федеральное государственное автономное образовательное учреждение высшего образования
«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»

ОТЧЕТ

по лабораторной работе №4
Вариант №1320

по дисциплине «**Информационные системы и базы данных**»

Автор: Кулаков Н. В.

Факультет: ПИиКТ

Группа: P33312

Преподаватель: Шешуков Д.М.

УНИВЕРСИТЕТ ИТМО

2022

1. **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:
   H_ТИПЫ_ВЕДОМОСТЕЙ, H_ВЕДОМОСТИ.
   Вывести атрибуты: H_ТИПЫ_ВЕДОМОСТЕЙ.НАИМЕНОВАНИЕ, H_ВЕДОМОСТИ.ЧЛВК_ИД.
   Фильтры (AND):
   a) H_ТИПЫ_ВЕДОМОСТЕЙ.НАИМЕНОВАНИЕ > Ведомость.
   b) H_ВЕДОМОСТИ.ИД > 1250981.
   Вид соединения: INNER JOIN.
2. Сделать запрос для получения атрибутов из указанных таблиц, применив фильтры по указанным условиям:
   Таблицы: H_ЛЮДИ, H_ОБУЧЕНИЯ, H_УЧЕНИКИ.
   Вывести атрибуты: H_ЛЮДИ.ИД, H_ОБУЧЕНИЯ.НЗК, H_УЧЕНИКИ.НАЧАЛО.
   Фильтры: (AND)
   a) H_ЛЮДИ.ОТЧЕСТВО > Владимирович.
   b) H_ОБУЧЕНИЯ.ЧЛВК_ИД > 163276.
   c) H_УЧЕНИКИ.ГРУППА < 4103.
   Вид соединения: INNER JOIN.

2. **Выполнение.**

Запрос №1:

- Запрос SQL:

```
select _ТИПЫ_ВЕДОМОСТЕЙ.НАИМЕНОВАНИЕ, _ВЕДОМОСТИ.ЧЛВК_ИД
from (
    select *
    from Н_ТИПЫ_ВЕДОМОСТЕЙ
    where НАИМЕНОВАНИЕ > 'Ведомость'
  ) as _ТИПЫ_ВЕДОМОСТЕЙ
  inner join (
    select *
    from Н_ВЕДОМОСТИ
    where ИД > 1250981
  ) as _ВЕДОМОСТИ on _ВЕДОМОСТИ.ТВ_ИД = _ТИПЫ_ВЕДОМОСТЕЙ.ИД
;

select Н_ТИПЫ_ВЕДОМОСТЕЙ.НАИМЕНОВАНИЕ, Н_ВЕДОМОСТИ.ЧЛВК_ИД
from Н_ТИПЫ_ВЕДОМОСТЕЙ
  inner join Н_ВЕДОМОСТИ on Н_ВЕДОМОСТИ.ТВ_ИД = Н_ТИПЫ_ВЕДОМОСТЕЙ.ИД
where
  Н_ТИПЫ_ВЕДОМОСТЕЙ.НАИМЕНОВАНИЕ > 'Ведомость' and Н_ВЕДОМОСТИ.ИД > 1250981
;
```

- Вывод EXPAIN_ANALYZE: (https://explain.depesz.com/s/i3iZa#html)

```
                                                          QUERY PLAN
--------------------------------------------------------------------------------------------------------------------------------
Hash Join  (cost=491.19..5047.40 rows=8676 width=422) (actual time=1.384..10.251 rows=6789 loops=1)
  Hash Cond: ("Н_ВЕДОМОСТИ"."ТВ_ИД" = "Н_ТИПЫ_ВЕДОМОСТЕЙ"."ИД")
  ->  Bitmap Heap Scan on "Н_ВЕДОМОСТИ"  (cost=490.14..4881.51 rows=26029 width=8) (actual time=1.314..5.661 rows=26125 loops=1)
        Recheck Cond: ("ИД" > 1250981)
        Heap Blocks: exact=642
        ->  Bitmap Index Scan on "ВЕД_PK"  (cost=0.00..483.64 rows=26029 width=0) (actual time=1.228..1.229 rows=26125 loops=1)
              Index Cond: ("ИД" > 1250981)
  ->  Hash  (cost=1.04..1.04 rows=1 width=422) (actual time=0.023..0.024 rows=2 loops=1)
        Buckets: 1024  Batches: 1  Memory Usage: 9kB
        ->  Seq Scan on "Н_ТИПЫ_ВЕДОМОСТЕЙ"  (cost=0.00..1.04 rows=1 width=422) (actual time=0.015..0.017 rows=2 loops=1)
              Filter: (("НАИМЕНОВАНИЕ")::text > 'Ведомость'::text)
              Rows Removed by Filter: 1
Planning Time: 0.213 ms
Execution Time: 10.601 ms
```

```
                                                          QUERY PLAN
--------------------------------------------------------------------------------------------------------------------------------
Hash Join  (cost=491.19..5047.40 rows=8676 width=422) (actual time=1.386..10.267 rows=6789 loops=1)
  Hash Cond: ("Н_ВЕДОМОСТИ"."ТВ_ИД" = "Н_ТИПЫ_ВЕДОМОСТЕЙ"."ИД")
  ->  Bitmap Heap Scan on "Н_ВЕДОМОСТИ"  (cost=490.14..4881.51 rows=26029 width=8) (actual time=1.309..5.652 rows=26125 loops=1)
        Recheck Cond: ("ИД" > 1250981)
        Heap Blocks: exact=642
        ->  Bitmap Index Scan on "ВЕД_PK"  (cost=0.00..483.64 rows=26029 width=0) (actual time=1.224..1.224 rows=26125 loops=1)
              Index Cond: ("ИД" > 1250981)
  ->  Hash  (cost=1.04..1.04 rows=1 width=422) (actual time=0.030..0.030 rows=2 loops=1)
        Buckets: 1024  Batches: 1  Memory Usage: 9kB
        ->  Seq Scan on "Н_ТИПЫ_ВЕДОМОСТЕЙ"  (cost=0.00..1.04 rows=1 width=422) (actual time=0.019..0.021 rows=2 loops=1)
              Filter: (("НАИМЕНОВАНИЕ")::text > 'Ведомость'::text)
              Rows Removed by Filter: 1
Planning Time: 0.213 ms
Execution Time: 10.625 ms
(14 строк)
```

Как можно заметить планы для этих запросов исполнились одинаковые.

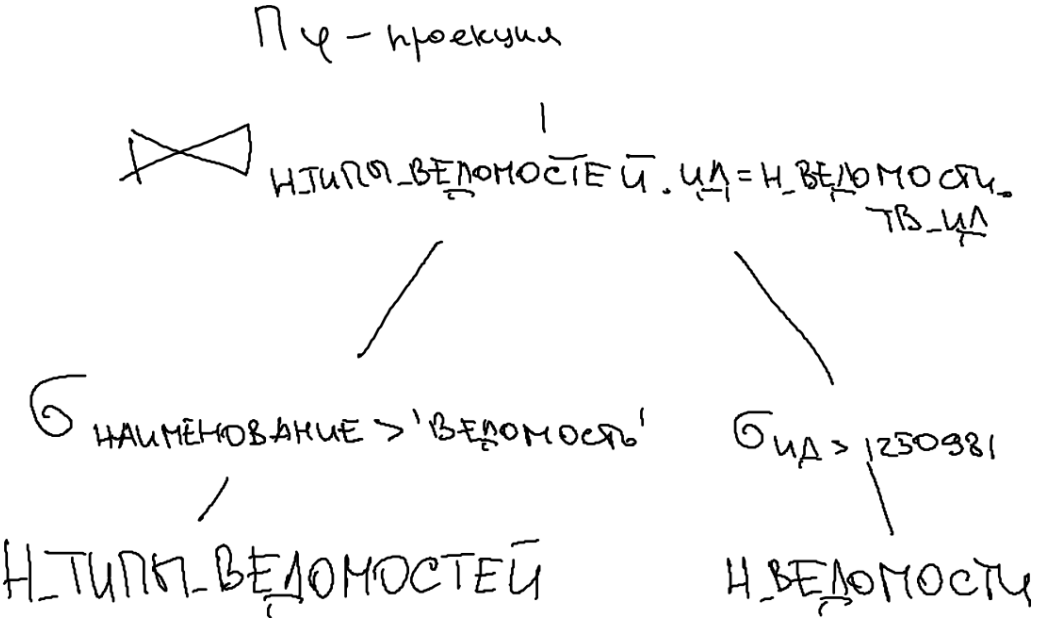
- Индексы:

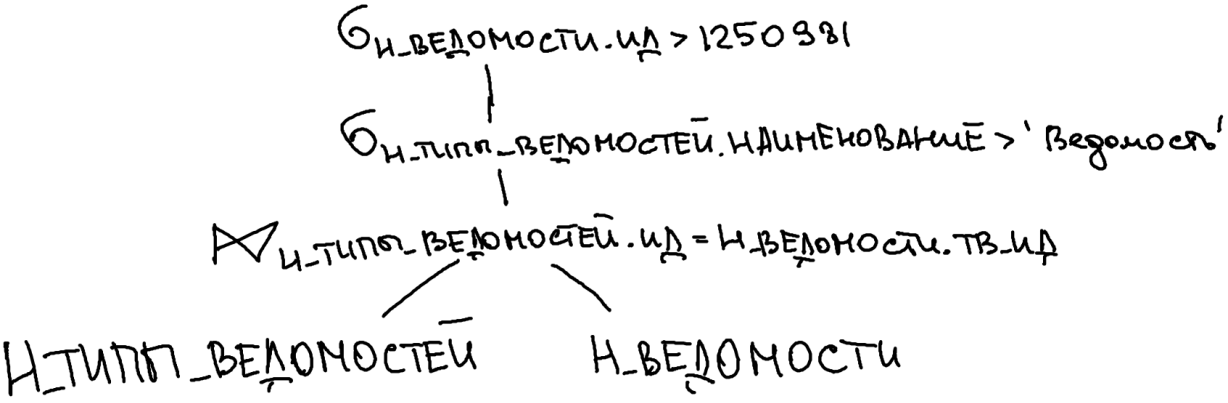
Для Н_ВЕДОМОСТИ.ИД можно использовать B3-tree, поскольку производится сравнение на больше меньше для отсечения данных.

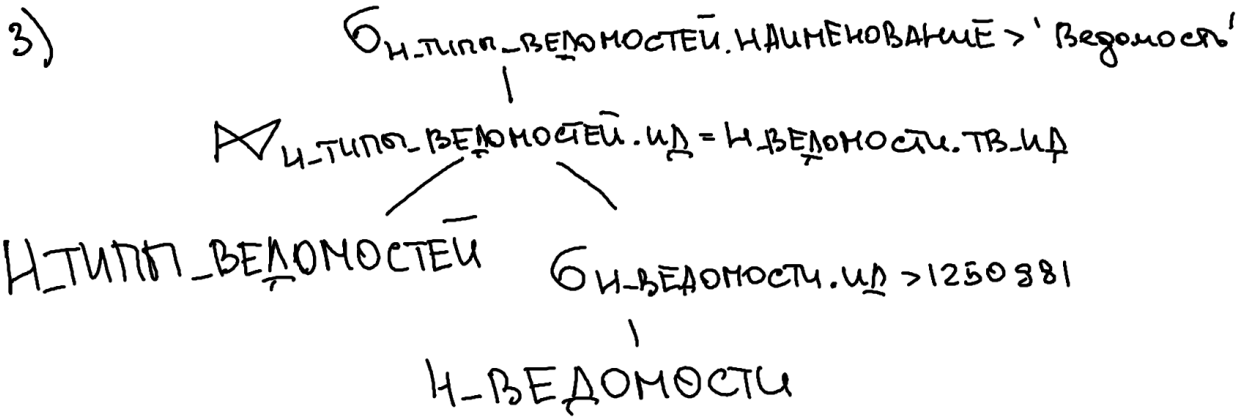
Для Н_ТИПЫ_ВЕДОМОСТЕЙ.ИД использовать B3-tree по той же причине.

Для Н_ТИПЫ_ВЕДОМОСТЕЙ.ИД и Н_ВЕДОМОСТИ.ТВ_ИД наиболее дающим выйгрыш в производительности будет давать HASH индексы, так как производится INNER JOIN на равенство, также можно использовать B3-дерево, так как его обход выполняется за O(мощность множества).

- Возможные планы (индексы отсутствуют):







* + проекцию можно писать на любом этапе, в котором это будет корректно.

- Наиболее оптимимальным планом является план 1, поскольку в случае отсутствия индексов одной из самых затратных операций является JOIN, поэтому сначала выборка данных с помощью WHERE, а затем JOIN будет наиболее эффективным планом.ъ\
- В случае добавления индексов, описанных выше, алгоритм скорее всего не поменяется, поскольку почти всегда, если мы сначала сделаем JOIN перед тем, как выбрать данные, то количество данных, которых придется фильтровать, будет скорее всего гораздо больше.

Запрос №2:

- Запрос SQL:

```sql
select Л_ЛЮДИ.ИД, Л_ОБУЧЕНИЯ.НЗК, Л_УЧЕНИКИ.НАЧАЛО
from (
    select *
    from Н_ЛЮДИ
    where ОТЧЕСТВО > 'Владимирович'
  ) as Л_ЛЮДИ
  inner join (
    select *
    from Н_ОБУЧЕНИЯ
    -- where Н_ОБУЧЕНИЯ.ЧЛВК_ИД > 163276 -- Слишком большое число, нет результатов
    where Н_ОБУЧЕНИЯ.ЧЛВК_ИД > 16327
  ) as Л_ОБУЧЕНИЯ on Л_ЛЮДИ.ИД = Л_ОБУЧЕНИЯ.ЧЛВК_ИД
  inner join (
    select *
    from Н_УЧЕНИКИ
    where Н_УЧЕНИКИ.ГРУППА::integer < 4103
  ) as Л_УЧЕНИКИ on
    Л_УЧЕНИКИ.ЧЛВК_ИД = Л_ОБУЧЕНИЯ.ЧЛВК_ИД and
    Л_УЧЕНИКИ.ВИД_ОБУЧ_ИД = Л_ОБУЧЕНИЯ.ВИД_ОБУЧ_ИД
;

select Н_ЛЮДИ.ИД, Н_ОБУЧЕНИЯ.НЗК, Н_УЧЕНИКИ.НАЧАЛО
from Н_ЛЮДИ
  inner join Н_ОБУЧЕНИЯ on Н_ЛЮДИ.ИД = Н_ОБУЧЕНИЯ.ЧЛВК_ИД
  inner join Н_УЧЕНИКИ on
    Н_УЧЕНИКИ.ЧЛВК_ИД = Н_ОБУЧЕНИЯ.ЧЛВК_ИД and
    Н_УЧЕНИКИ.ВИД_ОБУЧ_ИД = Н_ОБУЧЕНИЯ.ВИД_ОБУЧ_ИД
where
  Н_ЛЮДИ.ОТЧЕСТВО > 'Владимирович' and Н_ОБУЧЕНИЯ.ЧЛВК_ИД > 16327 and
  Н_УЧЕНИКИ.ГРУППА::integer < 4103
;
```

- Вывод EXPAIN_ANALYZE: (https://explain.depesz.com/s/i3iZa#html)

```
                                                    QUERY PLAN
-----------------------------------------------------------------------------------------------------------------------------
 Hash Join  (cost=390.87..1380.52 rows=3738 width=18) (actual time=5.452..17.028 rows=8389 loops=1)
   Hash Cond: (("Н_ЛЮДИ"."ИД" = "Н_ОБУЧЕНИЯ"."ЧЛВК_ИД") AND ("Н_УЧЕНИКИ"."ВИД_ОБУЧ_ИД" = "Н_ОБУЧЕНИЯ"."ВИД_ОБУЧ_ИД"))
   ->  Hash Join  (cost=195.79..1165.15 rows=3864 width=20) (actual time=3.461..12.563 rows=8389 loops=1)
         Hash Cond: ("Н_УЧЕНИКИ"."ЧЛВК_ИД" = "Н_ЛЮДИ"."ИД")
         ->  Seq Scan on "Н_УЧЕНИКИ"  (cost=0.00..948.94 rows=7770 width=16) (actual time=0.007..5.846 rows=16819 loops=1)
               Filter: (("ГРУППА")::integer < 4103)
               Rows Removed by Filter: 6492
         ->  Hash  (cost=163.97..163.97 rows=2545 width=4) (actual time=3.431..3.432 rows=2546 loops=1)
               Buckets: 4096  Batches: 1  Memory Usage: 122kB
               ->  Seq Scan on "Н_ЛЮДИ"  (cost=0.00..163.97 rows=2545 width=4) (actual time=0.009..3.069 rows=2546 loops=1)
                     Filter: (("ОТЧЕСТВО")::text > 'Владимирович'::text)
                     Rows Removed by Filter: 2572
   ->  Hash  (cost=119.76..119.76 rows=5021 width=14) (actual time=1.974..1.974 rows=5021 loops=1)
         Buckets: 8192  Batches: 1  Memory Usage: 295kB
         ->  Seq Scan on "Н_ОБУЧЕНИЯ"  (cost=0.00..119.76 rows=5021 width=14) (actual time=0.016..1.078 rows=5021 loops=1)
               Filter: ("ЧЛВК_ИД" > 16327)
 Planning Time: 1.054 ms
 Execution Time: 17.478 ms
(18 строк)
```

```
                                                    QUERY PLAN
-----------------------------------------------------------------------------------------------------------------------------
 Hash Join  (cost=390.87..1380.52 rows=3738 width=18) (actual time=5.448..17.132 rows=8389 loops=1)
   Hash Cond: (("Н_ЛЮДИ"."ИД" = "Н_ОБУЧЕНИЯ"."ЧЛВК_ИД") AND ("Н_УЧЕНИКИ"."ВИД_ОБУЧ_ИД" = "Н_ОБУЧЕНИЯ"."ВИД_ОБУЧ_ИД"))
   ->  Hash Join  (cost=195.79..1165.15 rows=3864 width=20) (actual time=3.451..12.653 rows=8389 loops=1)
         Hash Cond: ("Н_УЧЕНИКИ"."ЧЛВК_ИД" = "Н_ЛЮДИ"."ИД")
         ->  Seq Scan on "Н_УЧЕНИКИ"  (cost=0.00..948.94 rows=7770 width=16) (actual time=0.007..5.932 rows=16819 loops=1)
               Filter: (("ГРУППА")::integer < 4103)
               Rows Removed by Filter: 6492
         ->  Hash  (cost=163.97..163.97 rows=2545 width=4) (actual time=3.432..3.433 rows=2546 loops=1)
               Buckets: 4096  Batches: 1  Memory Usage: 122kB
               ->  Seq Scan on "Н_ЛЮДИ"  (cost=0.00..163.97 rows=2545 width=4) (actual time=0.007..3.078 rows=2546 loops=1)
                     Filter: (("ОТЧЕСТВО")::text > 'Владимирович'::text)
                     Rows Removed by Filter: 2572
   ->  Hash  (cost=119.76..119.76 rows=5021 width=14) (actual time=1.980..1.980 rows=5021 loops=1)
         Buckets: 8192  Batches: 1  Memory Usage: 295kB
         ->  Seq Scan on "Н_ОБУЧЕНИЯ"  (cost=0.00..119.76 rows=5021 width=14) (actual time=0.016..1.079 rows=5021 loops=1)
               Filter: ("ЧЛВК_ИД" > 16327)
 Planning Time: 0.893 ms
 Execution Time: 17.577 ms
(18 строк)
```

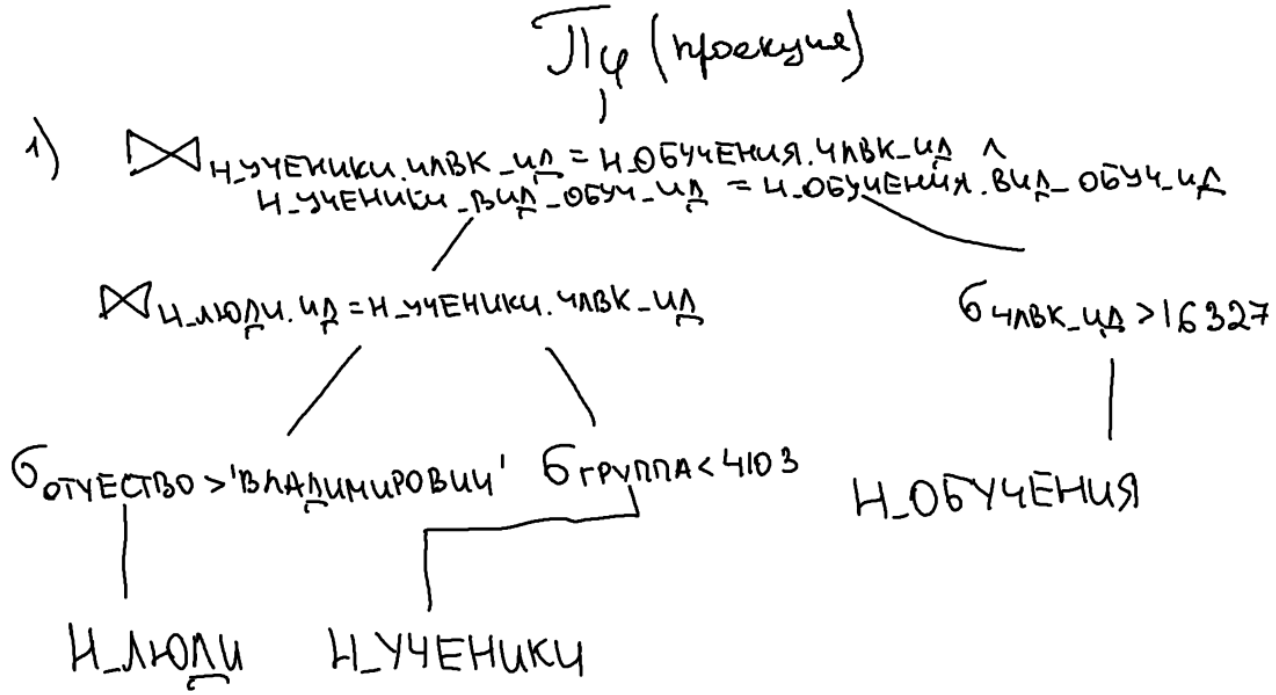
Также как и в первый раз планы для этих запросов исполнились одинаковые.

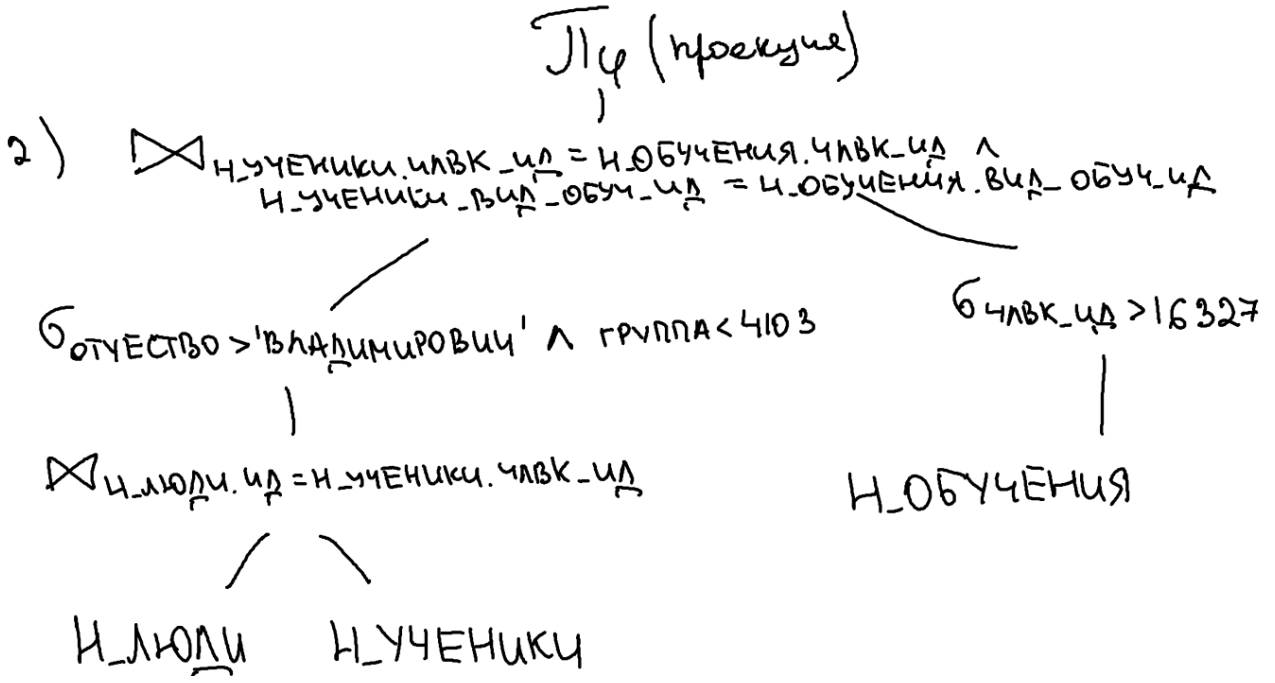
- Индексы:

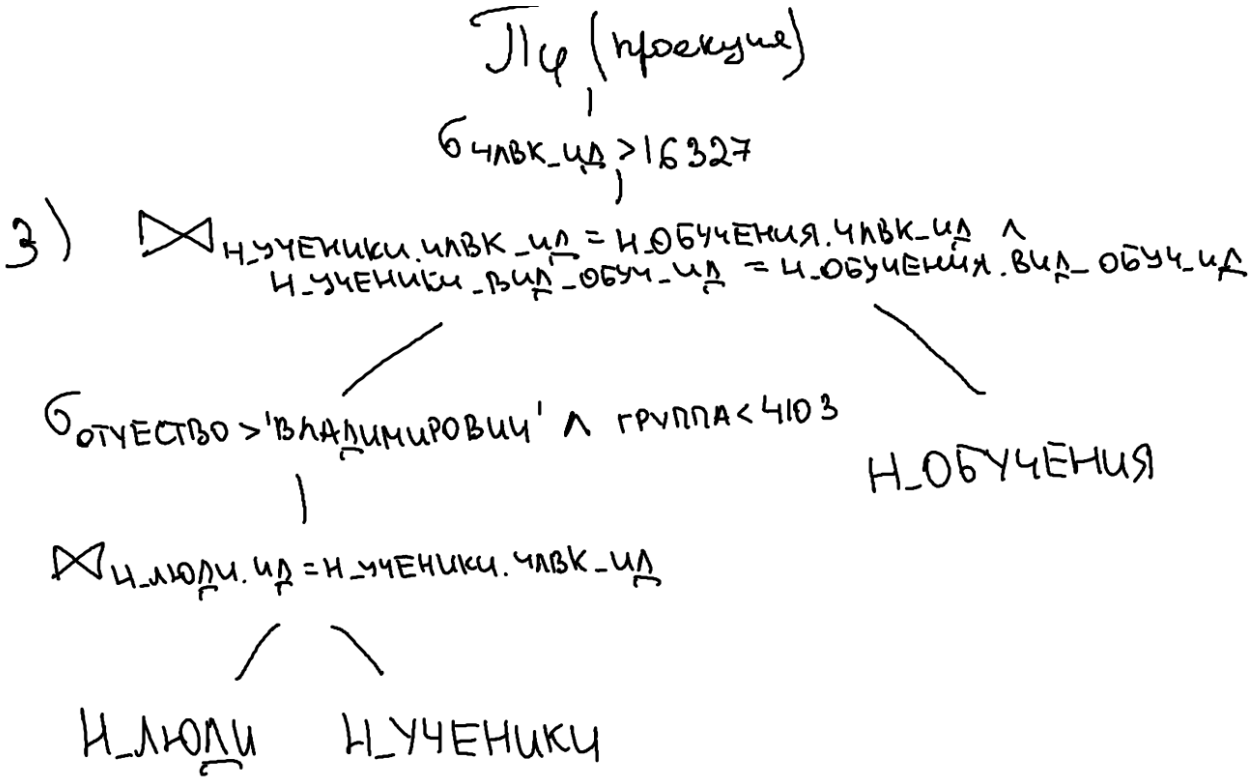
Для Н_ЛЮДИ.ИД использовать HASH, (Н_ОБУЧЕНИЯ.ЧЛВК_ИД и Н_ОБУЧЕНИЯ.ВИД_ОБУЧ_ИД) использовать HASH, Н_УЧЕНИКИ.ЧЛВК_ИД использовать HASH.

Для Н_ЛЮДИ.ОТЧЕСТВО, Н_УЧЕНИКИ.ГРУППА использовать B-tree из-за сравнения.

- Возможные планы (индексы отсутствуют):







- Наиболее оптимимальным планом является также план 1, по причине уменьшения данных во время JOIN, так как это самая затратная операция (так where O(n), a join — Ω(n + m), где n, m — мощности таблиц).
- В случае добавления индексов, описанных выше, алгоритм скорее всего не поменяется, поскольку почти всегда, если мы сначала сделаем JOIN перед тем, как выбрать данные, то количество данных, которых придется фильтровать, будет скорее всего гораздо больше.

Ссылка на гитхаб: https://github.com/zubrailx/University-ITMO/tree/main/Year-3/Information-systems-and-databases/lab-4/sql/main.sql

3. **Выводы по работе.**

В ходе выполнения данной лабораторной работы я познакомился с индексами, алгоритмом B-tree для построения индексов. Также научился анализировать вывод EXPLAIN (ANALYZE, BUFFERS) и строить планы для SQL запросов. Благодаря этим знаниям я смогу построить более быструю СУБД на курсе низкоуровневого программирования!