
         Rows Removed by Filter: 1
   ->  Index Scan using "ВЕД_ДАТА_I" on "Н_ВЕДОМОСТИ" v  (cost=0.29..7.19 rows=1
width=8) (actual time=0.002..0.002 rows=0 loops=2)
         Index Cond: ("ДАТА" < '1998-01-05 00:00:00'::timestamp without time zone)
 Planning Time: 0.226 ms
 Execution Time: 0.054 ms
(9 строк)
```

Запрос №2

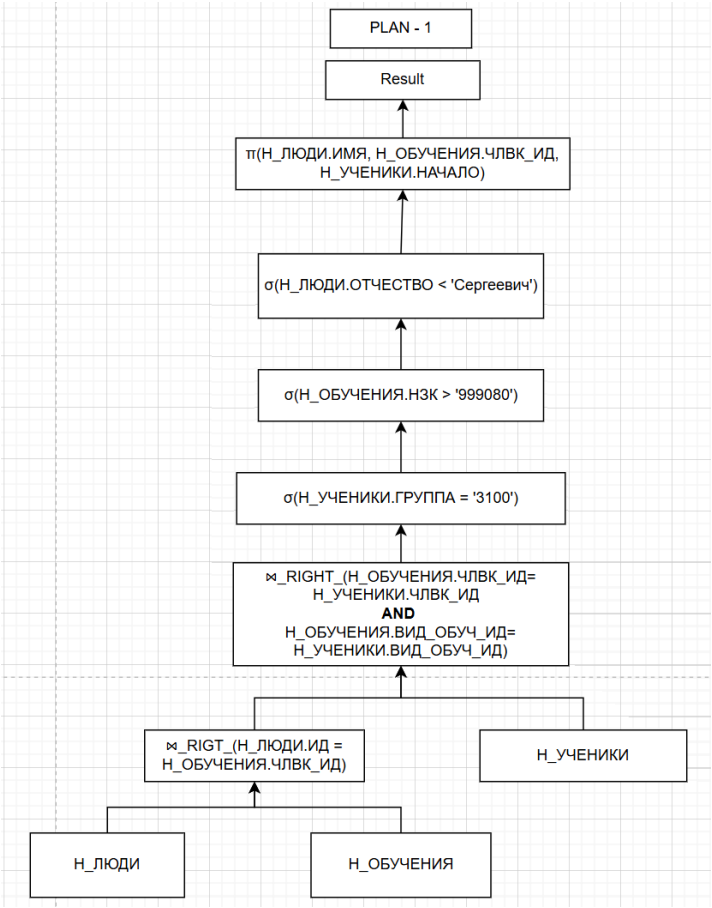
```
-- 2. Сделать запрос для получения атрибутов из указанных таблиц, применив
фильтры по указанным условиям:
-- Таблицы: Н_ЛЮДИ, Н_ОБУЧЕНИЯ, Н_УЧЕНИКИ.
-- Вывести атрибуты: Н_ЛЮДИ.ИМЯ, Н_ОБУЧЕНИЯ.ЧЛВК_ИД, Н_УЧЕНИКИ.НАЧАЛО.
-- Фильтры: (AND)
-- a) Н_ЛЮДИ.ОТЧЕСТВО < Сергеевич.
-- b) Н_ОБУЧЕНИЯ.НЗК > 999080.
-- c) Н_УЧЕНИКИ.ГРУППА = 3100.
-- Вид соединения: RIGHT JOIN.

SELECT
  L.ИМЯ,
  O.ЧЛВК_ИД,
  U.НАЧАЛО
FROM
  Н_ЛЮДИ L
RIGHT JOIN
  Н_ОБУЧЕНИЯ O ON L.ИД = O.ЧЛВК_ИД
RIGHT JOIN
  Н_УЧЕНИКИ U ON O.ЧЛВК_ИД = U.ЧЛВК_ИД AND O.ВИД_ОБУЧ_ИД = U.ВИД_ОБУЧ_ИД
WHERE
  L.ОТЧЕСТВО < 'Сергеевич'
  AND O.НЗК > '999080'
  AND U.ГРУППА = '3100';
```

```
ucheb=>     SELECT
L.ИМЯ,
O.ЧЛВК_ИД,
U.НАЧАЛО
    FROM
H_ЛЮДИ L
    RIGHT JOIN
H_ОБУЧЕНИЯ O ON L.ИД = O.ЧЛВК_ИД
    RIGHT JOIN
H_УЧЕНИКИ U ON O.ЧЛВК_ИД = U.ЧЛВК_ИД AND O.ВИД_ОБУЧ_ИД = U.ВИД_ОБУЧ_ИД
    WHERE
L.ОТЧЕСТВО < 'Сергеевич'
AND O.НЗК > '999080'
AND U.ГРУППА = '3100';
 ИМЯ | ЧЛВК_ИД | НАЧАЛО
-----+---------+--------
(0 строк)
```

Планы выполнения

PLAN-1 for Query-2

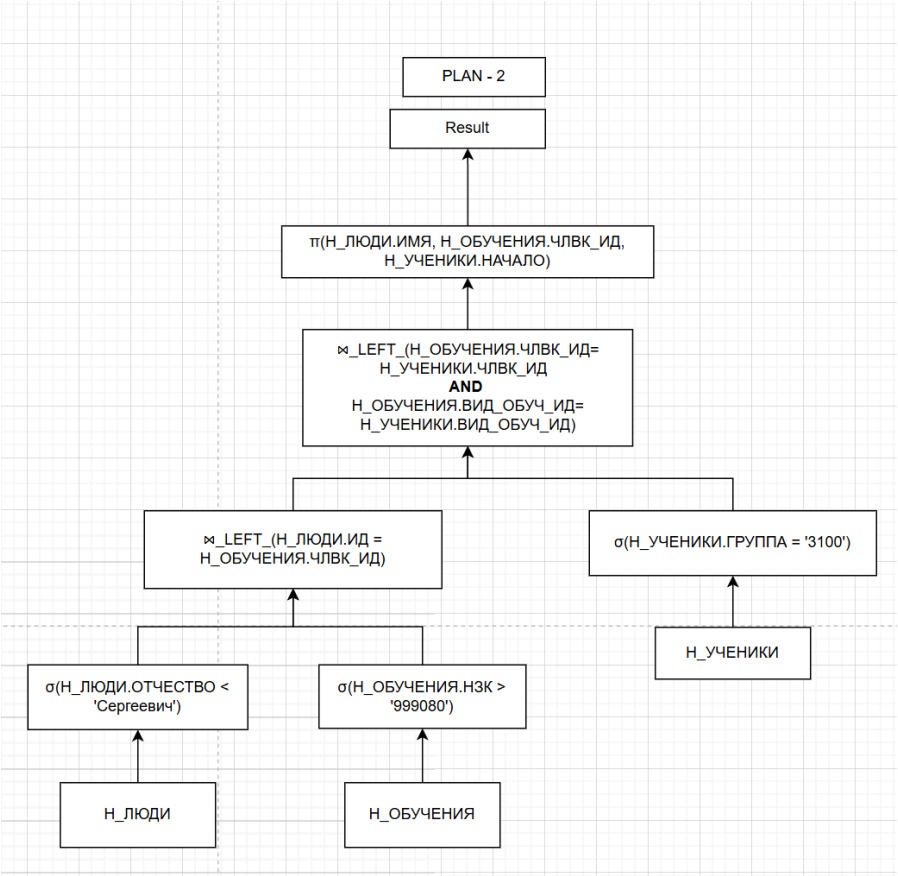
В этом плане сначала выполняются все соединения в указанном порядке, а затем к большому промежуточному результату применяются фильтры.



Сначала Н_ЛЮДИ соединяется с Н_ОБУЧЕНИЯ через RIGHT JOIN. Затем результат этого соединения соединяется с Н_УЧЕНИКИ через RIGHT JOIN. И только после всех соединений применяются три фильтра. Это неэффективно из-за обработки больших объемов данных на этапах соединений.

PLAN-2 for Query-2

В этом плане фильтры применяются к таблицам как можно раньше. Планировщик, скорее всего, преобразует RIGHT JOIN в LEFT JOIN или INNER JOIN из-за условий WHERE.



Каждая таблица сначала фильтруется по соответствующим условиям WHERE. Затем отфильтрованные Н_ЛЮДИ и Н_ОБУЧЕНИЯ соединяются (вероятно, как LEFT JOIN или INNER JOIN). Затем отфильтрованные Н_УЧЕНИКИ соединяются с результатом предыдущего JOIN (также, вероятно, как LEFT JOIN или INNER JOIN из-за семантики WHERE).

Оптимальный план (без новых индексов):

Оптимальным является **План №2**. Применение фильтров к таблицам до их соединения значительно уменьшает количество строк, участвующих в последующих операциях соединения. Это снижает вычислительную нагрузку и использование памяти, что приводит к более быстрому выполнению запроса.

Индексы

1. **Для таблицы Н_ЛЮДИ:**
    a. **По столбцу ОТЧЕСТВО:**

    CREATE INDEX IDX_ЛЮДИ_ОТЧЕСТВО **ON** Н_ЛЮДИ **USING** BTREE (ОТЧЕСТВО);

        i. **Тип индекса**: BTREE.
        ii. **Обоснование**: Для ускорения фильтрации L.ОТЧЕСТВО < 'Сергеевич'

    b. **По столбцу ИД**: (Первичный ключ ЧЛВК_PK уже существует).
        i. **Обоснование:** Используется в JOIN с Н_ОБУЧЕНИЯ

2. **Для таблицы Н_ОБУЧЕНИЯ:**
    a. **По столбцу НЗК**:
    CREATE INDEX IDX_ОБУЧЕНИЯ_НЗК **ON** Н_ОБУЧЕНИЯ **USING** BTREE (НЗК);
        i. **Тип индекса:** BTREE.
        ii. **Обоснование:** Для ускорения фильтрации О.НЗК > '999080'**.**

    b. **По столбцам ЧЛВК_ИД, ВИД_ОБУЧ_ИД:** (Первичный ключ ОБУЧ_PK ("ВИД_ОБУЧ_ИД", "ЧЛВК_ИД") и другие релевантные индексы, как ОБУЧ_ЧЛВК_FK_I, уже существуют).
        i. **Обоснование:** Используется в JOIN с Н_УЧЕНИКИ и Н_ЛЮДИ.

3. **Для таблицы Н_УЧЕНИКИ:**
    a. **По столбцу ГРУППА:**

    CREATE INDEX IDX_УЧЕНИКИ_ГРУППА **ON** Н_УЧЕНИКИ **USING** BTREE (ГРУППА);
        i. **Тип индекса**: BTREE.
        ii. **Обоснование**: Для ускорения фильтрации U.ГРУППА = '3100'.

    b. **По столбцам ЧЛВК_ИД, ВИД_ОБУЧ_ИД**: (Индекс УЧЕН_ОБУЧ_FK_I ("ЧЛВК_ИД", "ВИД_ОБУЧ_ИД") уже существует).
        i. **Обоснование**: Используется в JOIN с Н_ОБУЧЕНИЯ.

Изменение планов при добавлении индекса и как:
Эффекты от добавления индексов будут аналогичны предыдущему описанию:

- Фильтрация по ОТЧЕСТВО, НЗК и ГРУППА будет использовать **Index Scan** по соответствующим новым индексам.

- Операции JOIN смогут более эффективно использовать существующие и вновь созданные индексы на ключах соединения. Планировщик, вероятно, все равно преобразует RIGHT JOIN в более эффективные INNER JOIN или LEFT JOIN с учетом фильтров WHERE и будет выбирать оптимальные алгоритмы соединения (например, **Index Nested Loop Join** или **Merge Join**).

Общий эффект — значительное сокращение времени выполнения.

EXPLAIN ANALYZE

```
ucheb=> EXPLAIN ANALYZE
SELECT
    L.ИМЯ,
    O.ЧЛВК_ИД,
    U.НАЧАЛО
FROM
    Н_ЛЮДИ L
RIGHT JOIN
    Н_ОБУЧЕНИЯ O ON L.ИД = O.ЧЛВК_ИД
RIGHT JOIN
    Н_УЧЕНИКИ U ON O.ЧЛВК_ИД = U.ЧЛВК_ИД AND O.ВИД_ОБУЧ_ИД = U.ВИД_ОБУЧ_ИД
WHERE
    L.ОТЧЕСТВО < 'Сергеевич'
    AND O.НЗК > '999080'
    AND U.ГРУППА = '3100';
                                                        QUERY PLAN
-----------------------------------------------------------------------------------------------------------------------------
 Nested Loop  (cost=0.57..129.04 rows=1 width=25) (actual time=1.238..1.238 rows=0 loops=1)
   Join Filter: (u."ВИД_ОБУЧ_ИД" = o."ВИД_ОБУЧ_ИД")
   ->  Nested Loop  (cost=0.28..128.07 rows=1 width=25) (actual time=1.238..1.238 rows=0 loops=1)
         ->  Seq Scan on "Н_ОБУЧЕНИЯ" o  (cost=0.00..119.76 rows=1 width=8) (actual time=1.237..1.237 rows=0 loops=1)
               Filter: (("НЗК")::text > '999080'::text)
               Rows Removed by Filter: 5021
         ->  Index Scan using "ЧЛВК_РК" on "Н_ЛЮДИ" l  (cost=0.28..8.30 rows=1 width=17) (never executed)
               Index Cond: ("ИД" = o."ЧЛВК_ИД")
               Filter: (("ОТЧЕСТВО")::text < 'Сергеевич'::text)
   ->  Index Scan using "УЧЕН_ОБУЧ_FK_I" on "Н_УЧЕНИКИ" u  (cost=0.29..0.96 rows=1 width=16) (never executed)
         Index Cond: ("ЧЛВК_ИД" = l."ИД")
         Filter: (("ГРУППА")::text = '3100'::text)
 Planning Time: 0.833 ms
 Execution Time: 1.277 ms
(14 строк)

ucheb=>
```

QUERY PLAN

Nested Loop (cost=0.57..129.04 rows=1 width=25) (actual time=1.238..1.238 rows=0 loops=1)
  Join Filter: (u."ВИД_ОБУЧ_ИД" = o."ВИД_ОБУЧ_ИД")
  -> Nested Loop (cost=0.28..128.07 rows=1 width=25) (actual time=1.238..1.238 rows=0 loops=1)
      -> Seq Scan on "Н_ОБУЧЕНИЯ" o (cost=0.00..119.76 rows=1 width=8) (actual time=1.237..1.237 rows=0 loops=1)
          Filter: (("НЗК")::text > '999080'::text)
          Rows Removed by Filter: 5021
      -> Index Scan using "ЧЛВК_РК" on "Н_ЛЮДИ" l (cost=0.28..8.30 rows=1 width=17) (never executed)
          Index Cond: ("ИД" = o."ЧЛВК_ИД")
          Filter: (("ОТЧЕСТВО")::text < 'Сергеевич'::text)

```
  ->  Index Scan using "УЧЕН_ОБУЧ_FK_I" on "Н_УЧЕНИКИ" u  (cost=0.29..0.96 rows=1
width=16) (never executed)
        Index Cond: ("ЧЛВК_ИД" = l."ИД")
        Filter: (("ГРУППА")::text = '3100'::text)
 Planning Time: 0.833 ms
 Execution Time: 1.277 ms
(14 строк)
```

Вывод

При выполнении лабораторной работы я познакомился с использованием индексов для ускорения обработки запросов в SQL, а также планами выполнения запросов, их построением и анализом.