Федеральное государственное автономное образовательное учреждение высшего образования «Национальный исследовательский университет ИТМО»

Факультет Программной Инженерии и Компьютерной Техники

Лабораторная работа №4

По Базам Данных

Вариант 34561

Выполнил:

Ларионов Владислав Васильевич

Группа P3109

Практик:

Воронина Дарья Сергеевна

Санкт-Петербург 2025

Содержание

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.ДАТА.
   Фильтры (AND):
   a) Н_ТИПЫ_ВЕДОМОСТЕЙ.ИД < 3.
   b) Н_ВЕДОМОСТИ.ИД = 1457443.
   Вид соединения: LEFT JOIN.

2. 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.ЧЛВК_ИД, Н_СЕССИЯ.ДАТА.
   Фильтры (AND):
   a) Н_ЛЮДИ.ИМЯ = Ярослав.
   b) Н_ВЕДОМОСТИ.ИД > 1250972.

c) Н_СЕССИЯ.ИД < 27640.

Вид соединения: INNER JOIN.

Выполнение задания

Запрос 1

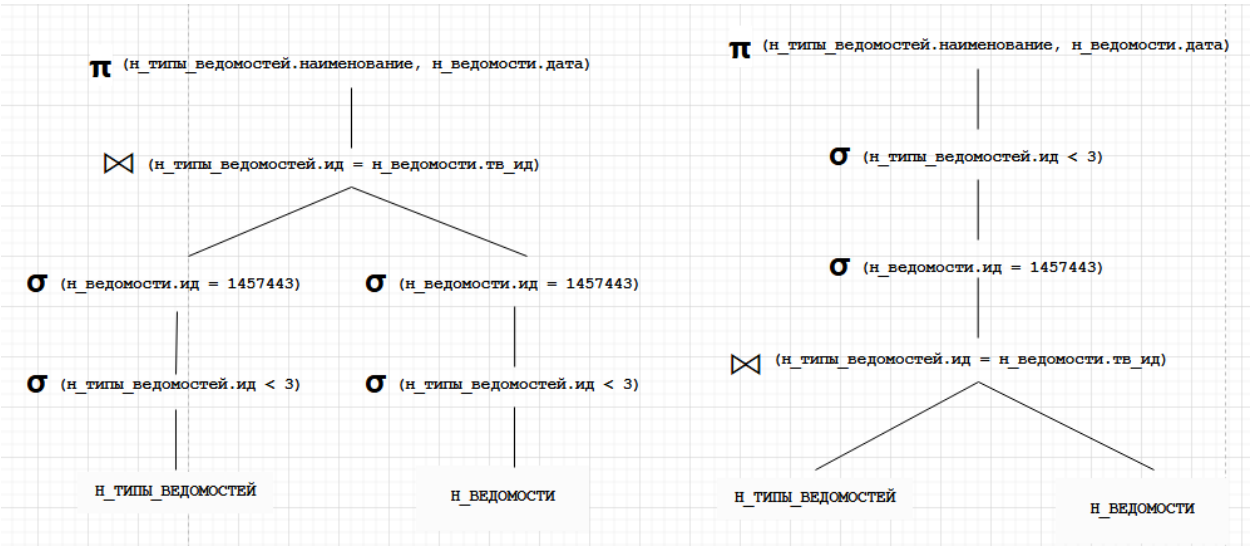
1.1 Запрос 1 на языке PostgreSQL

```sql
SELECT
    Н_ТИПЫ_ВЕДОМОСТЕЙ.НАИМЕНОВАНИЕ,
    Н_ВЕДОМОСТИ.ДАТА
FROM
    Н_ТИПЫ_ВЕДОМОСТЕЙ
LEFT JOIN
    Н_ВЕДОМОСТИ ON Н_ТИПЫ_ВЕДОМОСТЕЙ.ИД = Н_ВЕДОМОСТИ.ТВ_ИД
WHERE
    Н_ТИПЫ_ВЕДОМОСТЕЙ.ИД < 3
    AND Н_ВЕДОМОСТИ.ИД = 1457443;
```

1.2 Индексы для запроса 1

На PK индексы создавать не надо.

1.3 План выполнения без индексов для запроса 1



Как мы видим, первый план выполнения более оптимальный, так как сначала происходит отбор по условиям, а только потом объединение таблиц (то есть объединяется меньшее количество строк)

1.4 EXPLAIN ANALYZE для запроса 1

```
                                                         QUERY PLAN
------------------------------------------------------------------------------------------------------------------------------
 Nested Loop  (cost=0.42..9.49 rows=1 width=426) (actual time=0.028..0.034 rows=1 loops=1)
   Join Filter: ("Н_ТИПЫ_ВЕДОМОСТЕЙ"."ИД" = "Н_ВЕДОМОСТИ"."ТВ_ИД")
   Rows Removed by Join Filter: 1
   ->  Seq Scan on "Н_ТИПЫ_ВЕДОМОСТЕЙ"  (cost=0.00..1.04 rows=1 width=422) (actual time=0.013..0.014 rows=2 loops=1)
         Filter: ("ИД" < 3)
         Rows Removed by Filter: 1
   ->  Index Scan using "ВЕД_РК" on "Н_ВЕДОМОСТИ"  (cost=0.42..8.44 rows=1 width=12) (actual time=0.006..0.007 rows=1 loops=2)
         Index Cond: ("ИД" = 1457443)
 Planning Time: 0.314 ms
 Execution Time: 0.071 ms
(10 строк)
```

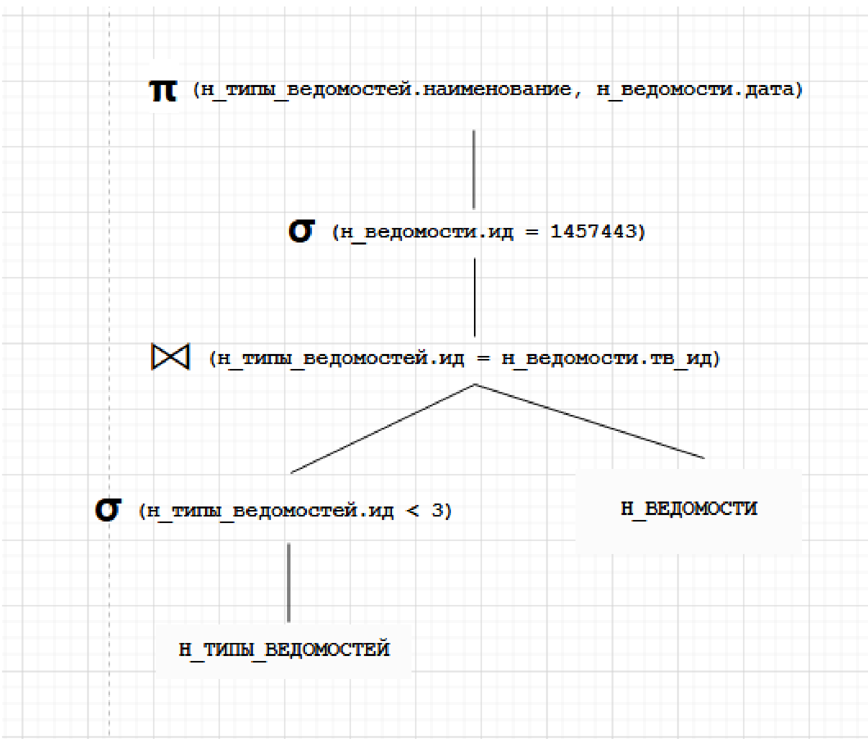
Запрос соединяет таблицы "Н_ТИПЫ_ВЕДОМОСТЕЙ" и "Н_ВЕДОМОСТИ" через Nested Loop.

Сначала выполняется полное сканирование "Н_ТИПЫ_ВЕДОМОСТЕЙ" с фильтром "ИД < 3", которое находит 2 строки и отбрасывает 1.

Затем для каждой из этих строк происходит поиск в "Н_ВЕДОМОСТИ" по индексу первичного ключа "ВЕД_РК" с условием "ИД = 1457443", каждый поиск занимает около 0.015 мс.

После этого применяется условие соединения ("Н_ТИПЫ_ВЕДОМОСТЕЙ"."ИД" = "Н_ВЕДОМОСТИ"."ТВ_ИД"), из двух возможных комбинаций остается одна строка, а вторая отбрасывается.

Общее время выполнения запроса — 0.122 мс, время планирования — 1.117 мс.



Запрос 2

2.1 Запрос 2 на языке PostgreSQL

```
SELECT
    Н_ЛЮДИ.ОТЧЕСТВО,
    Н_ВЕДОМОСТИ.ЧЛВК_ИД,
    Н_СЕССИЯ.ДАТА
FROM
    Н_ЛЮДИ
JOIN
    Н_ВЕДОМОСТИ ON Н_ЛЮДИ.ИД = Н_ВЕДОМОСТИ.ЧЛВК_ИД
JOIN
    Н_СЕССИЯ ON Н_ЛЮДИ.ИД = Н_СЕССИЯ.ЧЛВК_ИД
WHERE
    Н_ЛЮДИ.ИМЯ = 'Ярослав'
    AND Н_ВЕДОМОСТИ.ИД > 1250972
    AND Н_СЕССИЯ.ИД < 27640;
```

2.2 Индексы для запроса 2

CREATE INDEX idx_люди_имя ON Н_ЛЮДИ USING HASH

(ИМЯ);

Тип: Hash (проверяется всего одно значение, для проверки равенства быстрее, чем B-tree)

```
 tablename | attname | n_distinct | null_frac | avg_width | correlation
-----------+---------+------------+-----------+-----------+-------------
 Н_ЛЮДИ    | ИМЯ     |        350 |         0 |        13 | 0.032884054
```

Как мы видим, 350 уникальных значений, поэтому индекс лишним не будет. Также строки плохо отсортированы, индекс ускорит работу

CREATE INDEX idx_ведомости_члвк_ид ON Н_ВЕДОМОСТИ(ЧЛВК_ИД);

Тип: B-tree

Ускоряет соединение с Н_ЛЮДИ

tablename	attname	n_distinct	null_frac	avg_width	correlation
Н_ВЕДОМОСТИ	ЧЛВК_ИД	3261	0	4	0.51113844

indexname	indexdef
ВЕД_РК	CREATE UNIQUE INDEX "ВЕД_РК" ON public."Н_ВЕДОМОСТИ" USING btree ("ИД")
ВЕД_ИП_FK_I	CREATE INDEX "ВЕД_ИП_FK_I" ON public."Н_ВЕДОМОСТИ" USING btree ("СЭС_ИД")
ВЕД_ОТД_I	CREATE INDEX "ВЕД_ОТД_I" ON public."Н_ВЕДОМОСТИ" USING btree ("ОТД_ИД")
ВЕД_ТВ_FK_I	CREATE INDEX "ВЕД_ТВ_FK_I" ON public."Н_ВЕДОМОСТИ" USING btree ("ТВ_ИД")
ВЕД_ЧЛВК_FK_IFK	CREATE INDEX "ВЕД_ЧЛВК_FK_IFK" ON public."Н_ВЕДОМОСТИ" USING btree ("ЧЛВК_ИД")

Индекс уже существует, но если бы его не было, то его создание не было бы лишним – 3261 уникальных значений и отсортировано всего половина списка.

CREATE INDEX idx_сессия_члвк_ид ON Н_СЕССИЯ(ЧЛВК_ИД);

Тип: B-tree

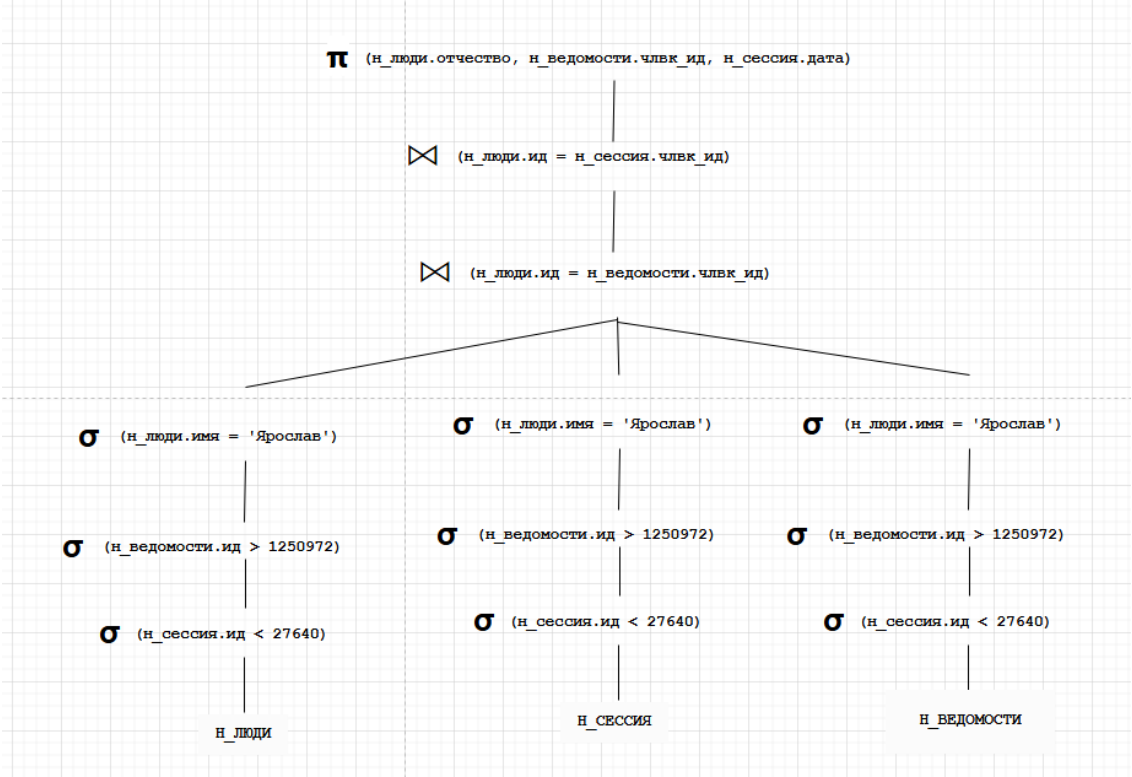
Ускорит соединение с Н_ЛЮДИ

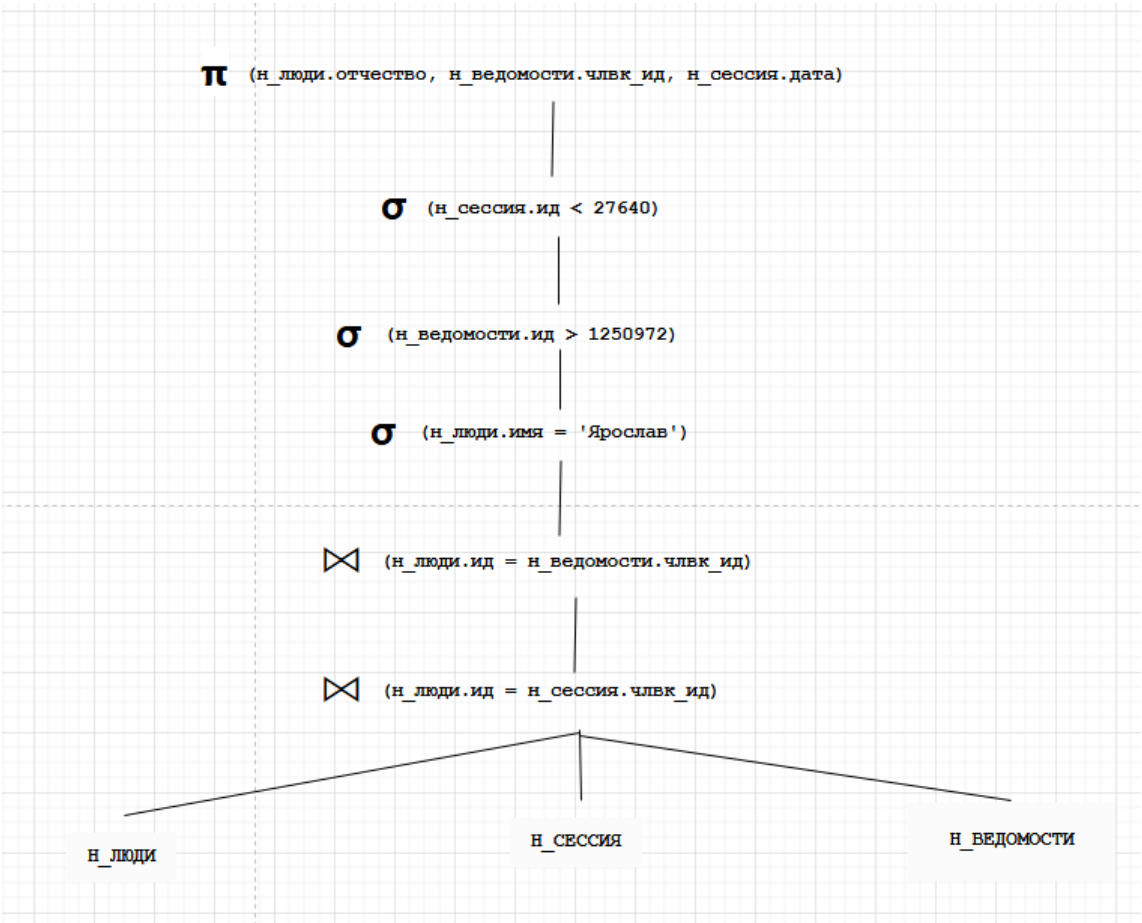
tablename	attname	n_distinct	null_frac	avg_width	correlation
Н_СЕССИЯ	ЧЛВК_ИД	180	0.13672708	4	0.13944401

indexname	indexdef
SYS_C003500_IFK	CREATE INDEX "SYS_C003500_IFK" ON public."Н_СЕССИЯ" USING btree ("ЧЛВК_ИД")
СЕС_СЭС_FK	CREATE INDEX "СЕС_СЭС_FK" ON public."Н_СЕССИЯ" USING btree ("СЭС_ИД")

Индекс уже существует, но если бы его не было, то его создание не было бы лишним – всего 180 уникальных значений, однако очень плохо отсортированный список.

2.3 План выполнения без индексов для запроса 2





Как мы видим, первый план выполнения более оптимальный, так как сначала происходит отбор по условиям, а только потом объединение таблиц (то есть объединяется меньшее количество строк)

Индексы сильно ускоряют запрос благодаря замене Seq Scan на Index Scan.

2.4 EXPLAIN ANALYZE для запроса 2

```
                                                          QUERY PLAN
-------------------------------------------------------------------------------------------------------------------------------
 Nested Loop  (cost=0.59..349.58 rows=51 width=32) (actual time=2.317..2.318 rows=0 loops=1)
   Join Filter: ("Н_ЛЮДИ"."ИД" = "Н_ВЕДОМОСТИ"."ЧЛВК_ИД")
   ->  Nested Loop  (cost=0.29..283.40 rows=10 width=36) (actual time=0.417..2.293 rows=16 loops=1)
         ->  Seq Scan on "Н_СЕССИЯ"  (cost=0.00..117.90 rows=3461 width=12) (actual time=0.009..0.672 rows=3460 loops=1)
               Filter: ("ИД" < 27640)
               Rows Removed by Filter: 292
         ->  Memoize  (cost=0.29..0.45 rows=1 width=24) (actual time=0.000..0.000 rows=0 loops=3460)
               Cache Key: "Н_СЕССИЯ"."ЧЛВК_ИД"
               Cache Mode: logical
               Hits: 3285  Misses: 175  Evictions: 0  Overflows: 0  Memory Usage: 12kB
               ->  Index Scan using "ЧЛВК_РК" on "Н_ЛЮДИ"  (cost=0.28..0.44 rows=1 width=24) (actual time=0.002..0.002 rows=0 loops=175)
                     Index Cond: ("ИД" = "Н_СЕССИЯ"."ЧЛВК_ИД")
                     Filter: (("ИМЯ")::text = 'Ярослав'::text)
                     Rows Removed by Filter: 1
   ->  Index Scan using "ВЕД_ЧЛВК_FK_IFK" on "Н_ВЕДОМОСТИ"  (cost=0.29..6.52 rows=8 width=4) (actual time=0.001..0.001 rows=0 loops=16)
         Index Cond: ("ЧЛВК_ИД" = "Н_СЕССИЯ"."ЧЛВК_ИД")
         Filter: ("ИД" > 1250972)
 Planning Time: 0.998 ms
 Execution Time: 2.396 ms
(19 строк)
```

Запрос использует Nested Loop для соединения таблиц "Н_ЛЮДИ" и "Н_ВЕДОМОСТИ" по условию "Н_ЛЮДИ"."ИД" = "Н_ВЕДОМОСТИ"."ЧЛВК_ИД", но не находит подходящих строк (rows=0). Внутри этого соединения выполняется еще один Nested Loop между таблицами "Н_СЕССИЯ" и "Н_ЛЮДИ".

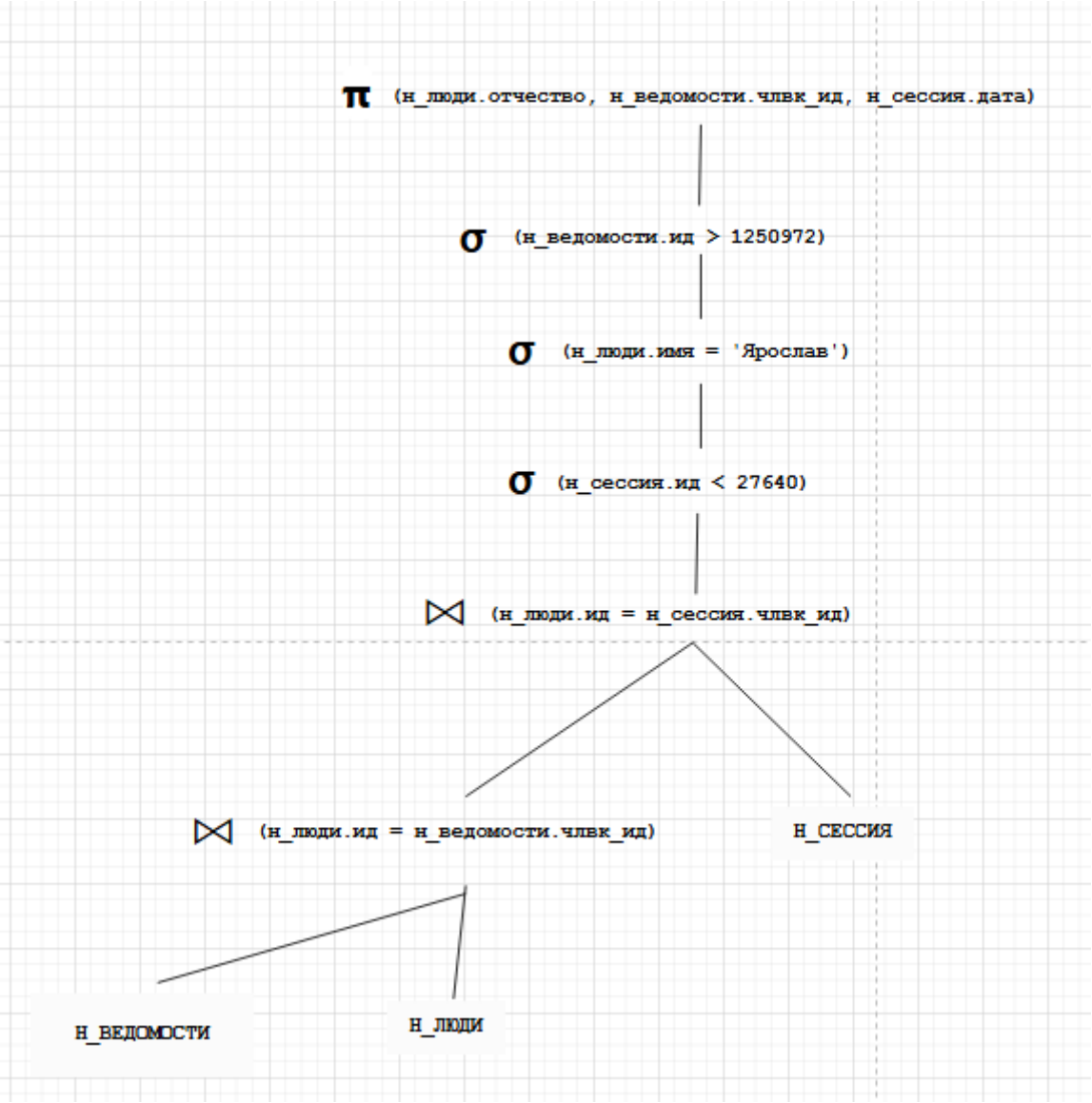
Сначала происходит Seq Scan таблицы "Н_СЕССИЯ" с фильтром "ИД" < 27640, который обрабатывает 3460 строк и отбрасывает 292.

Для каждой из этих строк выполняется поиск в "Н_ЛЮДИ" по индексу "ЧЛВК_РК" с кешированием (Memoize), где проверяется условие "ИД" = "Н_СЕССИЯ"."ЧЛВК_ИД" и фильтр "ИМЯ" = 'Ярослав'.

Кеш срабатывает 3285 раз, а 175 раз происходит обращение к индексу, но ни одна строка не проходит фильтр.

Затем для каждой из 16 строк из первого соединения выполняется Index Scan по индексу "ВЕД_ЧЛВК_FK_IFK" таблицы "Н_ВЕДОМОСТИ" с условием "ЧЛВК_ИД" = "Н_СЕССИЯ"."ЧЛВК_ИД" и фильтром "ИД" > 1250972, но также не находится подходящих строк.

Общее время выполнения запроса составляет 2.535 мс, время планирования — 1.126 мс.



Вывод:

Во время выполнения данной лабораторной работы я разобрался, что такое индексы, научился их писать и применять. Понял, как можно и нужно оптимизировать sql-запросы.