**Министерство науки и высшего образования Российской Федерации**

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧЕРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

**«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»**

**Факультет ПИиКТ**

**Дисциплина: Базы данных**

**Лабораторная работа №4**

**Нормализации и триггер**

**Вариант 4859201**

Выполнил: Михайлов Петр Сергеевич

Группа: Р3111

Преподаватель: Харитонова Анастасия Евгеньевна

Санкт-Петербург 2025г.

Содержание

[Текст задания 3](#_Toc198643384)

[Выполнение нормализации 4](#_Toc198643385)

[Исходная модель 4](#_Toc198643386)

[Функциональные зависимости 4](#_Toc198643387)

[Анализ зависимостей 4](#_Toc198643388)

[1NF 4](#_Toc198643389)

[2NF 5](#_Toc198643390)

[3NF 5](#_Toc198643391)

[BCNF 5](#_Toc198643392)

[Денормализация 6](#_Toc198643393)

[Триггер и связанная функция на языке PL/pgSQL 7](#_Toc198643394)

[Выводы по работе 9](#_Toc198643395)

# 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?

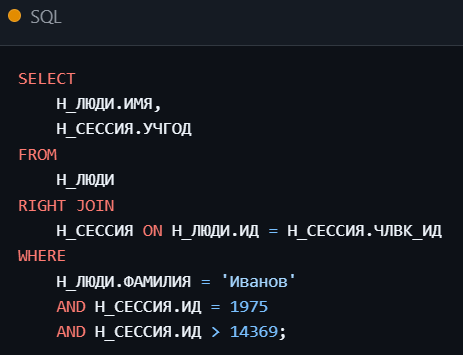
Для запросов 1-2 необходимо добавить в отчет вывод команды EXPLAIN ANALYZE [запрос]

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\_ЛЮДИ, Н\_СЕССИЯ.  
   Вывести атрибуты: Н\_ЛЮДИ.ИМЯ, Н\_СЕССИЯ.УЧГОД.  
   Фильтры (AND):  
   a) Н\_ЛЮДИ.ФАМИЛИЯ = Иванов.  
   b) Н\_СЕССИЯ.ИД = 1975.  
   c) Н\_СЕССИЯ.ИД > 14369.  
   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.ИД, Н\_ВЕДОМОСТИ.ЧЛВК\_ИД, Н\_СЕССИЯ.ИД.  
   Фильтры (AND):  
   a) Н\_ЛЮДИ.ОТЧЕСТВО < Сергеевич.  
   b) Н\_ВЕДОМОСТИ.ЧЛВК\_ИД = 153285.  
   Вид соединения: INNER JOIN.

# Запрос 1

Реализация



*Рис. 1. Реализация запроса 1 на SQL*

Возможные индексы

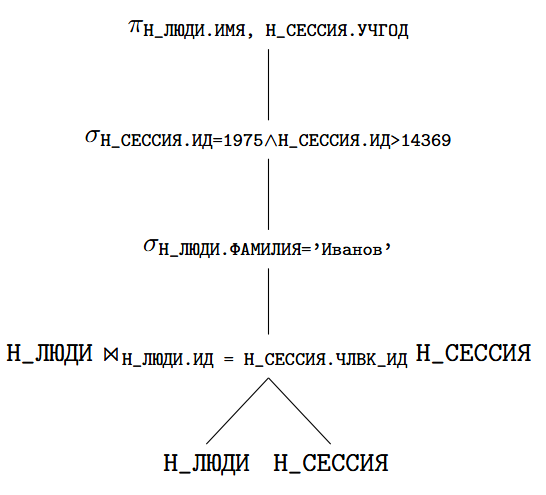
Таблица 1 – Предлагаемые индексы для атрибутов таблиц запроса 1

|  |  |  |
| --- | --- | --- |
| Тип индекса | Таблица.Атрибут | Обоснование |
| B-tree | Н\_ЛЮДИ.ФАМИЛИЯ | Ускоряет поиск по условию WHERE Н\_ЛЮДИ.ФАМИЛИЯ = 'Иванов'. Без индекса СУБД пришлось бы выполнять полный просмотр таблицы Н\_ЛЮДИ (Full Table Scan). С B-tree индексом поиск будет значительно быстрее (Index Seek/Scan), так как СУБД сможет быстро найти нужные записи. |
| B-tree | Н\_ЛЮДИ.ИД | Критически важен для операции RIGHT JOIN ... ON Н\_ЛЮДИ.ИД = Н\_СЕССИЯ.ЧЛВК\_ИД. При выполнении RIGHT JOIN (где Н\_СЕССИЯ является "правой", ведущей таблицей), СУБД для каждой строки из Н\_СЕССИЯ будет искать соответствующую строку в Н\_ЛЮДИ по полю Н\_ЛЮДИ.ИД. Индекс на Н\_ЛЮДИ.ИД превращает этот поиск из потенциально медленного (сканирование Н\_ЛЮДИ для каждой строки Н\_СЕССИЯ) в очень быстрый поиск по индексу (logN). |
| B-tree | Н\_СЕССИЯ.ИД | Ускоряет фильтрацию по условиям WHERE Н\_СЕССИЯ.ИД = 1975 и Н\_СЕССИЯ.ИД > 14369. B-tree индекс эффективен как для точечных запросов (равенство), так и для диапазонных (>). Даже если эти конкретные условия не вернут строк, индекс позволит СУБД быстро это определить. |
| B-tree | Н\_СЕССИЯ.ЧЛВК\_ИД | Также важен для операции JOIN ON Н\_ЛЮДИ.ИД = Н\_СЕССИЯ.ЧЛВК\_ИД. Этот индекс на столбце внешнего ключа в таблице Н\_СЕССИЯ улучшает производительность соединения. Он позволяет СУБД эффективно находить связанные строки при различных стратегиях выполнения JOIN (например, если оптимизатор решит использовать Merge Join или Hash Join, или если бы порядок таблиц в JOIN был другим). |

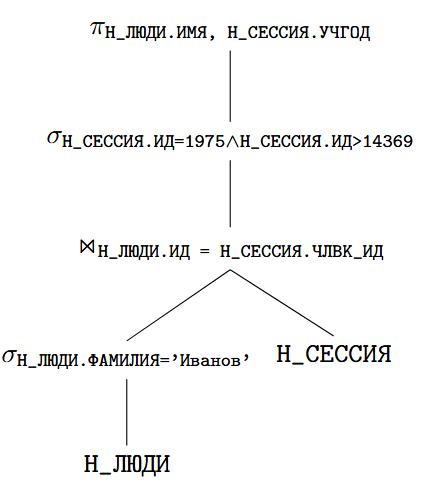
Для всех указанных атрибутов (Н\_ЛЮДИ.ФАМИЛИЯ, Н\_СЕССИЯ.ИД, Н\_ЛЮДИ.ИД, Н\_СЕССИЯ.ЧЛВК\_ИД) предпочтение отдается B-tree индексам из-за их универсальности: они эффективны как для точечных и диапазонных условий в WHERE, так и для операций JOIN. Индексы на Н\_ЛЮДИ.ИД и Н\_СЕССИЯ.ЧЛВК\_ИД критически важны для ускорения соединения таблиц. Важно отметить, что использование B-tree индексов на диске не мешает СУБД применять различные алгоритмы соединения, включая Hash Join, путем построения временных хэш-таблиц в памяти, если оптимизатор сочтет это более эффективным.

Возможные планы запроса

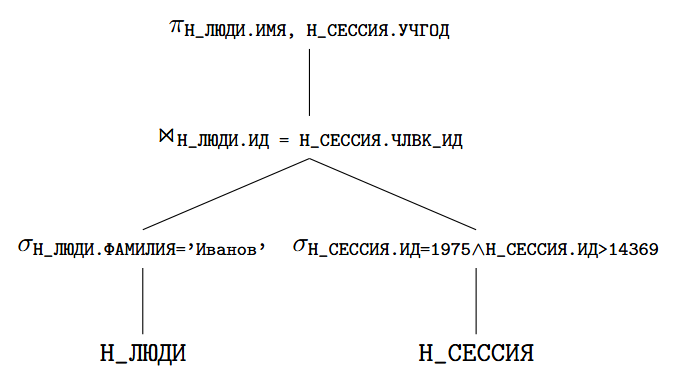
План 1



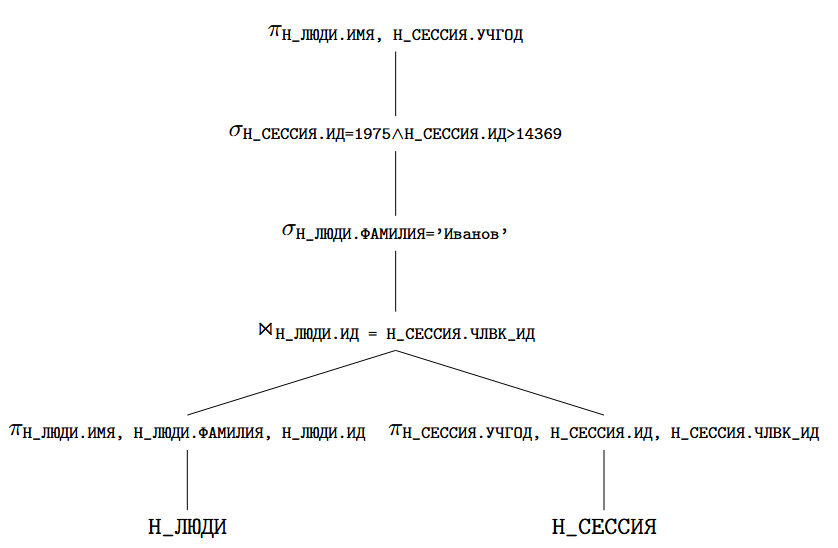
План 2



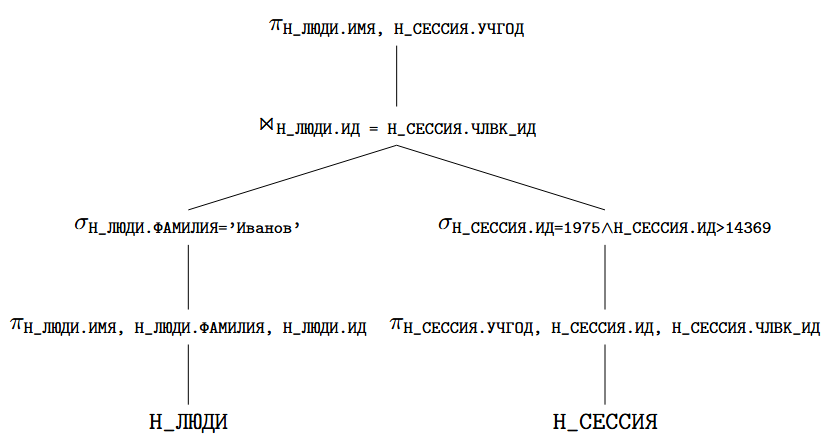
План 3



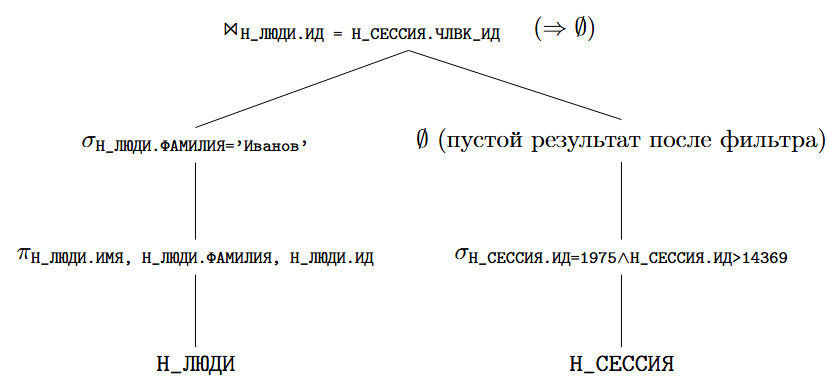
План 4



План 5 – Оптимальный



План 6 – Самый оптимальный (противоречивость фильтров)



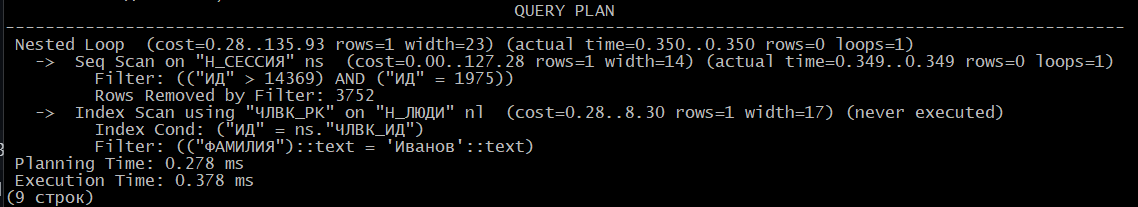
Наиболее оптимальным из рассмотренных планов является План 6. Его эффективность обусловлена использованием ключевой особенности запроса: противоречивого фильтра Н\_СЕССИЯ.ИД = 1975 AND Н\_СЕССИЯ.ИД > 14369. Этот фильтр немедленно определяет, что ветка обработки таблицы Н\_СЕССИЯ дает пустой результат, что делает весь последующий JOIN и итоговый результат запроса также пустыми, позволяя СУБД избежать значительных затрат на обработку таблицы Н\_ЛЮДИ.

В общем случае, при отсутствии таких специфических противоречивых условий, План 5 (с максимально ранним применением фильтраций и проекций для уменьшения промежуточных данных) представлял бы собой эффективную эвристическую стратегию оптимизации.

При добавлении предложенных B-tree индексов:

* План 6 выполняется еще быстрее, так как индекс на Н\_СЕССИЯ.ИД позволяет СУБД мгновенно (через поиск по индексу) обнаружить отсутствие строк, удовлетворяющих противоречивому фильтру, вместо полного сканирования.
* Для других планов (например, если бы План 5 был оптимальным из-за отсутствия противоречий) индексы кардинально изменяют и ускоряют физическое выполнение каждой операции (фильтрации, соединения), даже если логическая структура плана остается схожей. Это приводит к значительному повышению общей производительности запроса.

EXPLAIN ANALYZE для запроса

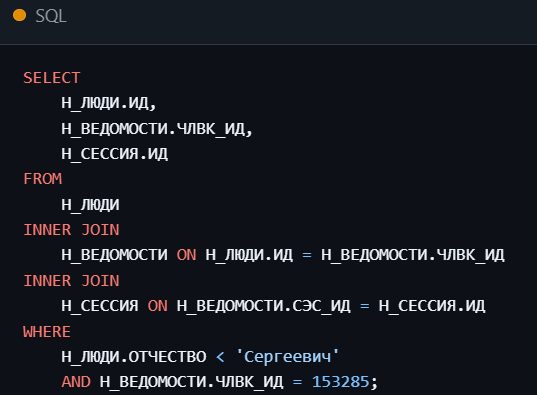


*Рис. 2. QUERY PLAN для запроса 1*

1. Результат запроса: Запрос вернул 0 строк (actual time=0.350..0.350 rows=0), что ожидаемо из-за взаимоисключающих условий NS.ИД = 1975 и NS.ИД > 14369.
2. Основная стратегия: PostgreSQL выбрал Nested Loop соединение.
3. Обработка Н\_СЕССИЯ: Первым шагом (-> Seq Scan on "Н\_СЕССИЯ" ns) было полное сканирование таблицы Н\_СЕССИЯ. Фильтр (("ИД" > 14369) AND ("ИД" = 1975)) был применен на этом этапе и не вернул ни одной строки (actual time=0.349..0.349 rows=0), отбросив все 3752 прочитанные строки.
4. Оптимизация для Н\_ЛЮДИ: Ключевым моментом является то, что доступ к таблице Н\_ЛЮДИ (-> Index Scan using "ЧЛВК\_PK" on "Н\_ЛЮДИ" nl) никогда не выполнялся (never executed). Поскольку первый этап (обработка Н\_СЕССИЯ) не дал строк для соединения, PostgreSQL разумно пропустил всю работу, связанную с таблицей Н\_ЛЮДИ.
5. Время выполнения: Общее время выполнения крайне мало (Execution Time: 0.378 ms), что подтверждает эффективность выбранного плана.

# Запрос 2

Реализация



*Рис. 3. Реализация запроса 2 на SQL*

Возможные индексы

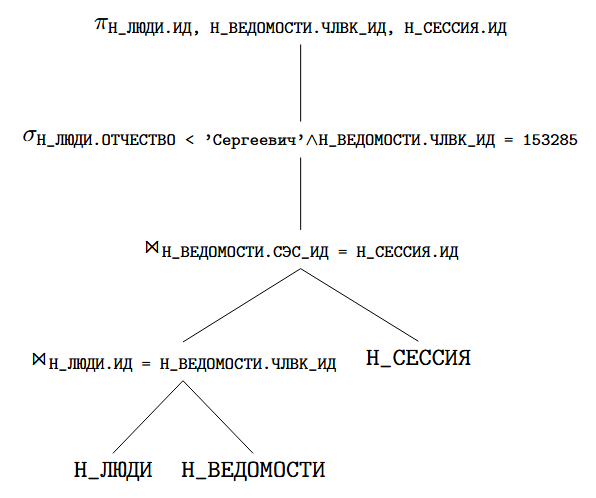
Таблица 2 – Предлагаемые индексы для атрибутов таблиц запроса 2

|  |  |  |
| --- | --- | --- |
| Тип индекса | Таблица.Атрибут | Обоснование |
| B-tree | Н\_ВЕДОМОСТИ. (ЧЛВК\_ИД, СЭС\_ИД) | Этот индекс для условия WHERE Н\_ВЕДОМОСТИ.ЧЛВК\_ИД = 153285. Так как это равенство конкретному значению, СУБД сможет очень быстро найти все строки в Н\_ВЕДОМОСТИ, удовлетворяющие этому условию. |
| B-tree | Н\_ЛЮДИ.ИД | Этот атрибут используется в условии соединения ON Н\_ЛЮДИ.ИД = Н\_ВЕДОМОСТИ.ЧЛВК\_ИД. Он позволяет быстро находить строки в Н\_ЛЮДИ при соединении с отфильтрованным (и, вероятно, небольшим) набором данных из Н\_ВЕДОМОСТИ. |
| B-tree | Н\_ЛЮДИ. ОТЧЕСТВО | Этот атрибут используется в условии WHERE Н\_ЛЮДИ.ОТЧЕСТВО < 'Сергеевич'. B-tree индексы хорошо подходят для операций диапазонного сравнения (таких как <, >, <=, >=). Индекс позволит СУБД быстрее находить строки, удовлетворяющие этому условию, вместо полного сканирования таблицы Н\_ЛЮДИ. |
| B-tree | Н\_СЕССИЯ.ИД | Этот атрибут используется в условии соединения ON Н\_ВЕДОМОСТИ.СЭС\_ИД = Н\_СЕССИЯ.ИД. Он необходим для быстрого поиска соответствующих записей в Н\_СЕССИЯ по значениям СЭС\_ИД, полученным из таблицы Н\_ВЕДОМОСТИ. |

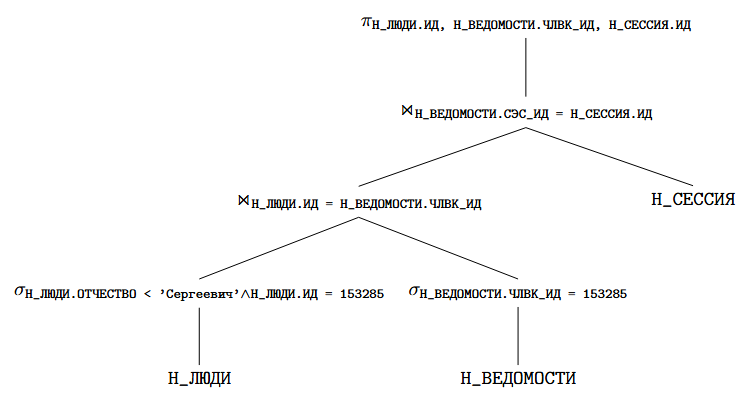
Без этих индексов СУБД, вероятно, пришлось бы выполнять полные сканирования таблиц (Table Scan) для поиска нужных строк, что значительно медленнее, особенно на больших таблицах. Предложенные B-tree индексы позволяют СУБД использовать более эффективные методы доступа к данным, такие как поиск по индексу (Index Seek) или сканирование диапазона индекса (Index Range Scan).

Возможные планы запроса

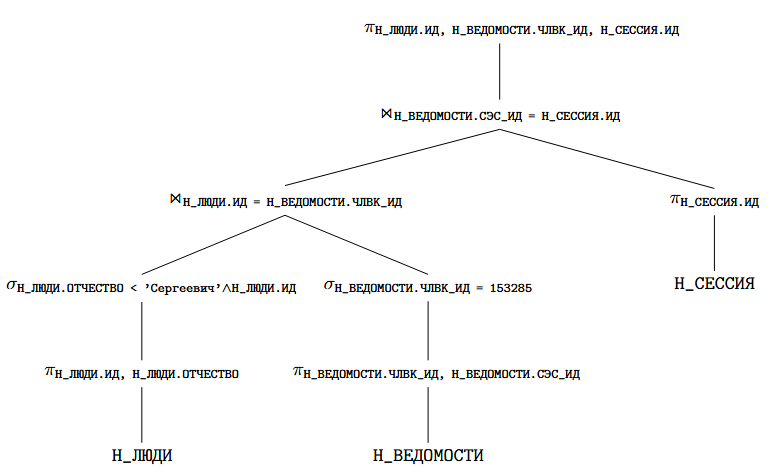
План 1



План 2



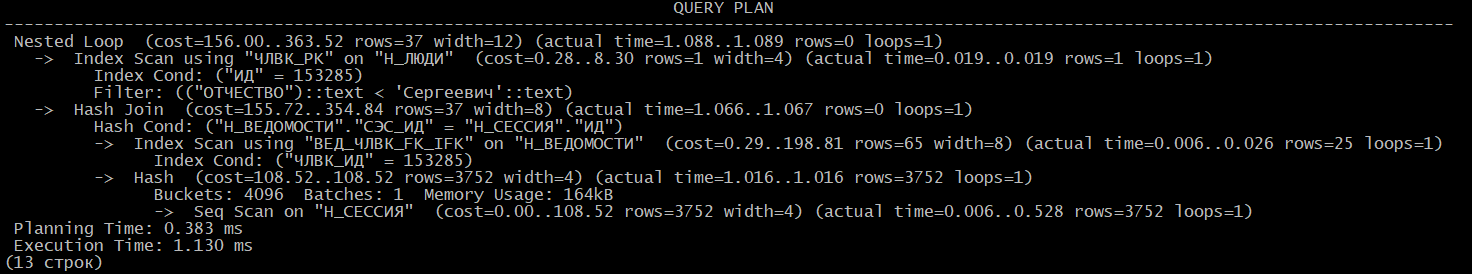
План 3 – Оптимальный



План 3 является наиболее эффективным из предложенных, поскольку он следует ключевым принципам оптимизации запросов при отсутствии индексов. Во-первых, он применяет наиболее селективный фильтр (Н\_ВЕДОМОСТИ.ЧЛВК\_ИД = 153285) как можно раньше, что кардинально сокращает количество строк из таблицы Н\_ВЕДОМОСТИ для дальнейшей обработки. Во-вторых, остальные фильтры (Н\_ЛЮДИ.ОТЧЕСТВО < 'Сергеевич' и выведенный Н\_ЛЮДИ.ИД = 153285) также применяются до операций соединения.

Кроме того, План 3 использует ранние проекции, удаляя ненужные столбцы из таблиц перед соединениями. Это уменьшает "ширину" данных, снижая объем информации, обрабатываемой на каждом этапе, и, как следствие, стоимость самих операций соединения. В результате, соединения выполняются над значительно меньшими по объему (как по количеству строк, так и по количеству столбцов) наборами данных, что ведет к существенному повышению производительности по сравнению с планами, где фильтры или проекции применяются позже.

EXPLAIN ANALYZE для запроса



*Рис. 4. QUERY PLAN для запроса 2*

PostgreSQL эффективно использовал имеющиеся индексы (ЧЛВК\_РК на Н\_ЛЮДИ и ВЕД\_ЧЛВК\_FK\_IFK на Н\_ВЕДОМОСТИ) для применения условий равенства. Оптимизатор определил, что соединение между отфильтрованными данными Н\_ВЕДОМОСТИ и таблицей Н\_СЕССИЯ не дает результатов. Это "раннее" определение пустого набора данных на одном из этапов соединения позволило избежать дальнейшей ненужной обработки и обеспечило быстрое выполнение запроса. План демонстрирует, что отсутствие совпадений по ключу Н\_ВЕДОМОСТИ.СЭС\_ИД = Н\_СЕССИЯ.ИД для человека с ИД=153285 является причиной пустого результата.

# Выводы по работе

В процессе выполнения лабораторной работы я познакомился с различными видами индексации, разобрался, когда какой способ лучше применять. Также узнал о том, как psql оптимизирует запросы, выбирая план выполнения, автоматическую индексацию, а также методику соединения таблиц (hash join, sort-merge, nested-loop, …).