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 базам данных

Выполнила: Гафурова Фарангиз

Фуркатовна

Группа: P3120

Принял: Николаев Владимир Вячеславович

г. Санкт-Петербург, 2024г

1. **Текст задания**

   Вариант: 9325

   1. Составить запросы на языке SQL (пункты 1–2).
   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).
   3. 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.
   4. Изменятся ли планы при добавлении индекса и как?
   5. Для запросов 1–2 необходимо добавить в отчет вывод команды EXPLAIN ANALYZE [запрос]
   6. Подробные ответы на все вышеперечисленные вопросы должны присутствовать в отчете (планы выполнения запросов должны быть нарисованы, ответы на вопросы - представлены в текстовом виде).

Сделать запрос для получения атрибутов из указанных таблиц, применив фильтры по указанным условиям:

- Таблицы: H_ЛЮДИ, H_СЕССИЯ.
- Вывести атрибуты: H_ЛЮДИ.ИД, H_СЕССИЯ.УЧГОД.
- Фильтры (AND):
    - H_ЛЮДИ.ИМЯ = Александр.
    - H_СЕССИЯ.ЧЛВК_ИД > 105948q.
    - H_СЕССИЯ.ЧЛВК_ИД = 100012.
- Вид соединения: RIGHT JOIN.

Сделать запрос для получения атрибутов из указанных таблиц, применив фильтры по указанным условиям:

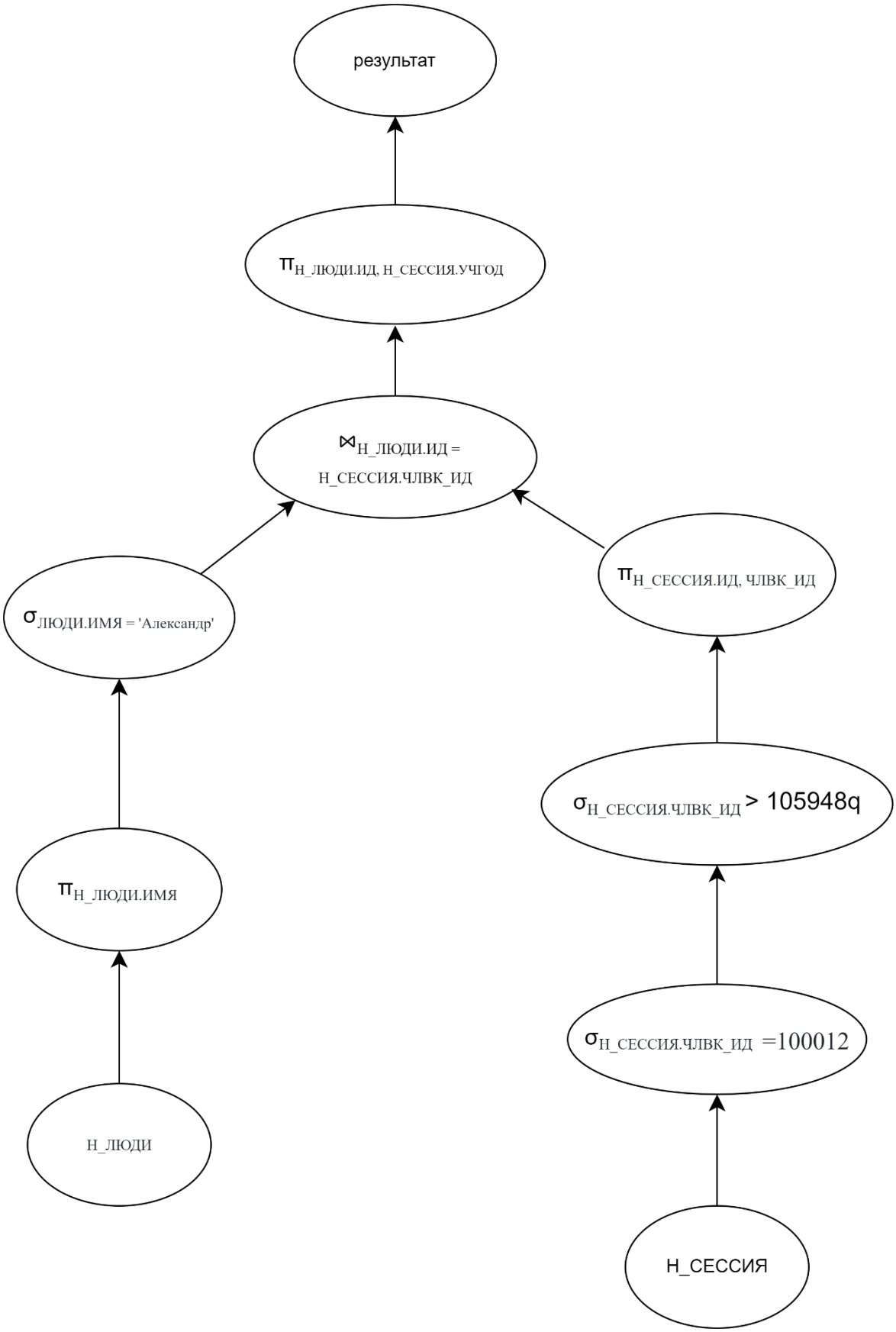
- Таблицы: H_ЛЮДИ, H_ОБУЧЕНИЯ, H_УЧЕНИКИ.
- Вывести атрибуты: H_ЛЮДИ.ОТЧЕСТВО, H_ОБУЧЕНИЯ.ЧЛВК_ИД, H_УЧЕНИКИ.ГРУППА.
- Фильтры: (AND)
    - H_ЛЮДИ.ИД < 142095.
    - H_ОБУЧЕНИЯ.ЧЛВК_ИД < 163276.
- Вид соединения: INNER JOIN.

2. **Реализация первого запроса**

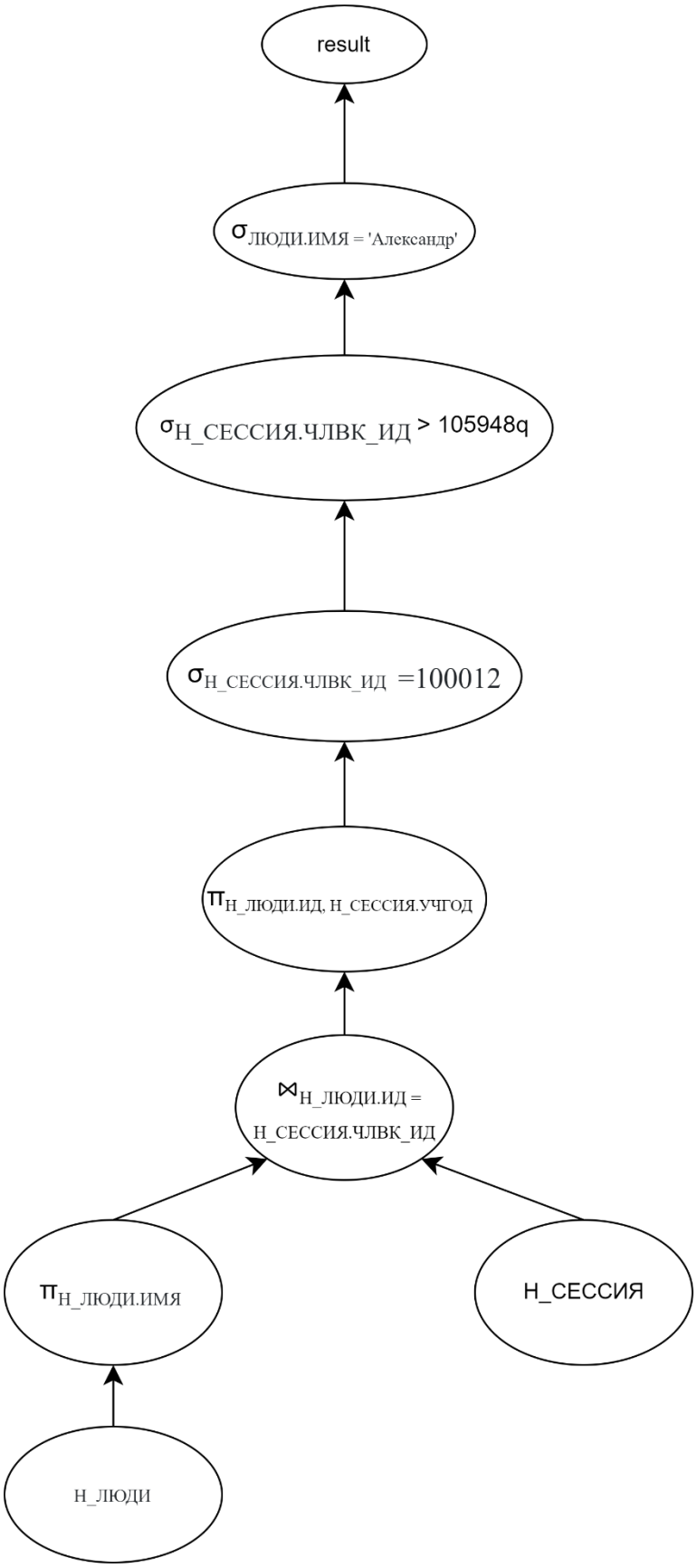
```
EXPLAIN (ANALYZE) SELECT H_ЛЮДИ.ИД AS "id человека", H_СЕССИЯ.УЧГОД
AS "учебный год"
FROM H_ЛЮДИ
RIGHT JOIN H_СЕССИЯ ON H_ЛЮДИ.ИД = H_СЕССИЯ.ЧЛВК_ИД
WHERE H_ЛЮДИ.ИМЯ = 'Александр'
AND H_СЕССИЯ.ЧЛВК_ИД > 105948
AND H_СЕССИЯ.ЧЛВК_ИД = 100012;
```

Планы выполнения:

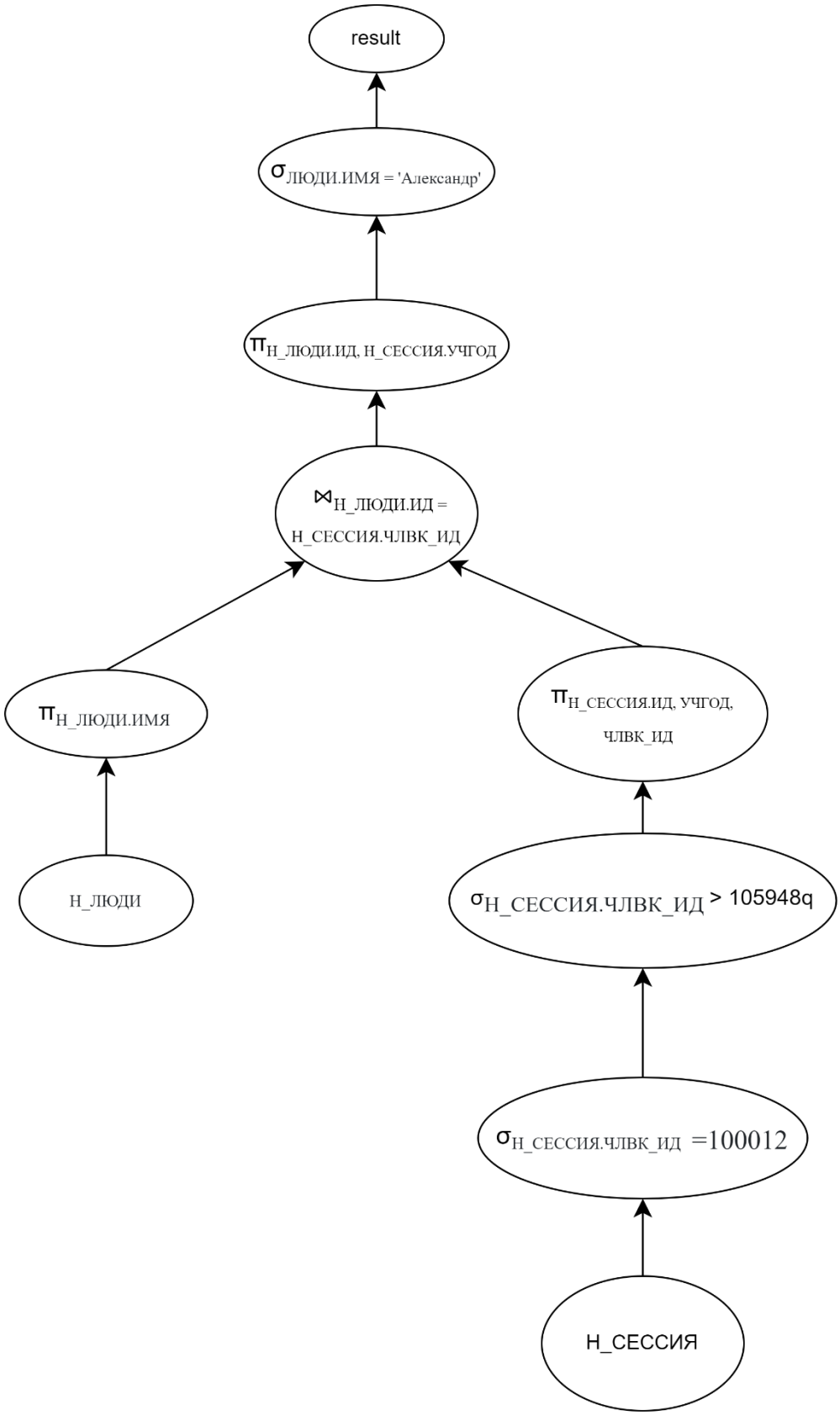
а)



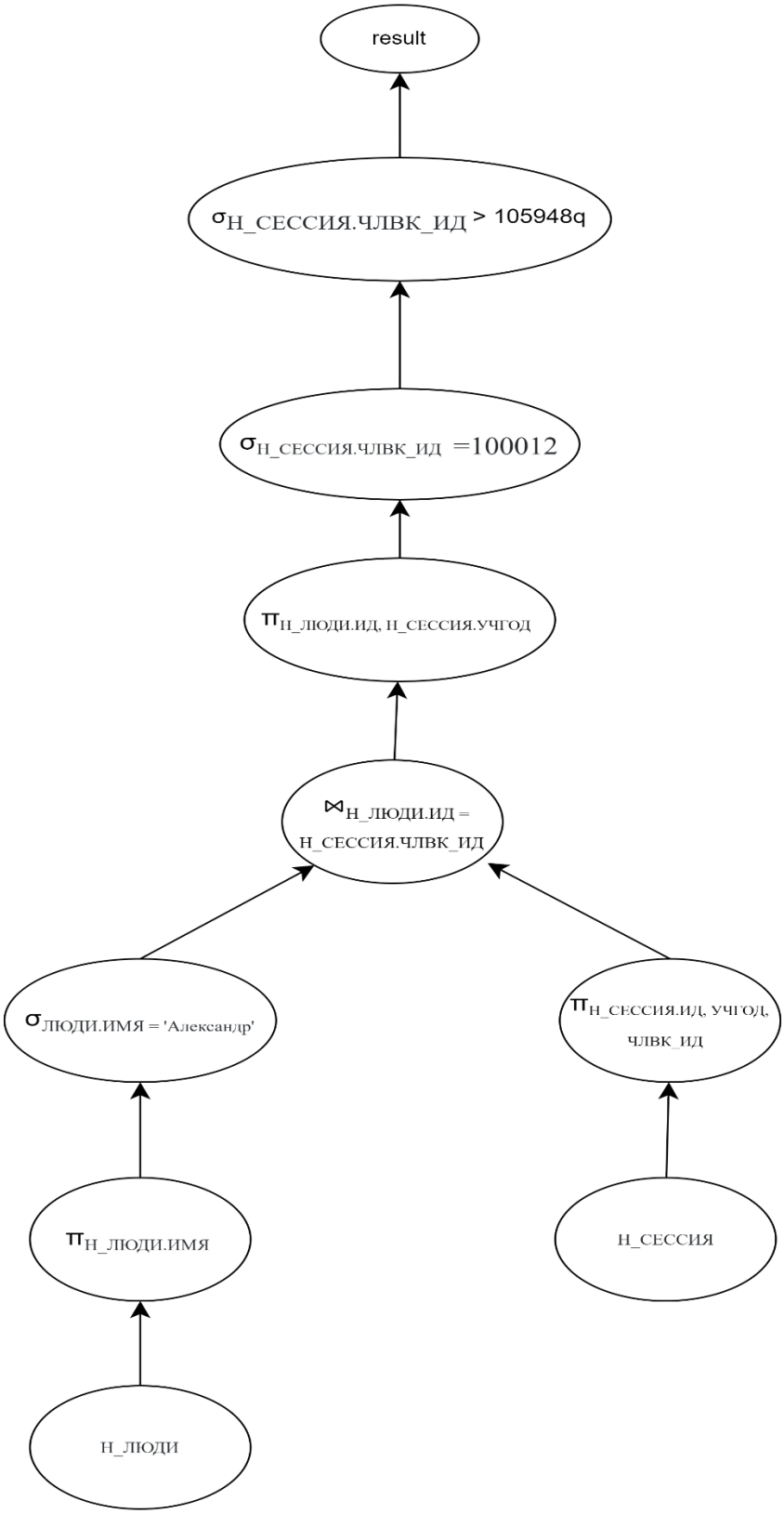
b)


result
σЛЮДИ.ИМЯ = 'Александр'
σН_СЕССИЯ.ЧЛВК_ИД > 105948q
σН_СЕССИЯ.ЧЛВК_ИД =100012
πН_ЛЮДИ.ИД, Н_СЕССИЯ.УЧГОД
⋈Н_ЛЮДИ.ИД = Н_СЕССИЯ.ЧЛВК_ИД
πН_ЛЮДИ.ИМЯ
Н_СЕССИЯ
Н_ЛЮДИ

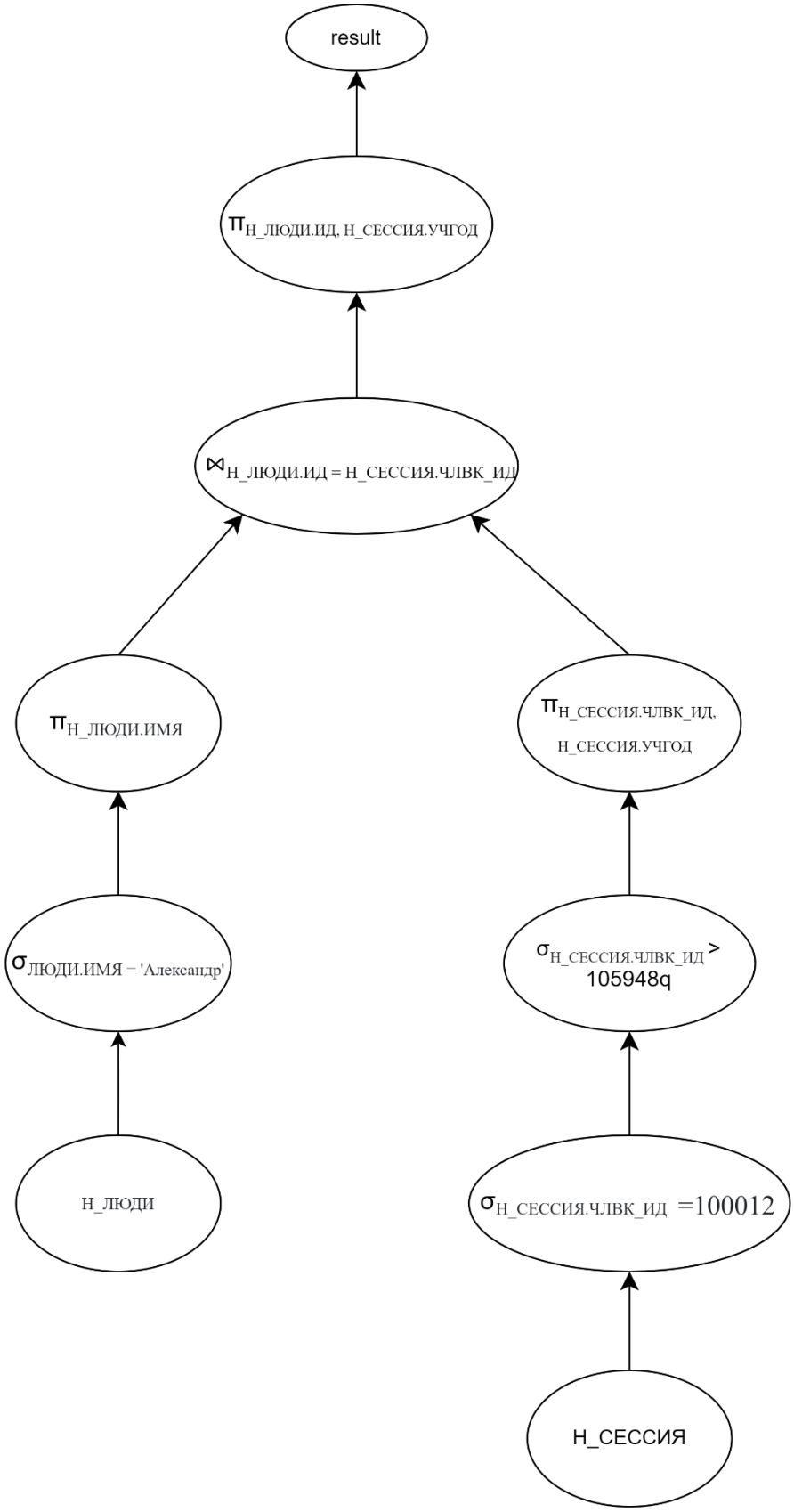
c)


result
σЛЮДИ.ИМЯ = 'Александр'
πН_ЛЮДИ.ИД, Н_СЕССИЯ.УЧГОД
⋈Н_ЛЮДИ.ИД = Н_СЕССИЯ.ЧЛВК_ИД
πН_ЛЮДИ.ИМЯ
πН_СЕССИЯ.ИД, УЧГОД, ЧЛВК_ИД
Н_ЛЮДИ
σН_СЕССИЯ.ЧЛВК_ИД > 105948q
σН_СЕССИЯ.ЧЛВК_ИД =100012
Н_СЕССИЯ

d)


result
σН_СЕССИЯ.ЧЛВК_ИД > 105948q
σН_СЕССИЯ.ЧЛВК_ИД =100012
πН_ЛЮДИ.ИД, Н_СЕССИЯ.УЧГОД
⋈Н_ЛЮДИ.ИД = Н_СЕССИЯ.ЧЛВК_ИД
σЛЮДИ.ИМЯ = 'Александр'
πН_СЕССИЯ.ИД, УЧГОД, ЧЛВК_ИД
πН_ЛЮДИ.ИМЯ
Н_СЕССИЯ
Н_ЛЮДИ

е)



По моему мнению, наилучшим вариантом является последняя схема (е), поскольку это дерево с правосторонней конвейерной обработкой данных.

Индексы:

CREATE INDEX "idx" ON "Н_ЛЮДИ" USING HASH ("ИД");
CREATE INDEX "idx_имя" ON "Н_ЛЮДИ" USING HASH ("ИМЯ");
CREATE INDEX "idx_члвк_ид" ON " Н_СЕССИЯ" USING BTREE ("ЧЛВК_ИД");

- Необходим индекс для столбца Н_СЕССИЯ.ЧЛВК_ИД (тип btree, так как выборка происходит с использованием операторов сравнения), который ускорит процесс соединения таблиц по этому столбцу.

- Возможно также создание индексов для столбца Н_ЛЮДИ.ИМЯ (тип hash-index, т. к. он будет эффективен для проверки условий на равенство) для ускорения проверки условий по этим столбцам.

Explain analyze:

```
AND Н_СЕССИЯ.ЧЛВК_ИД = 100012;
                                                    QUERY PLAN
-----------------------------------------------------------------------------------------------------------------------------
 Nested Loop  (cost=0.56..16.61 rows=1 width=14) (actual time=0.020..0.021 rows=0 loops=1)
   ->  Index Scan using "ЧЛВК_РК" on "Н_ЛЮДИ"  (cost=0.28..8.30 rows=1 width=4) (actual time=0.020..0.020 rows=0 loops=1)
         Index Cond: ("ИД" = 100012)
         Filter: (("ИМЯ")::text = 'Александр'::text)
         Rows Removed by Filter: 1
   ->  Index Scan using "SYS_C003500_IFK" on "Н_СЕССИЯ"  (cost=0.28..8.30 rows=1 width=14) (never executed)
         Index Cond: (("ЧЛВК_ИД" > 105948) AND ("ЧЛВК_ИД" = 100012))
 Planning Time: 0.144 ms
 Execution Time: 0.046 ms
(9 строк)
```

В этом запросе используется алгоритм Nested Loop Right Join.

Объяснение:
1. Сначала выполняется Index Scan по индексу idx_н_люди_ид в таблице Н_ЛЮДИ, чтобы найти записи, где ИМЯ = 'Александр'.
2. Затем для каждой найденной записи в Н_ЛЮДИ выполняется Bitmap Heap Scan по таблице Н_СЕССИЯ, чтобы найти записи, где ЧЛВК_ИД > 105948 и ЧЛВК_ИД = 100012.
3. Результаты объединяются в Right Join.

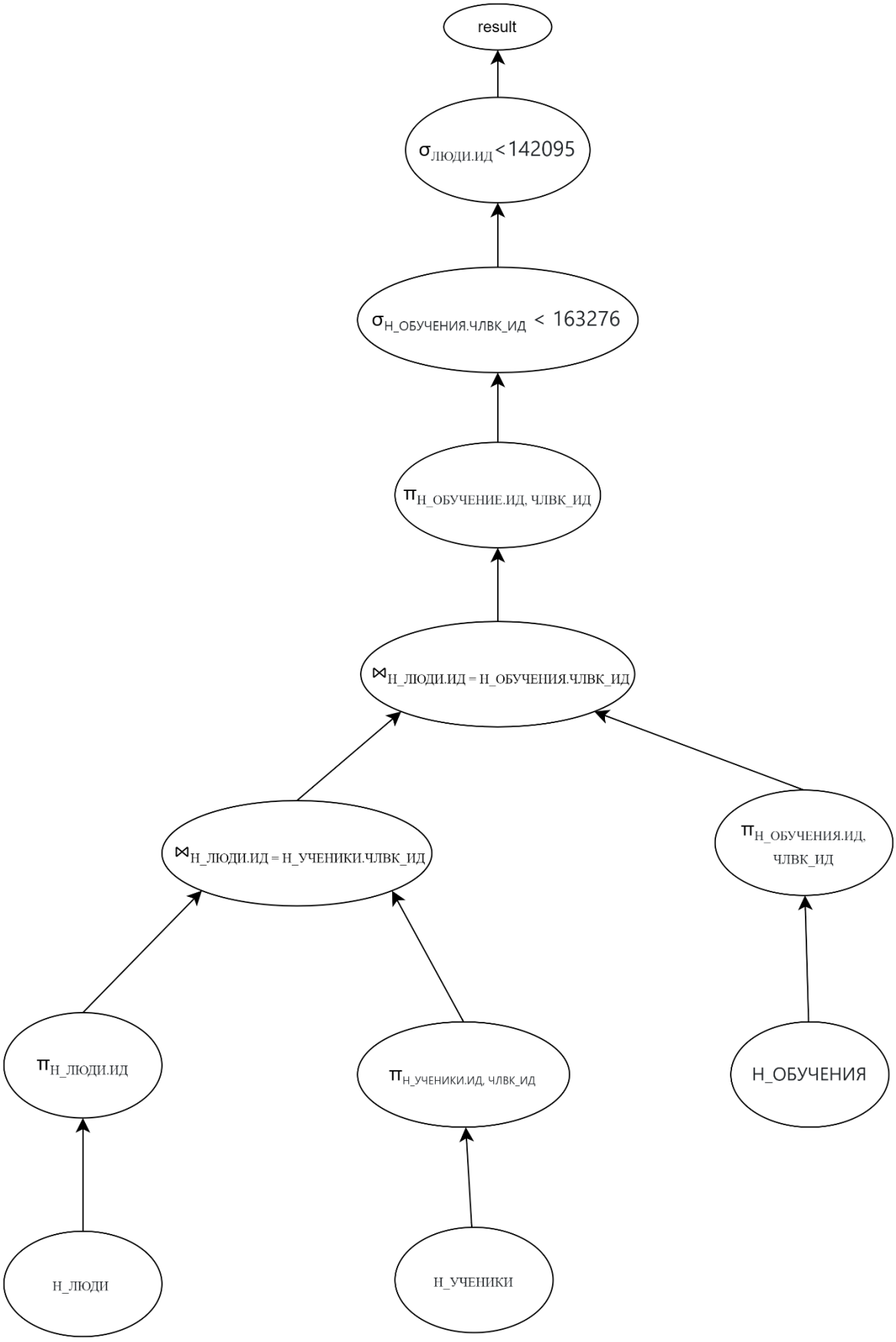
Этот алгоритм используется, потому что:
- Правая таблица (Н_СЕССИЯ) относительно небольшая, и для нее может быть применен индекс по полю ЧЛВК_ИД.
- Левая таблица (Н_ЛЮДИ) более селективна, и поиск по ней с использованием индекса idx_н_люди_ид более эффективен.

3. Реализация второго запроса

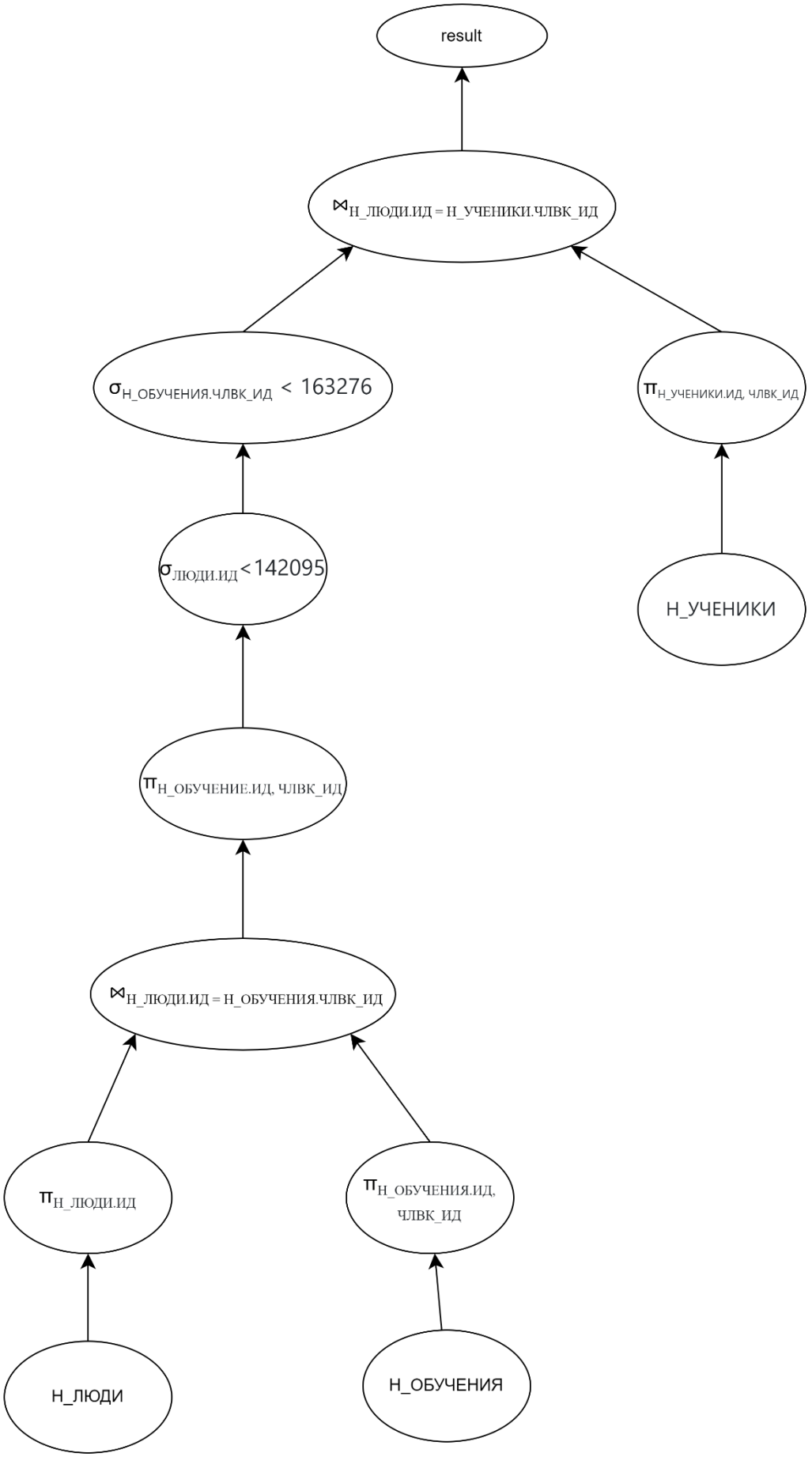
```
EXPLAIN ANALYZE SELECT
Н_ЛЮДИ.ОТЧЕСТВО AS "отчество человека",
Н_ОБУЧЕНИЯ.ЧЛВК_ИД AS "id человека",
Н_УЧЕНИКИ.ГРУППА AS "группа студента"
FROM Н_ЛЮДИ
INNER JOIN Н_ОБУЧЕНИЯ ON Н_ЛЮДИ.ИД = Н_ОБУЧЕНИЯ.ЧЛВК_ИД
INNER JOIN Н_УЧЕНИКИ ON Н_УЧЕНИКИ.ЧЛВК_ИД = Н_ОБУЧЕНИЯ.ЧЛВК_ИД
WHERE Н_ЛЮДИ.ИД < 142095
AND Н_ОБУЧЕНИЯ.ЧЛВК_ИД < 163276;
```

Планы выполнения:

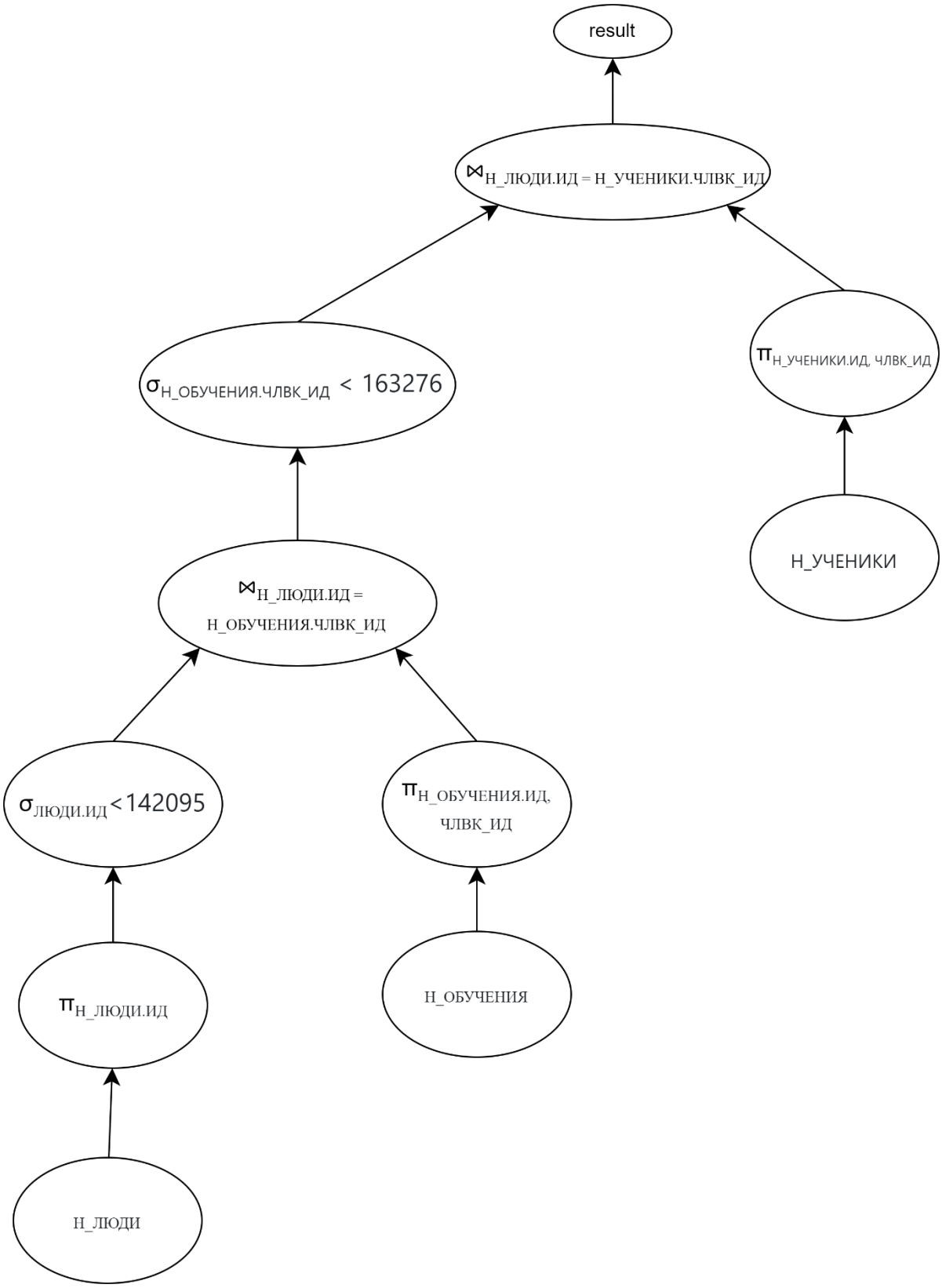
а)



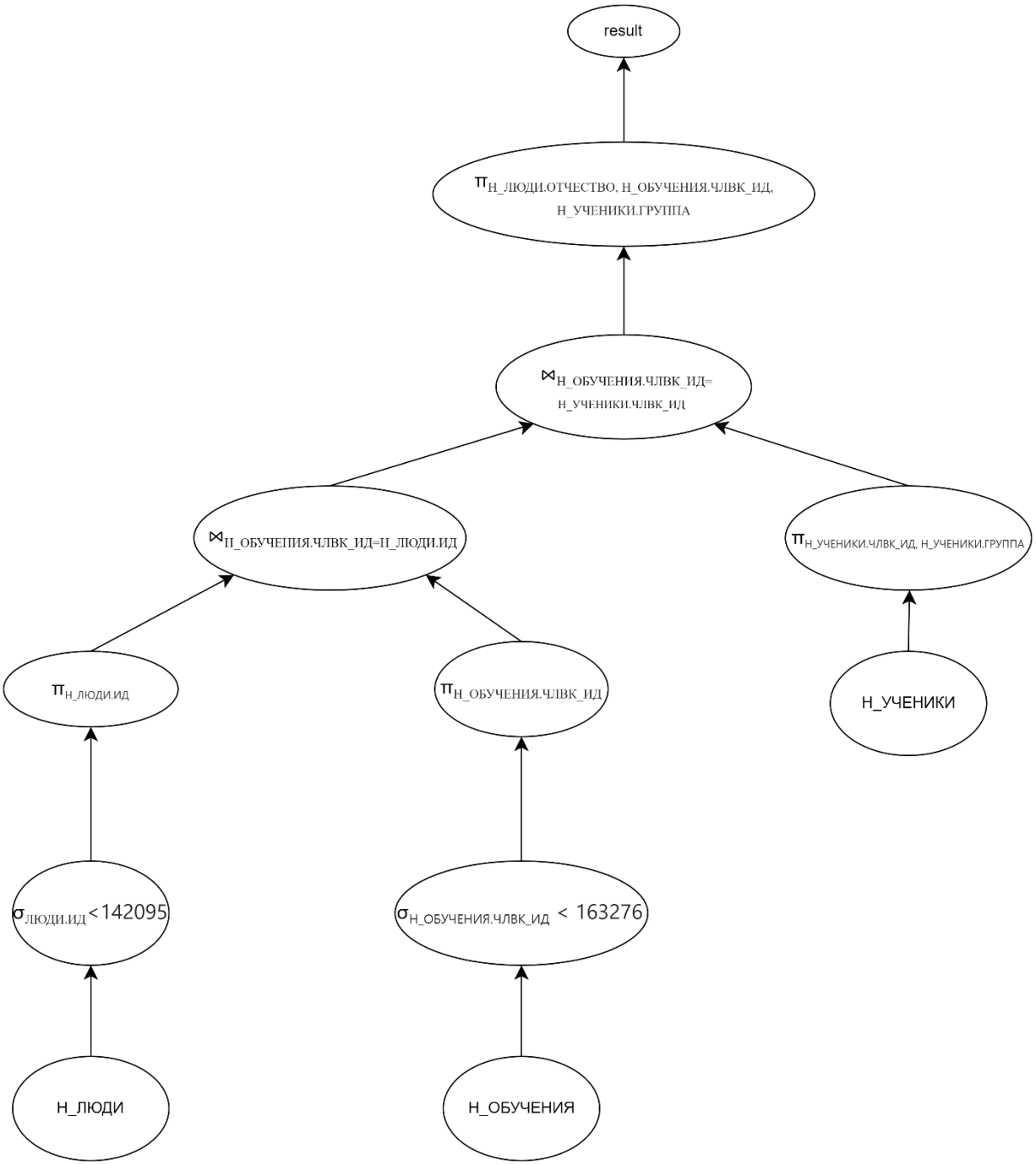
b)


result
⋈Н_ЛЮДИ.ИД = Н_УЧЕНИКИ.ЧЛВК_ИД
σН_ОБУЧЕНИЯ.ЧЛВК_ИД < 163276
πН_УЧЕНИКИ.ИД, ЧЛВК_ИД
σЛЮДИ.ИД<142095
Н_УЧЕНИКИ
πН_ОБУЧЕНИЕ.ИД, ЧЛВК_ИД
⋈Н_ЛЮДИ.ИД = Н_ОБУЧЕНИЯ.ЧЛВК_ИД
πН_ЛЮДИ.ИД
πН_ОБУЧЕНИЯ.ИД, ЧЛВК_ИД
Н_ЛЮДИ
Н_ОБУЧЕНИЯ

c)


result
⋈ Н_ЛЮДИ.ИД = Н_УЧЕНИКИ.ЧЛВК_ИД
σ Н_ОБУЧЕНИЯ.ЧЛВК_ИД < 163276
π Н_УЧЕНИКИ.ИД, ЧЛВК_ИД
Н_УЧЕНИКИ
⋈ Н_ЛЮДИ.ИД = Н_ОБУЧЕНИЯ.ЧЛВК_ИД
σ ЛЮДИ.ИД < 142095
π Н_ОБУЧЕНИЯ.ИД, ЧЛВК_ИД
π Н_ЛЮДИ.ИД
Н_ОБУЧЕНИЯ
Н_ЛЮДИ

d)



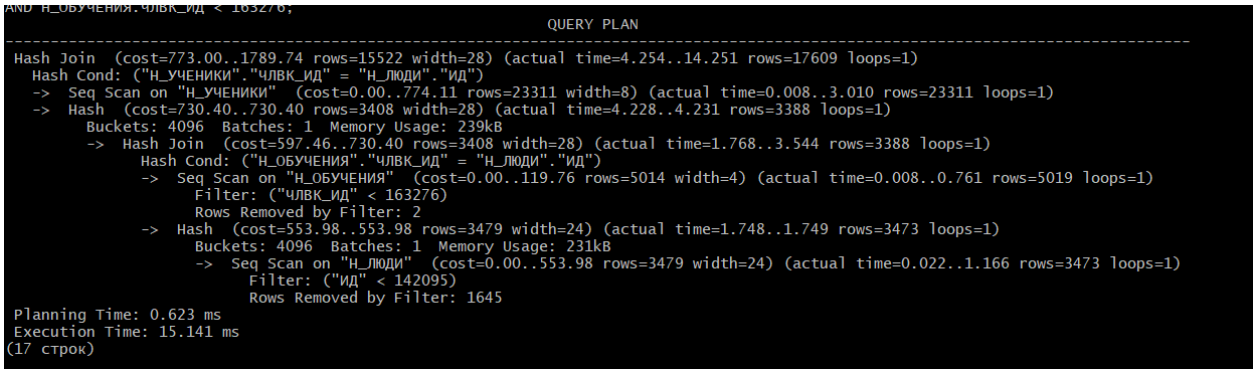
Оптимальный на мой взгляд последний план (d), потому что сначала производится выбор строк по указанным условиям, а только потом отфильтрованные строки соединяются с помощью INNER JOIN.

Индексы:

```
CREATE INDEX "idx" ON "Н_ЛЮДИ" USING BTREE ("ИД");
CREATE INDEX "idx_члвк_ид" ON "Н_ОБУЧЕНИЯ " USING BTREE ("ЧЛВК_ИД");
CREATE INDEX "idx_уч_члвк_ид" ON "Н_УЧЕНИКИ" USING BTREE ("ЧЛВК_ИД ");
```

- Выборка происходит с использованием операторов сравнения, поэтому оптимально использование BTREE.

Explain analyze:



В данном запросе используется алгоритм Hash Join для выполнения внутреннего соединения (INNER JOIN) между таблицами Н_ЛЮДИ, Н_ОБУЧЕНИЯ и Н_УЧЕНИКИ.

Hash Join — это алгоритм соединения, который использует хэш-таблицу для быстрого сопоставления строк между двумя таблицами. Он работает следующим образом:

1. Первый оператор соединения (left table) сканируется, и его ключи соединения хэшируются в памяти.
2. Второй оператор соединения (right table) сканируется, и его ключи соединения проверяются на совпадение с хэшем, построенным на первом шаге.

Почему используется Hash Join:
1. Размеры таблиц: если таблицы, участвующие в соединении, достаточно большие, то Hash Join будет более эффективным, чем другие алгоритмы соединения, такие как Nested Loop Join.
2. Типы данных: Hash Join работает хорошо, когда ключи соединения имеют простые типы данных, такие как числа или строки.
3. Отсутствие индексов: если на ключах соединения нет индексов, Hash Join может быть более эффективным, чем использование индексов.

В моем случае выбирается Hash Join, так как это эффективный способ соединить большие таблицы Н_ЛЮДИ, Н_ОБУЧЕНИЯ и Н_УЧЕНИКИ без индексов на ключах соединения.

4. **Влияние индексов на планы:**

Предложенные мной индексы не сильно повлияли бы на планы выполнения запросов: условия проверки будут проводиться по индексам атрибутов, что ускорит выполнение запроса, а для объединения таблиц Н_ЛЮДИ и Н_СЕССИЯ (и Н_ОБУЧЕНИЯ, Н_УЧЕНИКИ во втором запросе) будут использованы индексы Н_СЕССИЯ.ЧЛВК_ИД (Н_ОБУЧЕНИЯ.ЧЛВК_ИД). Сути действий это не изменит, но каждое действие будет занимать меньше времени.

5. **Отчёт:**

Ссылка на репозиторий - [karillisa/Databases/Laboratory_work_4]

6. Вывод:

В ходе выполнения лабораторной работы я разобралась в работе с реляционной алгеброй и изучила, как строить планы выполнения запросов и их диаграммы. Также я освоила различные виды индексов и поняла, как их использовать для оптимизации скорости выполнения запросов. Теперь я готова применять эти знания на практике для эффективной работы с базами данных и улучшения производительности моих SQL-запросов.