МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ

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

Факультет программной инженерии и компьютерной техники

ЛАБОРАТОРНАЯ РАБОТА №4

по дисциплине

“Информационные системы и базы данных”

Вариант № 312446

Выполнил: Воробьев Кирилл

Группа: P33302

Преподаватель: Шешуков Дмитрий
Михайлович

Санкт-Петербург, 2022

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]

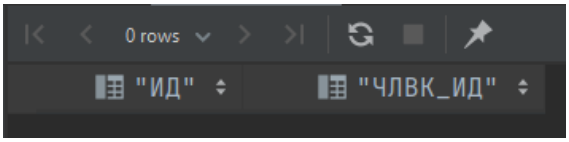
Подробные ответы на все вышеперечисленные вопросы должны присутствовать в отчете (планы выполнения запросов должны быть нарисованы, ответы на вопросы - представлены в текстовом виде).

1. Сделать запрос для получения атрибутов из указанных таблиц, применив фильтры по указанным условиям:
   Н_ТИПЫ_ВЕДОМОСТЕЙ, Н_ВЕДОМОСТИ.
   Вывести атрибуты: Н_ТИПЫ_ВЕДОМОСТЕЙ.ИД, Н_ВЕДОМОСТИ.ЧЛВК_ИД.
   Фильтры (AND):
   a) Н_ТИПЫ_ВЕДОМОСТЕЙ.НАИМЕНОВАНИЕ > Экзаменационный лист.
   b) Н_ВЕДОМОСТИ.ИД > 1490007.
   Вид соединения: RIGHT 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.ИД.
   Фильтры (AND):
   a) Н_ЛЮДИ.ИМЯ = Роман.
   b) Н_ВЕДОМОСТИ.ИД < 1457443.
   c) Н_СЕССИЯ.ИД > 27640.
   Вид соединения: RIGHT JOIN.

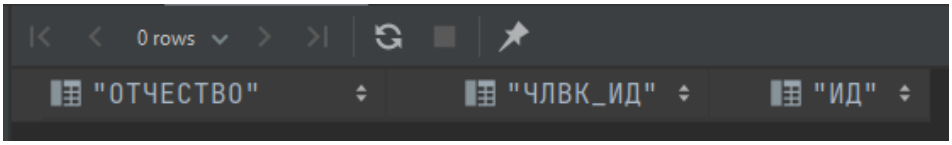
Выполнение

Запросы

1. SELECT "Н_ТИПЫ_ВЕДОМОСТЕЙ"."ИД", "Н_ВЕДОМОСТИ"."ЧЛВК_ИД" FROM "Н_ТИПЫ_ВЕДОМОСТЕЙ" RIGHT JOIN "Н_ВЕДОМОСТИ" ON "Н_ТИПЫ_ВЕДОМОСТЕЙ"."ИД" = "Н_ВЕДОМОСТИ"."ТВ_ИД" WHERE "Н_ТИПЫ_ВЕДОМОСТЕЙ"."НАИМЕНОВАНИЕ" > 'Экзаменационный лист' AND "Н_ВЕДОМОСТИ"."ИД" > 1490007



2. SELECT "Н_ЛЮДИ"."ОТЧЕСТВО", "Н_ВЕДОМОСТИ"."ЧЛВК_ИД", "Н_СЕССИЯ"."ИД" FROM "Н_ЛЮДИ" RIGHT JOIN "Н_ВЕДОМОСТИ" ON "Н_ЛЮДИ"."ИД" = "Н_ВЕДОМОСТИ"."ЧЛВК_ИД" RIGHT JOIN "Н_СЕССИЯ" ON "Н_ЛЮДИ"."ИД" = "Н_СЕССИЯ"."ЧЛВК_ИД" WHERE "Н_ЛЮДИ"."ИМЯ" = 'Роман' AND "Н_ВЕДОМОСТИ"."ИД" < 1457443 AND "Н_СЕССИЯ"."ИД" > 27640



Индексы

1. CREATE INDEX statements_id ON "Н_ВЕДОМОСТИ" USING btree ("ИД")

   Я считаю, что для оптимизации выполнения первого запроса имеет смысл использовать B-tree индекс для атрибута "ИД" таблицы "Н_ВЕДОМОСТИ", так как в запросе у нас производится сравнение «меньше». Для условия со сравнением "Н_ТИПЫ_ВЕДОМОСТЕЙ"."НАИМЕНОВАНИЕ" считаю не имеет смысла использовать индекс, поскольку возможных значений всего 3 и производится сравнение «больше»

2. CREATE INDEX session_id ON "Н_СЕССИЯ" USING btree ("ИД")
   CREATE INDEX statements_id ON "Н_ВЕДОМОСТИ" USING btree("ИД")
   CREATE INDEX person_name ON "Н_ЛЮДИ" USING hash("ИМЯ")

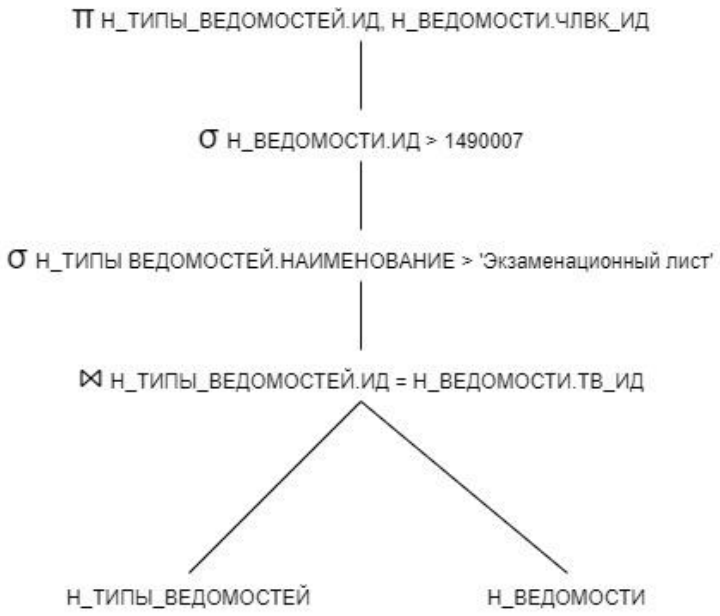
   Я считаю, что для оптимизации выполнения второго запроса имеет смысл использовать B-tree индекс для атрибута "ИД" таблицы "Н_СЕССИЯ", так как в запросе у нас производится сравнение «больше». Также, данный индекс имеет смысл использовать для атрибута "ИД" таблицы "Н_ВЕДОМОСТИ", так как в запросе у нас производится сравнение «меньше». А для проверки

третьего условия имеет смысл использовать Hash индекс, так как производится проверка равенства строк и в этой ситуации данный индекс будет отрабатывать эффективнее.

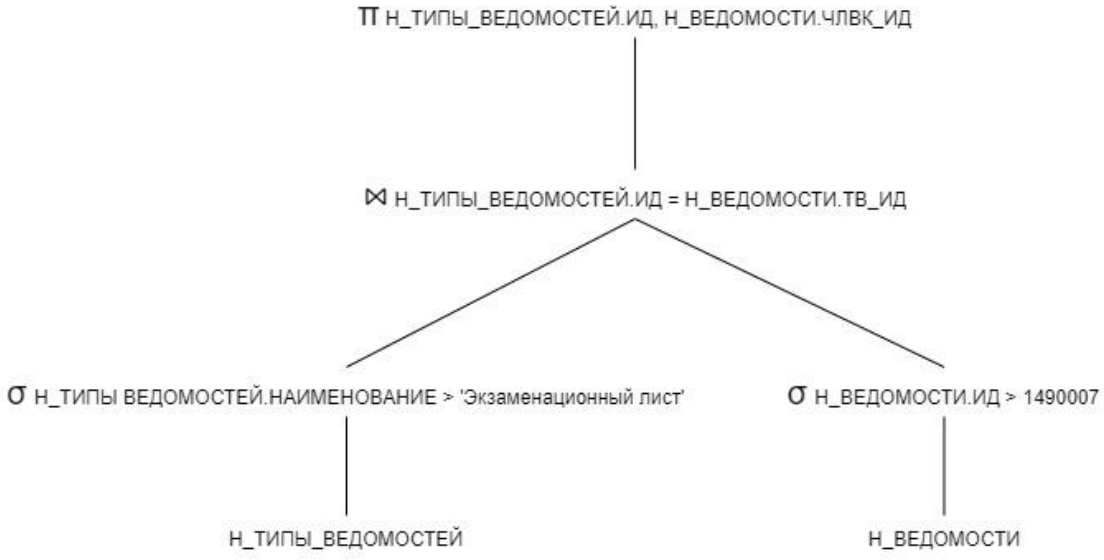
Возможные планы выполнения запросов

Первый запрос

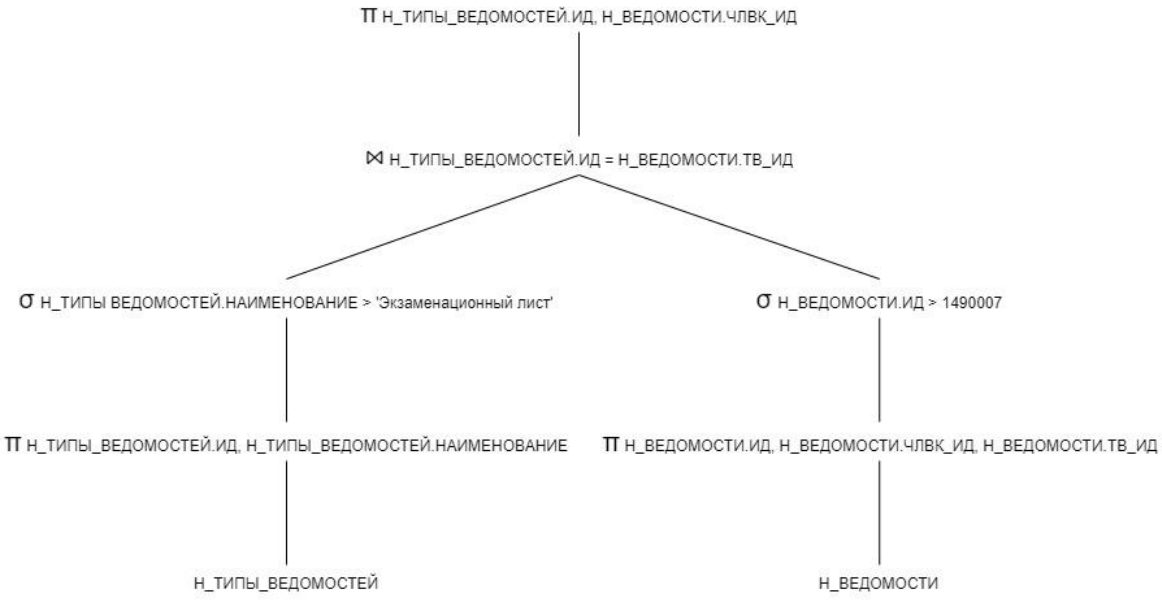
1. Соединение двух отношений, после которого производятся операции выборки по первому и второму предикатам. Результатом является проекция двух атрибутов.



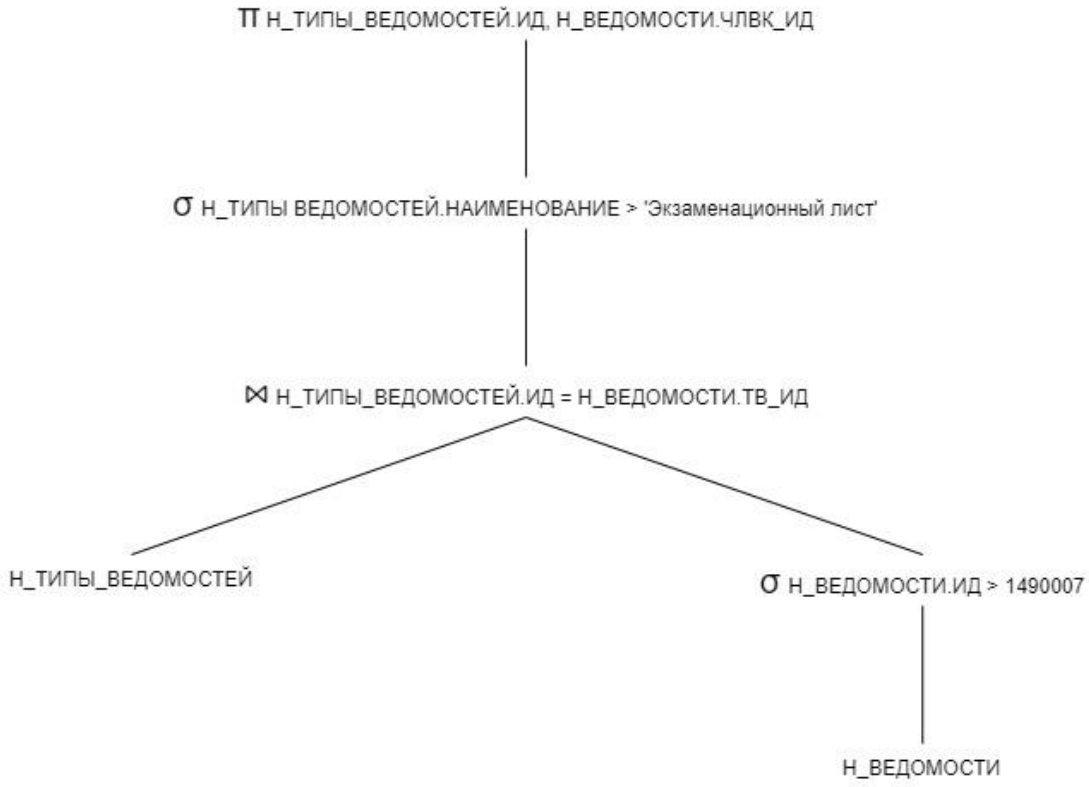
2. Изначально производятся операции выборки, далее соединение двух отношений. Результатом является проекция двух атрибутов.



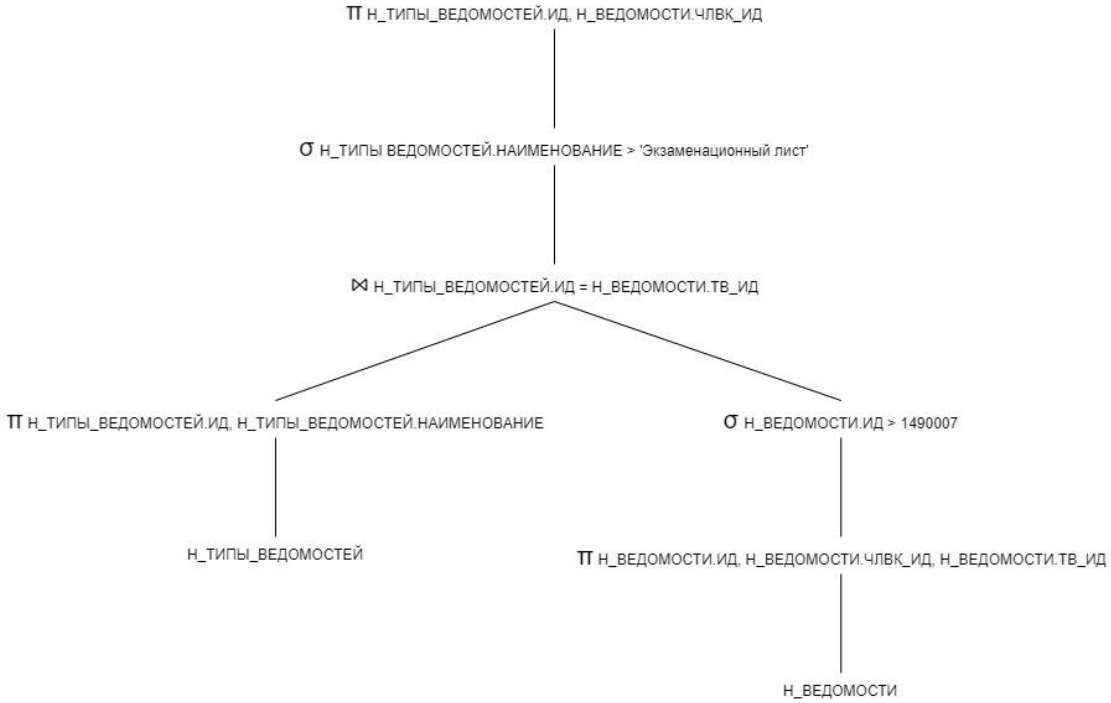
3. Для начала получается проекция для двух отношений из атрибутов, необходимых для выборки, соединения и итоговой проекции, далее производится выборка и соединение. Результатом является проекция двух атрибутов.



4. Для начала производится выборка для одного из отношений, далее соединение двух отношений, после чего производится выборка по результату соединения. Результатом является проекция двух атрибутов



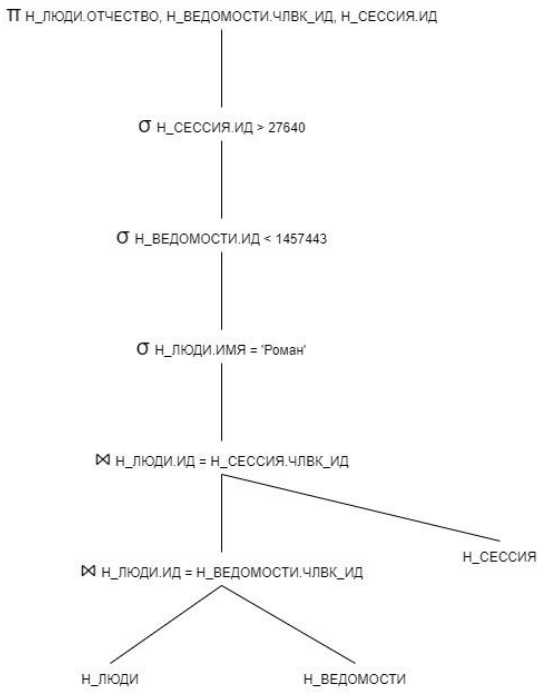
5. Для начала получается проекция для двух отношений из атрибутов, необходимых для выборки, соединения и итоговой проекции, далее производится выборка для одного из отношений, а после соединение двух отношений, после которого производится выборка для результата соединения. Результатом является проекция двух атрибутов.



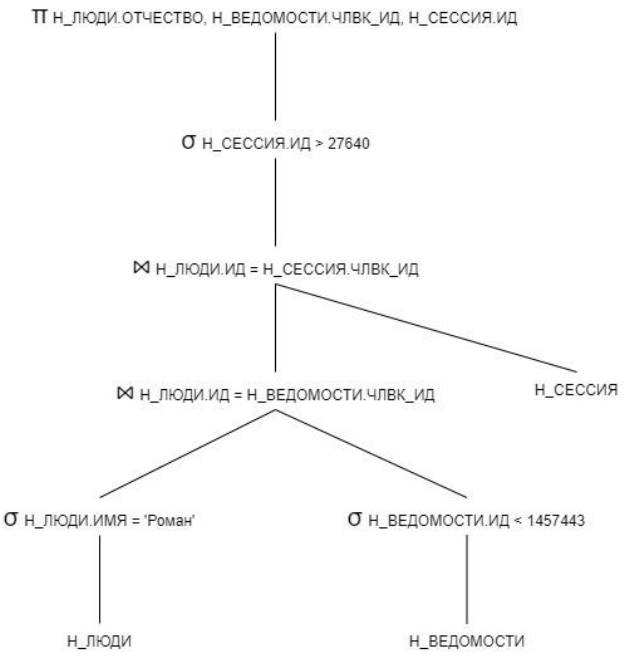
Из составленных возможных планов выполнения запроса лучшим является третий, поскольку в нем изначально получаются проекции и производятся выборки, а уже после этого выполняется соединение. Это позволяет уменьшить размер хранимых данных.

Второй запрос

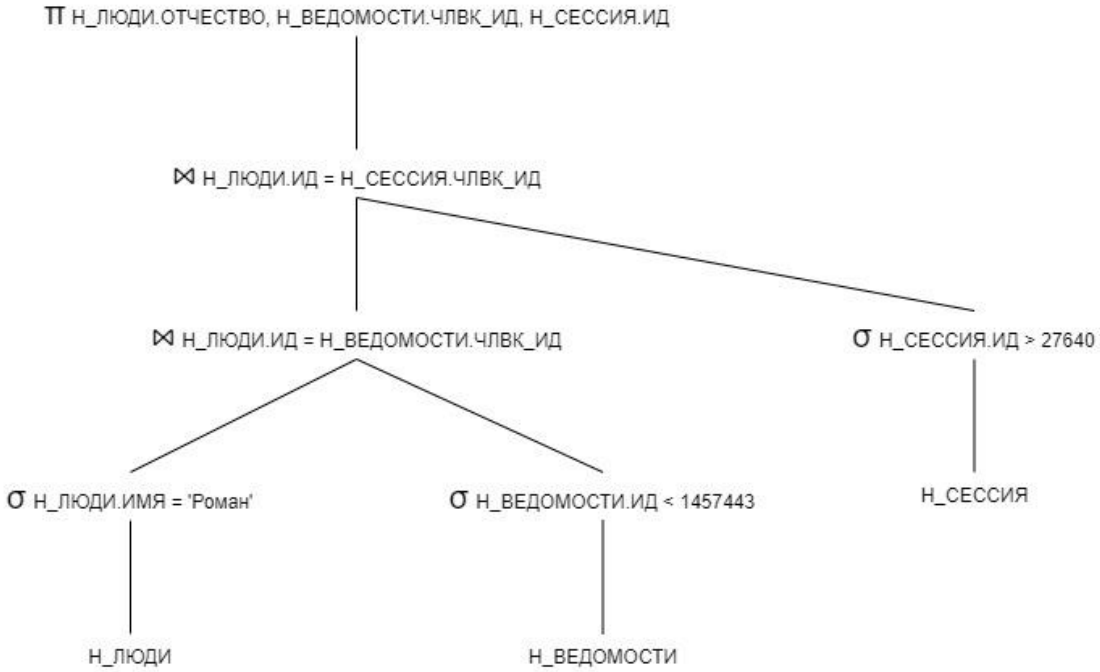
1. Соединение двух отношений, далее результат соединения соединяется с третьим отношением, а для результата двух соединений последовательно производится выборка. Результатом является проекция трех атрибутов.



2. Для начала производится выборка для первых двух отношений, далее производится соединение этих двух отношений, а после результат соединения соединяется с третьим отношением. По полученному соединению производится выборка. Результатом является проекция трех атрибутов.

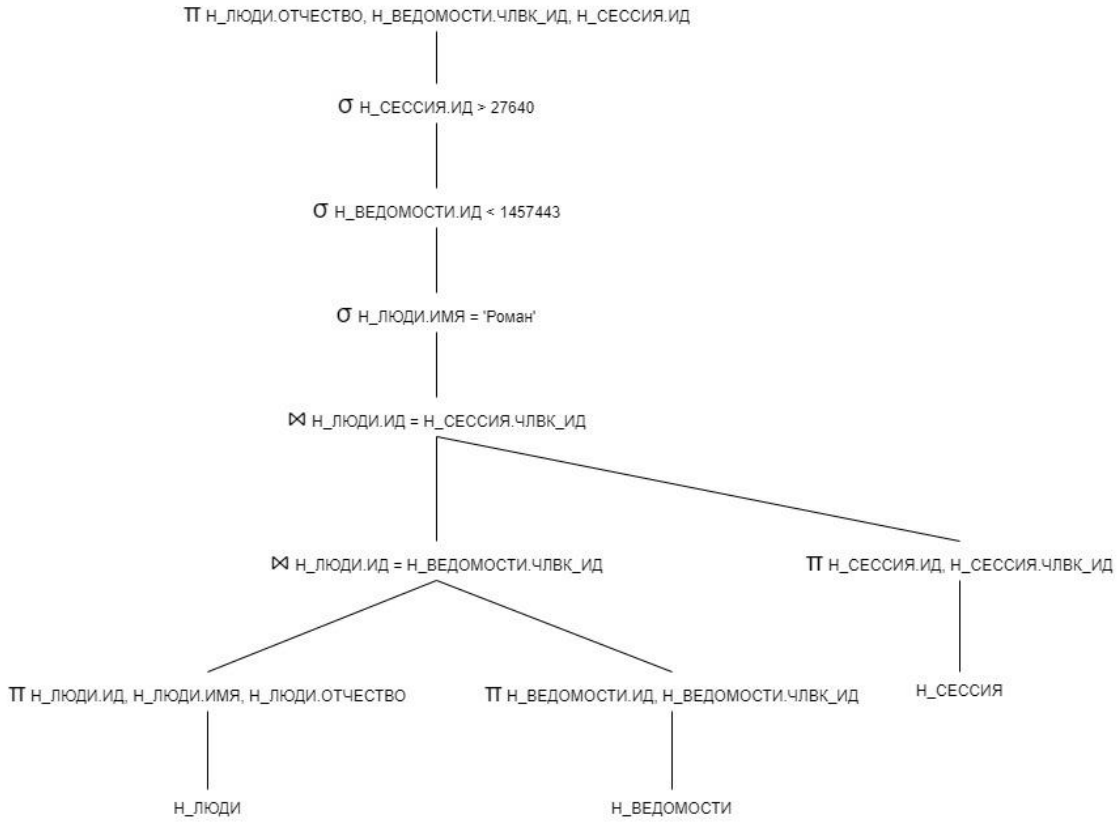


3. Для начала производится выборка для двух отношений, после соединение этих двух отношений, а далее соединение с третьим отношением, по которому перед соединением была произведена выборка. Результатом является проекция трех атрибутов.



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 и произведена выборка. Результатом является проекция трех атрибутов.

5. 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 трех атрибутов.



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

Команда EXPLAIN ANALYZE

1. EXPLAIN ANALYZE SELECT "Н_ТИПЫ_ВЕДОМОСТЕЙ"."ИД", "Н_ВЕДОМОСТИ"."ЧЛВК_ИД" FROM "Н_ТИПЫ_ВЕДОМОСТЕЙ" RIGHT JOIN "Н_ВЕДОМОСТИ" ON "Н_ТИПЫ_ВЕДОМОСТЕЙ"."ИД" = "Н_ВЕДОМОСТИ"."ТВ_ИД" WHERE "Н_ТИПЫ_ВЕДОМОСТЕЙ"."НАИМЕНОВАНИЕ" > 'Экзаменационный лист' AND "Н_ВЕДОМОСТИ"."ИД" > 1490007

```
QUERY PLAN
Nested Loop  (cost=0.42..8.30 rows=1 width=8) (actual time=0.026..0.027 rows=0 loops=1)
  Join Filter: ("Н_ТИПЫ_ВЕДОМОСТЕЙ"."ИД" = "Н_ВЕДОМОСТИ"."ТВ_ИД")
  ->  Seq Scan on "Н_ТИПЫ_ВЕДОМОСТЕЙ"  (cost=0.00..1.04 rows=1 width=4) (actual time=0.025..0.026 rows=0 loops=1)
        Filter: (("НАИМЕНОВАНИЕ")::text > 'Экзаменационный лист'::text)
        Rows Removed by Filter: 3
  ->  Index Scan using "ВЕД_РК" on "Н_ВЕДОМОСТИ"  (cost=0.42..7.25 rows=1 width=8) (never executed)
        Index Cond: ("ИД" > 1490007)
Planning Time: 0.315 ms
Execution Time: 0.077 ms
```

2. EXPLAIN ANALYZE SELECT "Н_ЛЮДИ"."ОТЧЕСТВО", "Н_ВЕДОМОСТИ"."ЧЛВК_ИД", "Н_СЕССИЯ"."ИД" FROM "Н_ЛЮДИ" RIGHT JOIN "Н_ВЕДОМОСТИ" ON "Н_ЛЮДИ"."ИД" = "Н_ВЕДОМОСТИ"."ЧЛВК_ИД" RIGHT JOIN "Н_СЕССИЯ" ON "Н_ЛЮДИ"."ИД" = "Н_СЕССИЯ"."ЧЛВК_ИД" WHERE "Н_ЛЮДИ"."ИМЯ" = 'Роман' AND "Н_ВЕДОМОСТИ"."ИД" < 1457443 AND "Н_СЕССИЯ"."ИД" > 27640

```
QUERY PLAN
Nested Loop  (cost=165.37..476.42 rows=184 width=28) (actual time=1.332..1.334 rows=0 loops=1)
  Join Filter: ("Н_ЛЮДИ"."ИД" = "Н_ВЕДОМОСТИ"."ЧЛВК_ИД")
  ->  Hash Join  (cost=165.07..283.74 rows=4 width=32) (actual time=1.241..1.321 rows=2 loops=1)
        Hash Cond: ("Н_СЕССИЯ"."ЧЛВК_ИД" = "Н_ЛЮДИ"."ИД")
        ->  Seq Scan on "Н_СЕССИЯ"  (cost=0.00..117.90 rows=290 width=8) (actual time=0.066..0.494 rows=291 loops=1)
              Filter: ("ИД" > 27640)
              Rows Removed by Filter: 3461
        ->  Hash  (cost=163.97..163.97 rows=88 width=24) (actual time=0.774..0.774 rows=88 loops=1)
              Buckets: 1024  Batches: 1  Memory Usage: 14kB
              ->  Seq Scan on "Н_ЛЮДИ"  (cost=0.00..163.97 rows=88 width=24) (actual time=0.011..0.753 rows=88 loops=1)
                    Filter: (("ИМЯ")::text = 'Роман'::text)
                    Rows Removed by Filter: 5030
  ->  Index Scan using "ВЕД_ЧЛВК_FK_IFK" on "Н_ВЕДОМОСТИ"  (cost=0.29..47.33 rows=67 width=4) (actual time=0.004..0.005 rows=0 loops=2)
        Index Cond: ("ЧЛВК_ИД" = "Н_СЕССИЯ"."ЧЛВК_ИД")
        Filter: ("ИД" < 1457443)
Planning Time: 0.673 ms
Execution Time: 1.376 ms
```

Вывод

В ходе выполнения данной лабораторной работы я познакомился с индексами, тем как они влияют на нагрузку на систему. Также я познакомился с планом выполнения запроса, узнал каким образом СУБД выбирает оптимальный. И еще узнал что выполняет команда EXPLAIN ANALYZE.