МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

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

«ИНФОРМАЦИОННЫЕ СИСТЕМЫ И БАЗЫ ДАННЫХ»

вариант 11209

Выполнил:

Студент группы P33312

Лысенко А.К.

Преподаватель:

Наумова Н.А.

Санкт-Петербург, 2023

Текст задания

Введите вариант: 11209

Внимание! У разных вариантов разный 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_ОЦЕНКИ, Н_ВЕДОМОСТИ.
   Вывести атрибуты: Н_ОЦЕНКИ.КОД, Н_ВЕДОМОСТИ.ЧЛВК_ИД.
   Фильтры (AND):
   a) Н_ОЦЕНКИ.ПРИМЕЧАНИЕ = отлично.
   b) Н_ВЕДОМОСТИ.ЧЛВК_ИД > 142390.
   c) Н_ВЕДОМОСТИ.ЧЛВК_ИД > 153285.
   Вид соединения: LEFT 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.ИД.
   Фильтры: (AND)
   a) Н_ЛЮДИ.ИМЯ < Николай.
   b) Н_ОБУЧЕНИЯ.ЧЛВК_ИД > 105590.
   Вид соединения: LEFT JOIN.

1 запрос:

```
select "Н_ОЦЕНКИ"."КОД", "Н_ВЕДОМОСТИ"."ЧЛВК_ИД"
                from "Н_ОЦЕНКИ"
                         left join "Н_ВЕДОМОСТИ" on "Н_ВЕДОМОСТИ"."ОЦЕНКА" = "Н_ОЦЕНКИ"."КОД"
                where
                        "Н_ОЦЕНКИ"."ПРИМЕЧАНИЕ" = 'отлично' and
                        "Н_ВЕДОМОСТИ"."ЧЛВК_ИД" > 142390 and
                        "Н_ВЕДОМОСТИ"."ЧЛВК_ИД" > 153285;
```

Output | Result 5 | Result 4-2 | Tx

1-500 of 501+

	"КОД"	"ЧЛВК_ИД"
1	5	153942
2	5	153355
3	5	153291
4	5	154531
5	5	158528
6	5	155223
7	5	155225
8	5	153310
9	5	153367
10	5	153342
11	5	153329
12	5	158528
13	5	155219
14	5	155221
15	5	155222
16	5	155223

2 запрос

```
select "Н_ЛЮДИ"."ИМЯ", "Н_ОБУЧЕНИЯ"."ЧЛВК_ИД", "Н_УЧЕНИКИ"."ИД"
                from "Н_ЛЮДИ"
                         left join "Н_ОБУЧЕНИЯ" on "Н_ОБУЧЕНИЯ"."ЧЛВК_ИД" = "Н_ЛЮДИ"."ИД"
                         left join "Н_УЧЕНИКИ" on "Н_УЧЕНИКИ"."ЧЛВК_ИД" = "Н_ОБУЧЕНИЯ"."ЧЛВК_ИД"
                where "Н_ЛЮДИ"."ИМЯ" < 'Николай' and
                      "Н_ОБУЧЕНИЯ"."ЧЛВК_ИД" > 105590;
```

Output | Result 3 | Tx | 1-500 of 501+ | CSV

	"ИМЯ"	"ЧЛВК_ИД"	"ИД"
3	Александр	116438	47146
4	Максим	118663	47147
5	Вениамин	118678	47148
6	Андрей	118718	47150
7	Михаил	125662	47275
8	Дмитрий	125683	47276
9	Илья	125691	47277
10	Александра	125697	47278
11	Джозефин Абаках	121965	34327
12	Кристина	120910	34256
13	Владимир	120189	34563
14	Зоя	119463	34603

Создание индексов

1 запрос:

```
create index notes on "Н_ОЦЕНКИ" using hash("ПРИМЕЧАНИЕ");
create index ids on "Н_ВЕДОМОСТИ" using btree("ЧЛВК_ИД");
```

Для таблицы Н_ОЦЕНКИ для атрибута ПРИМЕЧАНИЕ имеет смысл создать индекс, так как он используется в фильтре. Имеет смысл использовать хэш-индекс, так как в фильтре используется проверка на равенства. Хэши строк сравниваются за константу. Это поможет улучшить скорость операций WHERE и ON.

Для таблицы Н_ВЕДОМОСТИ для атрибута ЧЛВК_ИД имеет смысл создать индекс, так как он используется в фильтре. Имеет смысл использовать индекс на основе b-tree дерева, так в фильтре используется оператор >. Используя такой индекс мы снизим время поиска с линейного до логарифмического (O(n) → O(logN) при выполнении выборки данных при использовании WHERE.

2 запрос:

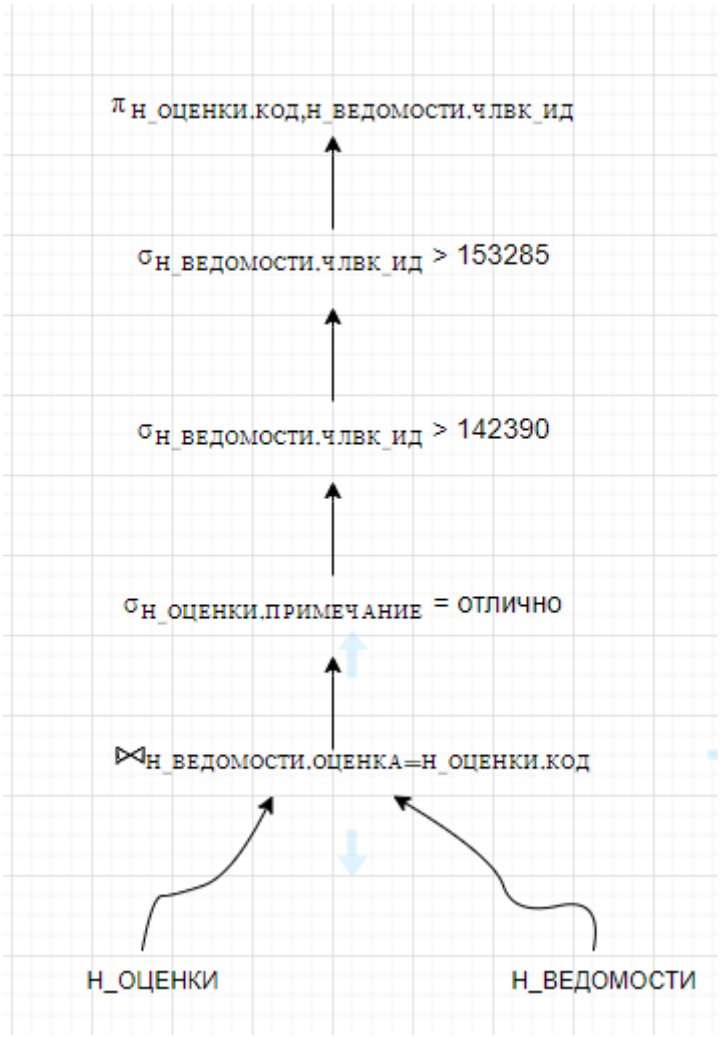
```
create index names on "Н_ЛЮДИ" using btree("ИМЯ");
create index ids on "Н_ОБУЧЕНИЯ" using btree("ЧЛВК_ИД");
```

Для таблицы Н_ЛЮДИ для атрибута ИМЯ имеет смысл создать индекс, так как он используется в фильтре. Имеет смысл использовать индекс на основе b-tree дерева, так как мы используем оператор >. Создание данного индекса улучшить выборку данных при использовании WHERE.

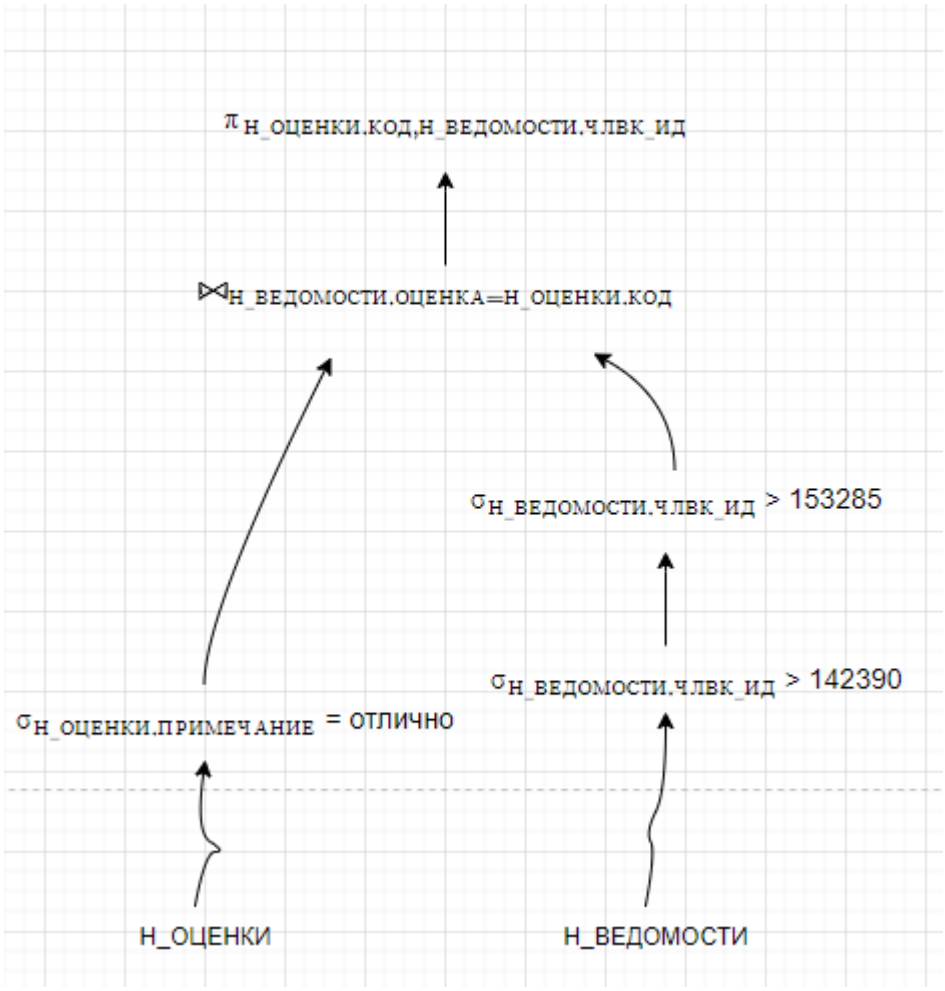
Для таблицы Н_ОБУЧЕНИЯ для атрибута ЧЛВК_ИД имеет смысл создать индекс, так как он используется в фильтре. Имеет смысл использовать индекс на основе b-tree дерева, так в фильтре используется оператор >. Используя такой индекс мы снизим время поиска с линейного до логарифмического (O(n) → O(logN) при выполнении выборки данных при использовании WHERE.

1 запрос. План выполнения.

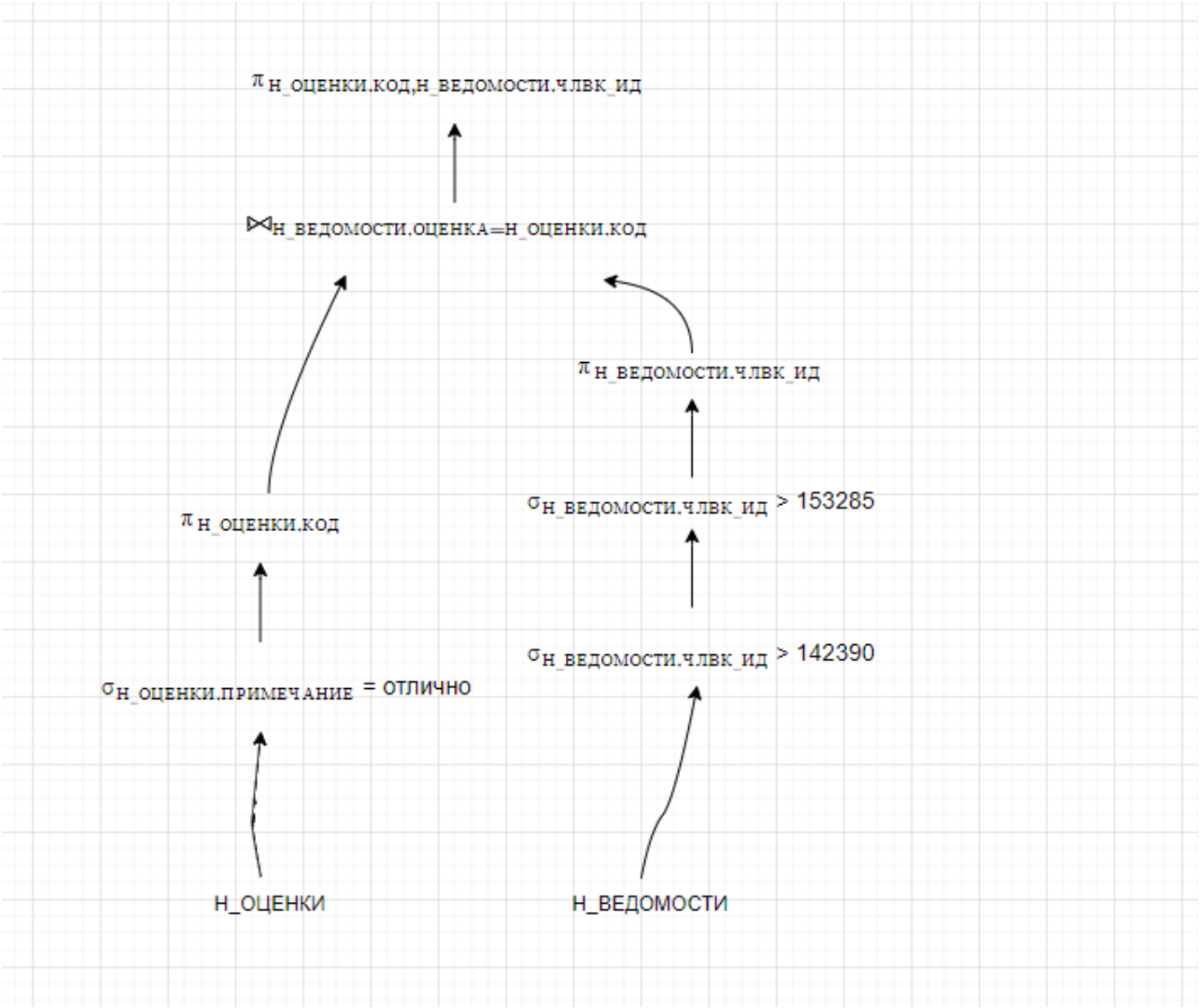
1) Сначала происходит соединение отношений, далее последовательная выборка, результатом будет проекция двух атрибутов



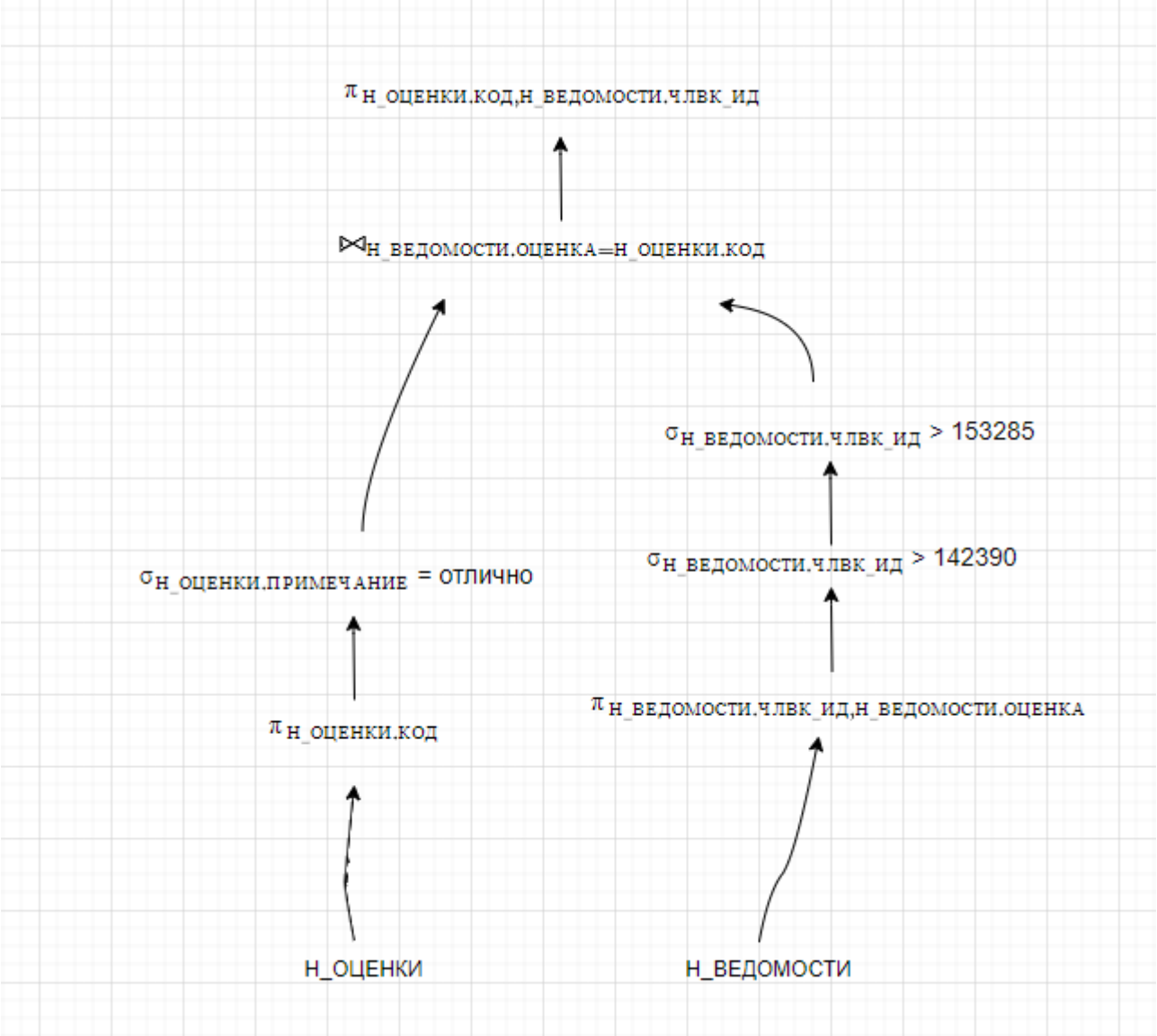
2) Сначала происходит операция выборки, после соединение двух отношений, результатом будет проекция двух атрибутов.



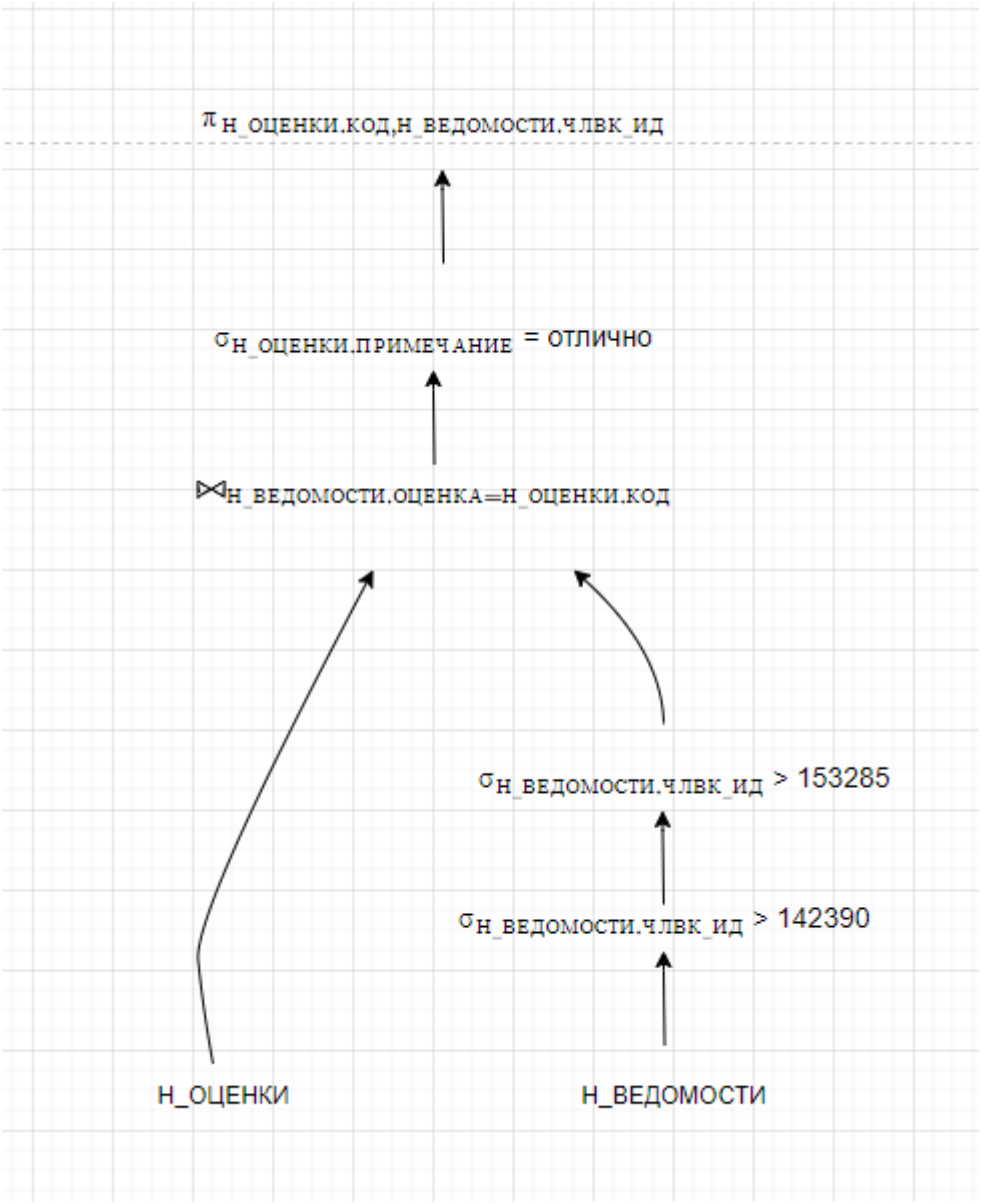
3) Сначала производится выборка для двух отношений, далее получается проекция для двух отношений из атрибутов, необходимых для выборки, соединения и итоговой проекции, далее соединение двух отношений, результатом является проекция двух атрибутов.



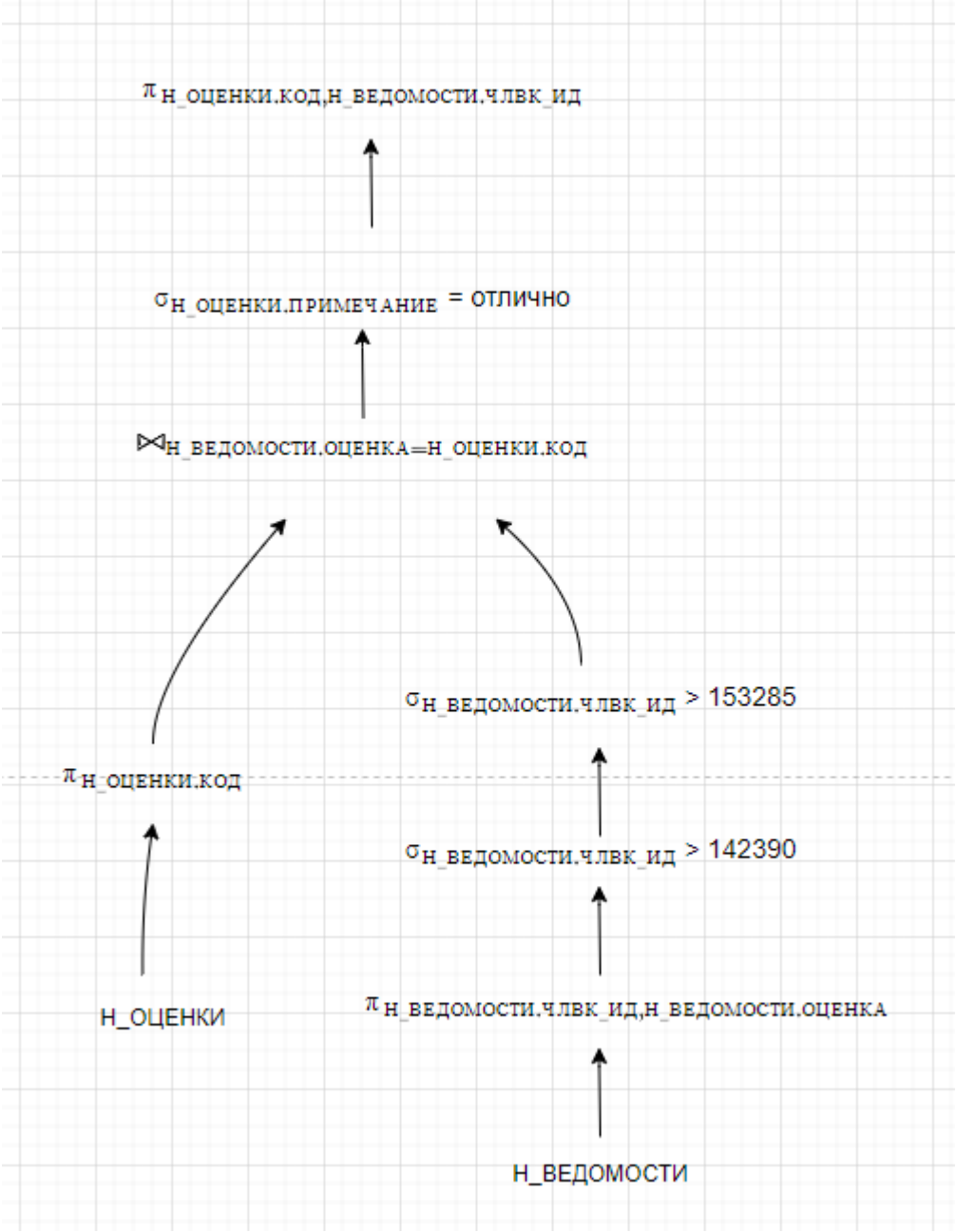
4) Для начала получается проекция для двух отношений из атрибутов, необходимых для выборки, соединения и итоговой проекции, далее производится выборка и соединение. Результатом является проекция двух атрибутов.



5) Для начала получается выборка для одного из отношений из атрибутов, далее происходит соединение таблиц, после которого производится выборка для результата соединения. Результатом является проекция двух атрибутов. В данном плане атрибуты выборки после соединения могут меняться – могут быть выборки из ветки Н_ВЕДОМОСТИ.



6) Для начала получается проекция для двух отношений из атрибутов, необходимых для выборки, соединения и итоговой проекции, далее производится выборка для одного из отношений, а после соединения двух отношений, после которого производится выборка для результата отношения. Результатом является проекция двух атрибутов. В данном плане атрибуты выборки после соединения могут меняться – могут быть выборки из ветки Н_ВЕДОМОСТИ.



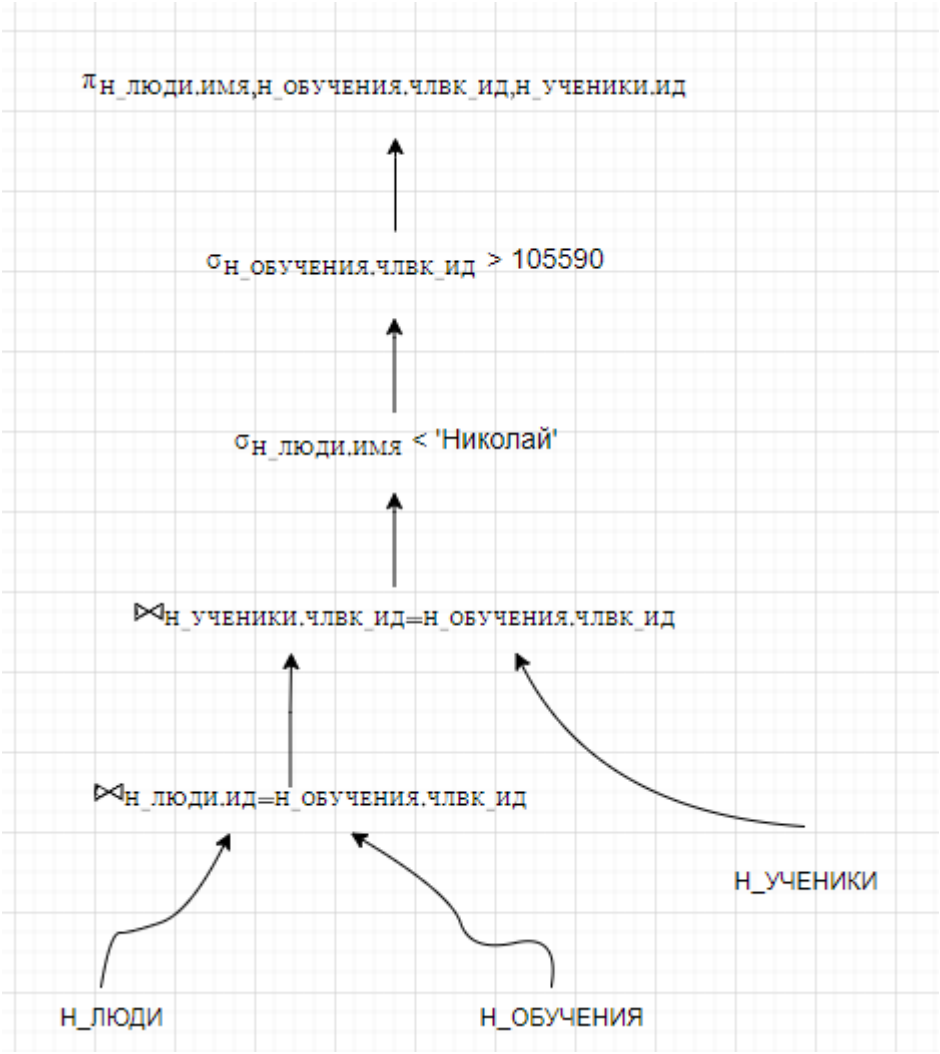
Из составленных возможных планов выполнения запроса лучшим является четвертый, поскольку в нем изначально получаются проекции и производятся выборки, а уже после этого выполняются соединение. Это позволяет уменьшить размер хранимых данных.

При создании индексов четвертый план останется оптимальным, при этом даже ускорится за счёт ускорения поиска. Также довольно эффективным будет третий план, потому что скорость поиска в нем увеличится. В нем также соединение происходит после выборки, что позволяет ускорить выполнение запроса, но, в отличие от четвертого плана, в нем выборка будет

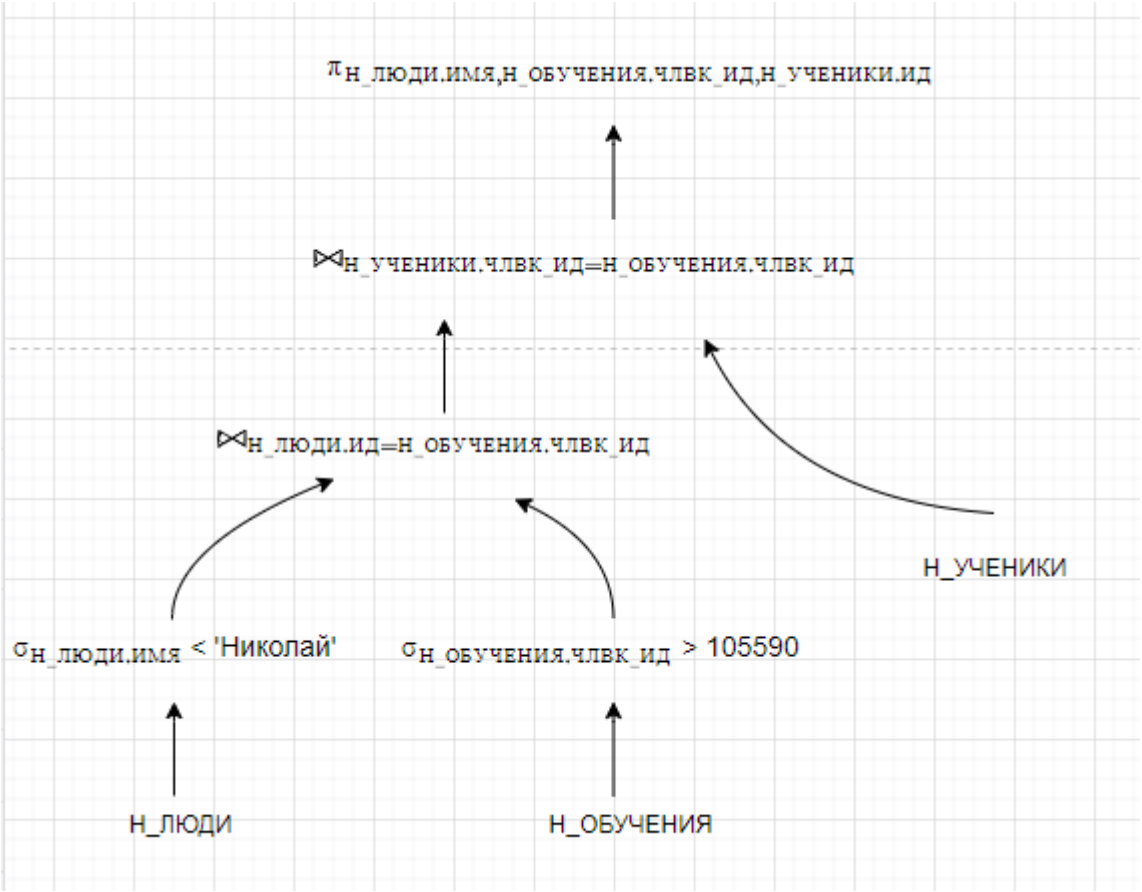
происходить по все таблице целиком, а не по отдельным проекциям.

2 запрос. План выполнения.

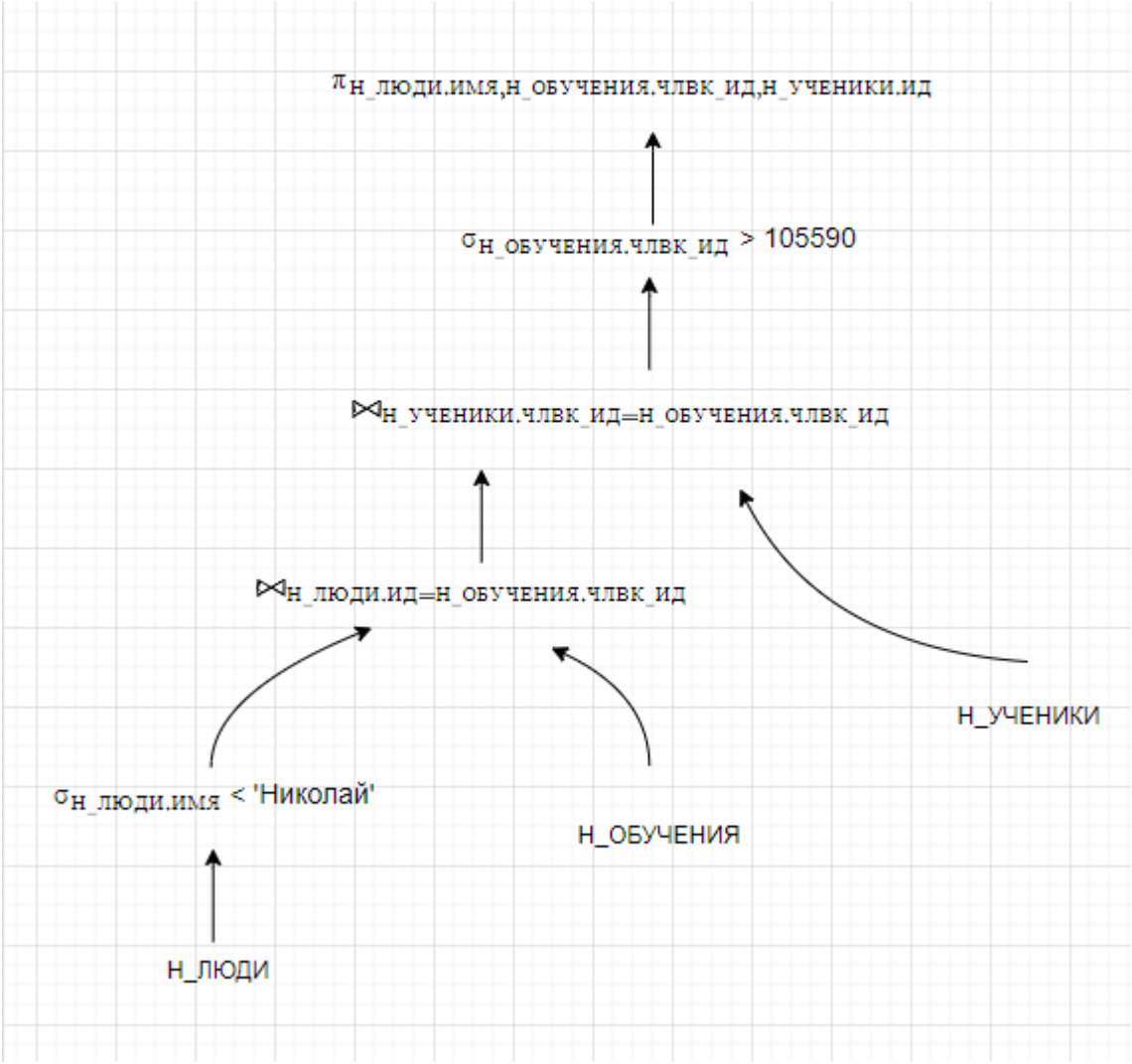
1) Соединение двух отношений, далее результат соединения соединяется с третьим отношением, а для результата двух соединений последовательно производится выборка. Результатом является проекция трех атрибутов.



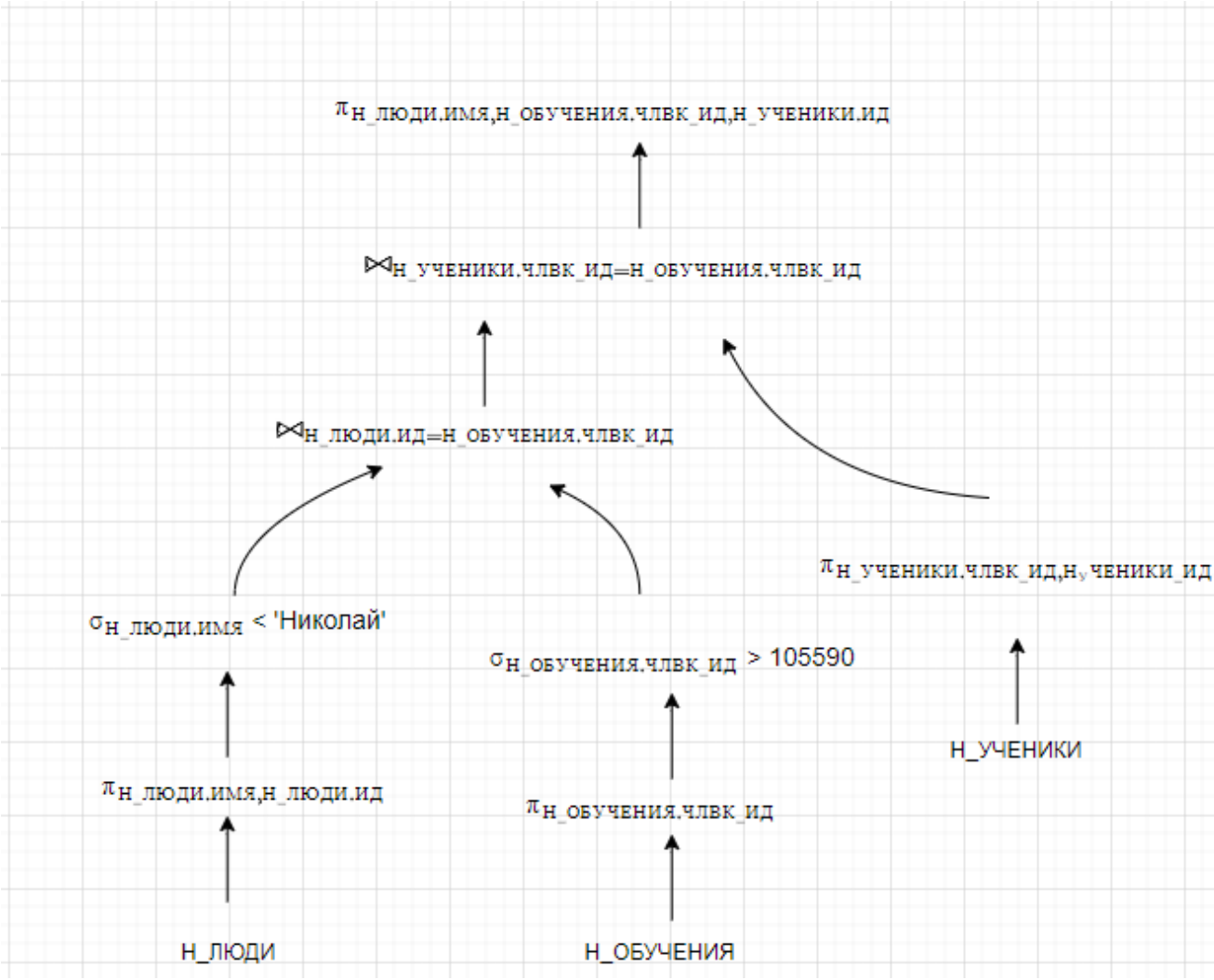
2) Для начала производится выборка для первых двух отношений, далее производится соединение этих двух отношений, а после результат соединения соединяется третьим отношением. Результатом является проекция трёх атрибутов.



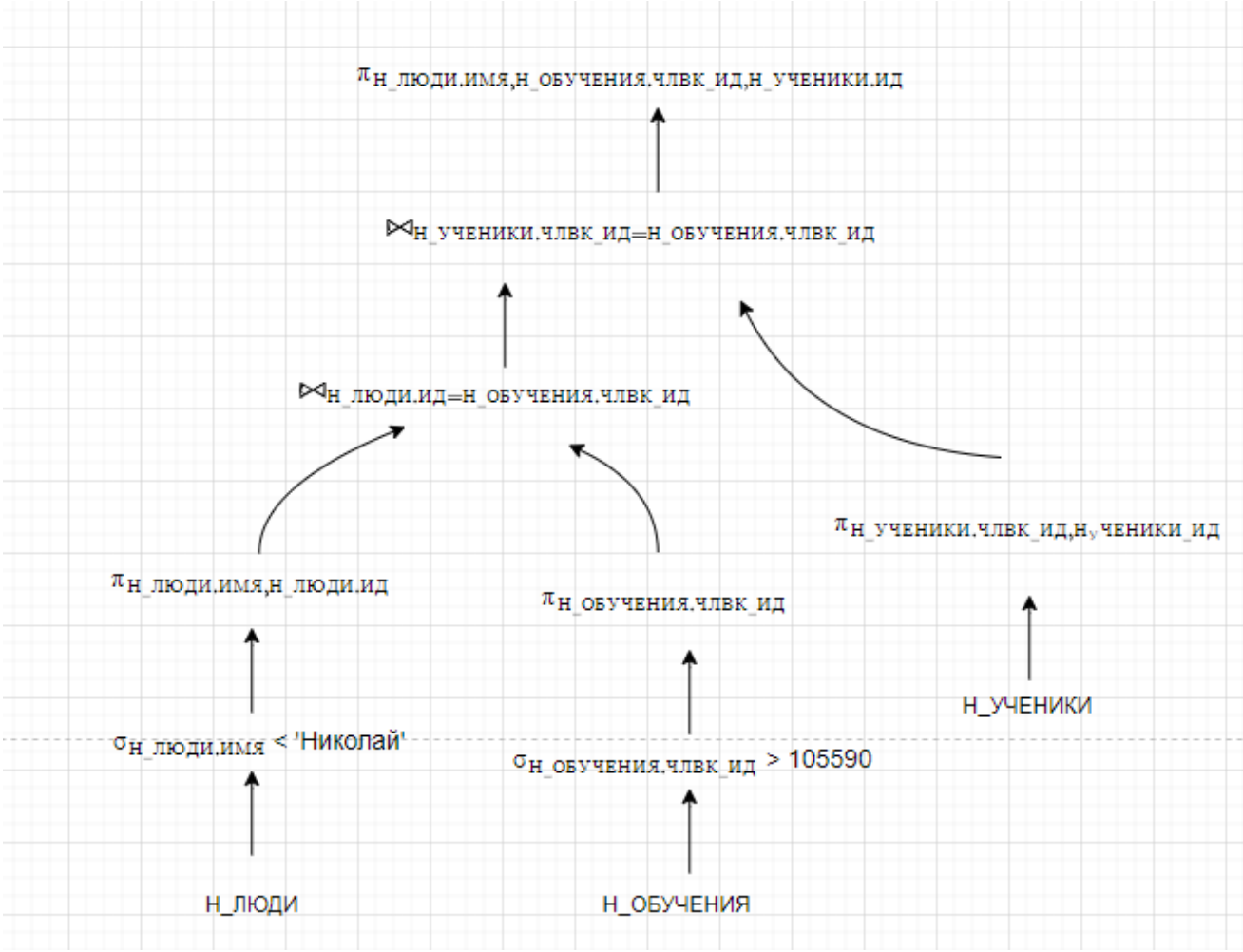
3) Сначала производится выборка для одного отношения, далее производится соединение первых двух отношений, а после результат соединения соединяется с третьим отношением. Происходит выборка по результату соединения. Результатом является проекция трёх атрибутов. На первом шаге выборка может быть у ветки Н_ОБУЧЕНИЯ.



4) Сначала получается проекция для двух отношений, после по ним производится выборка и соединение. Далее результат соединения соединяется с третьим отношением, для которого перед соединением была получена проекция. Результатом является проекция трёх атрибутов.

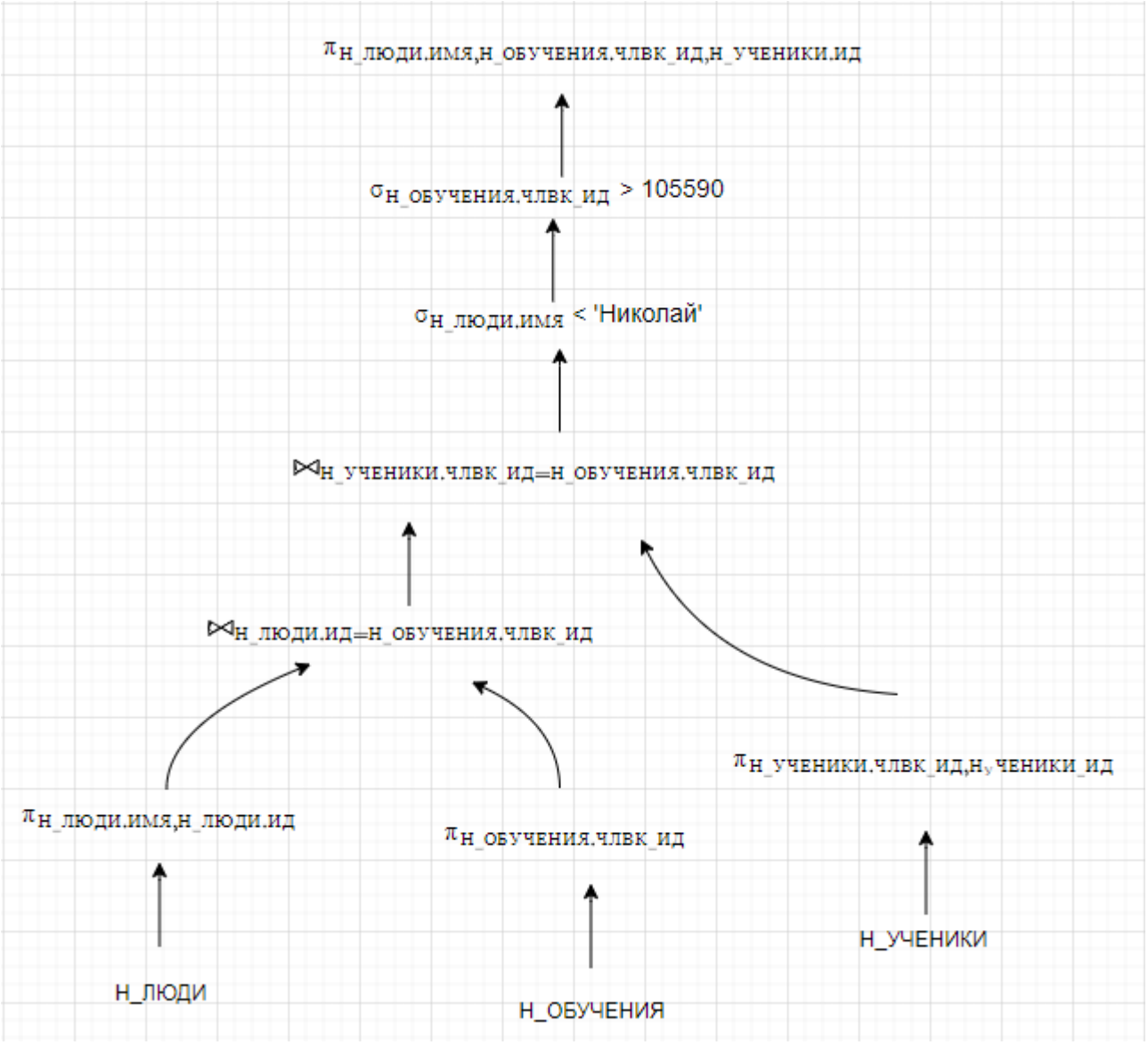


5) Сначала формируются выборки для каждого отношения, далее строятся проекции. После чего данные проекции соединяются. Результатом является проекция трёх атрибутов.



6) Получаются проекции для двух отношений, после чего эти отношения соединяются, а

результат соединения соединяется с третьим отношением, для которого перед соединением тоже была получена проекция. Далее производится последовательная выборка. Результатом является проекция трёх атрибутов.



Из составленных возможных планов выполнения запросы лучшим является четвертый, поскольку в нем изначально получаются проекции и производятся выборки, а уже после этого выполняется соединение. Это позволяет уменьшить размер хранимых данных.

При создании индексов четвертый план останется оптимальным, при этом даже ускорится за счёт ускорения поиска. Также довольно эффективным будет пятый план, потому что скорость поиска в нем увеличится. В нем также соединение происходит после выборки, что позволяет ускорить выполнение запроса, но, в отличие от четвертого плана, в нем выборка будет происходить по всей таблице целиком, а не по отдельным проекциям.

```
Nested Loop  (cost=398.97..2742.22 rows=1001 width=9) (actual time=2.240..2.916 rows=815 loops=1)
  ->  Seq Scan on "Н_ОЦЕНКИ"  (cost=0.00..1.11 rows=1 width=5) (actual time=0.008..0.010 rows=1 loops=1)
        Filter: (("ПРИМЕЧАНИЕ")::text = 'отлично'::text)
        Rows Removed by Filter: 8
  ->  Bitmap Heap Scan on "Н_ВЕДОМОСТИ"  (cost=398.97..2731.10 rows=1001 width=10) (actual time=2.226..2.784 rows=815 loops=1)
        Recheck Cond: (("ЧЛВК_ИД" > 142390) AND ("ЧЛВК_ИД" > 153285) AND (("ОЦЕНКА")::text = ("Н_ОЦЕНКИ"."КОД")::text))
        Heap Blocks: exact=253
        ->  BitmapAnd  (cost=398.97..398.97 rows=1001 width=0) (actual time=2.186..2.187 rows=0 loops=1)
              ->  Bitmap Index Scan on "ВЕД_ЧЛВК_FK_IFK"  (cost=0.00..122.42 rows=9013 width=0) (actual time=0.483..0.483 rows=9332 loops=1)
                    Index Cond: (("ЧЛВК_ИД" > 142390) AND ("ЧЛВК_ИД" > 153285))
              ->  Bitmap Index Scan on "ВЕД_ОЦЕНКА_I"  (cost=0.00..273.79 rows=24716 width=0) (actual time=1.658..1.658 rows=37825 loops=1)
                    Index Cond: (("ОЦЕНКА")::text = ("Н_ОЦЕНКИ"."КОД")::text)
Planning Time: 1.349 ms
Execution Time: 3.016 ms
```

```
QUERY PLAN
Hash Right Join  (cost=393.61..1464.28 rows=18000 width=21) (actual time=6.329..15.649 rows=18011 loops=1)
  Hash Cond: ("Н_УЧЕНИКИ"."ЧЛВК_ИД" = "Н_ОБУЧЕНИЯ"."ЧЛВК_ИД")
  ->  Seq Scan on "Н_УЧЕНИКИ"  (cost=0.00..774.11 rows=23311 width=8) (actual time=0.013..3.248 rows=23311 loops=1)
  ->  Hash  (cost=345.74..345.74 rows=3830 width=17) (actual time=6.297..6.301 rows=3835 loops=1)
        Buckets: 4096  Batches: 1  Memory Usage: 225kB
        ->  Hash Join  (cost=212.79..345.74 rows=3830 width=17) (actual time=3.558..5.552 rows=3835 loops=1)
              Hash Cond: ("Н_ОБУЧЕНИЯ"."ЧЛВК_ИД" = "Н_ЛЮДИ"."ИД")
              ->  Seq Scan on "Н_ОБУЧЕНИЯ"  (cost=0.00..119.76 rows=5020 width=4) (actual time=0.017..0.835 rows=5020 loops=1)
                    Filter: ("ЧЛВК_ИД" > 105590)
                    Rows Removed by Filter: 1
              ->  Hash  (cost=163.97..163.97 rows=3905 width=17) (actual time=3.525..3.527 rows=3903 loops=1)
                    Buckets: 4096  Batches: 1  Memory Usage: 228kB
                    ->  Seq Scan on "Н_ЛЮДИ"  (cost=0.00..163.97 rows=3905 width=17) (actual time=0.010..2.779 rows=3903 loops=1)
                          Filter: (("ИМЯ")::text < 'Николай'::text)
                          Rows Removed by Filter: 1215
Planning Time: 27.746 ms
Execution Time: 16.578 ms
```

Вывод: узнал, что такое индексы, какие индексы бывают и как они помогают уменьшить время выполнения запроса.