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: P3111
Преподаватель: Харитонова Анастасия Евгеньевна

Санкт-Петербург 2025г.

Содержание

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: H_ЛЮДИ, H_СЕССИЯ.
   Вывести атрибуты: H_ЛЮДИ.ИМЯ, H_СЕССИЯ.УЧГОД.
   Фильтры (AND):
   a) H_ЛЮДИ.ФАМИЛИЯ = Иванов.
   b) H_СЕССИЯ.ИД = 1975.
   c) H_СЕССИЯ.ИД > 14369.
   Вид соединения: RIGHT JOIN.
2. Сделать запрос для получения атрибутов из указанных таблиц, применив фильтры по указанным условиям:
   Таблицы: H_ЛЮДИ, H_ВЕДОМОСТИ, H_СЕССИЯ.
   Вывести атрибуты: H_ЛЮДИ.ИД, H_ВЕДОМОСТИ.ЧЛВК_ИД, H_СЕССИЯ.ИД.
   Фильтры (AND):
   a) H_ЛЮДИ.ОТЧЕСТВО < Сергеевич.
   b) H_ВЕДОМОСТИ.ЧЛВК_ИД = 153285.
   Вид соединения: INNER JOIN.

Запрос 1

Реализация

```sql
SELECT
    Н_ЛЮДИ.ИМЯ,
    Н_СЕССИЯ.УЧГОД
FROM
    Н_ЛЮДИ
RIGHT JOIN
    Н_СЕССИЯ ON Н_ЛЮДИ.ИД = Н_СЕССИЯ.ЧЛВК_ИД
WHERE
    Н_ЛЮДИ.ФАМИЛИЯ = 'Иванов'
    AND Н_СЕССИЯ.ИД = 1975
    AND Н_СЕССИЯ.ИД > 14369;
```

Рис. 1. Реализация запроса 1 на SQL

Возможные индексы

Таблица 1 – Предлагаемые индексы для атрибутов таблиц запроса 1

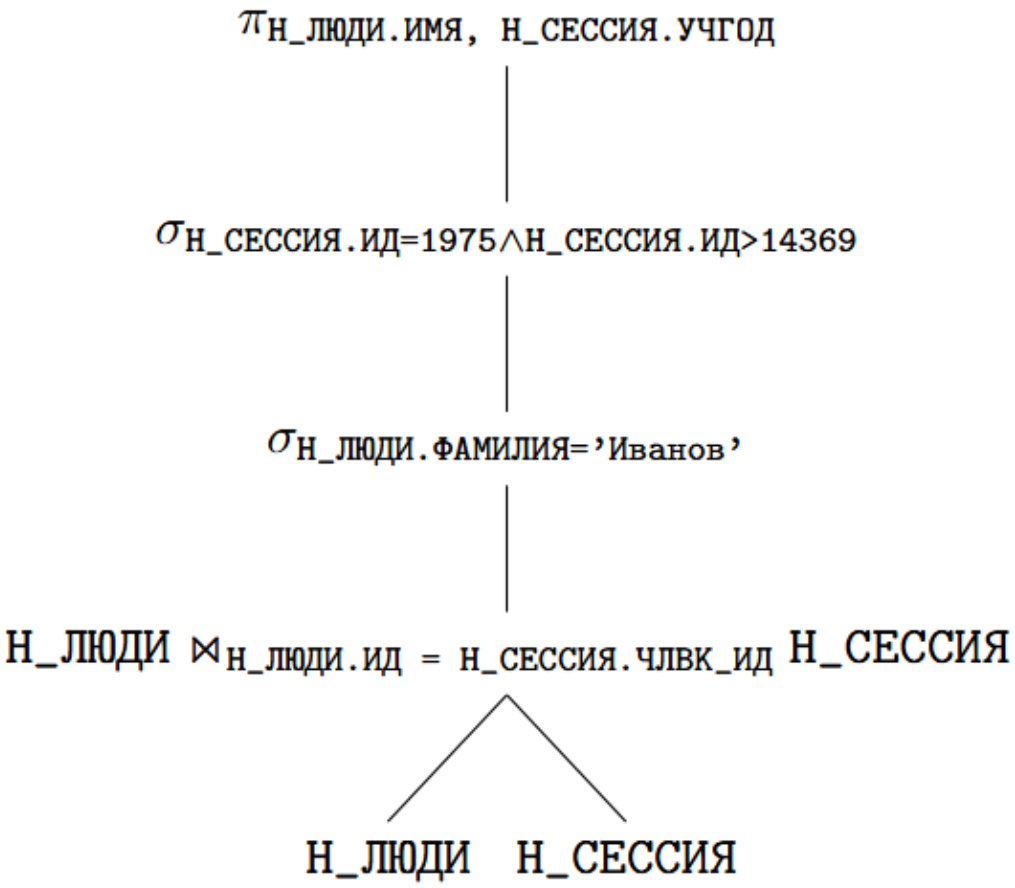
Тип индекса	Таблица.Атрибут	Обоснование
B-tree	Н_ЛЮДИ.ФАМИЛИЯ	Ускоряет поиск по условию WHERE Н_ЛЮДИ.ФАМИЛИЯ = 'Иванов'. Без индекса СУБД пришлось бы выполнять полный просмотр таблицы Н_ЛЮДИ (Full Table Scan). С B-tree индексом поиск будет значительно быстрее (Index Seek/Scan), так как СУБД сможет быстро 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).

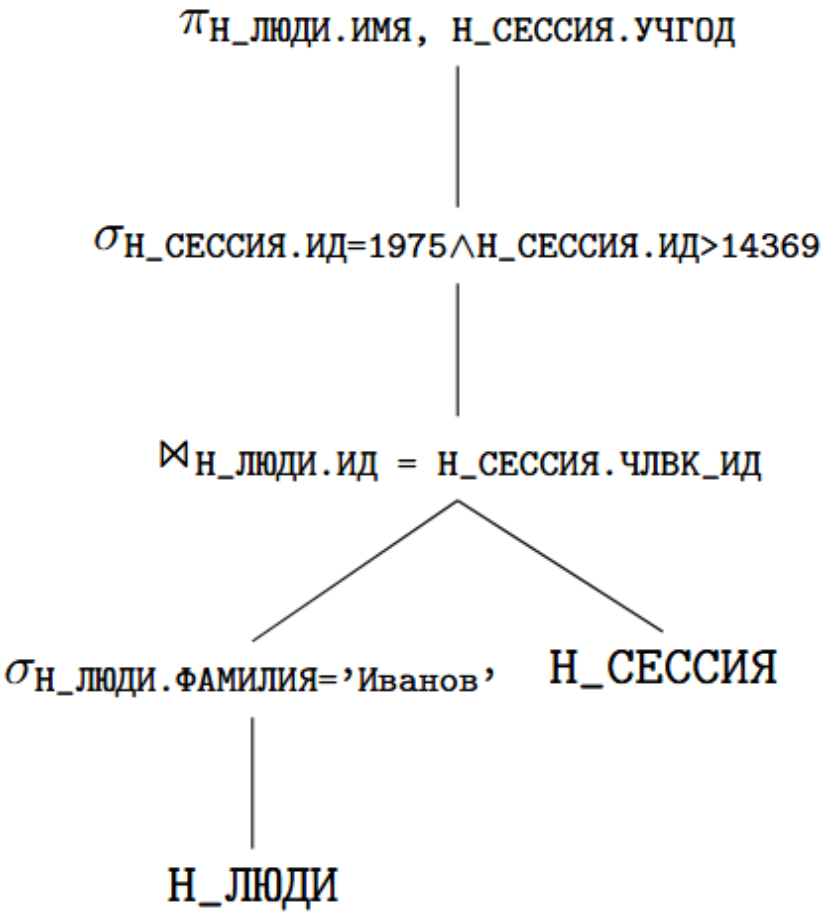
Для всех указанных атрибутов (Н_ЛЮДИ.ФАМИЛИЯ, Н_СЕССИЯ.ИД, Н_ЛЮДИ.ИД, Н_СЕССИЯ.ЧЛВК_ИД) предпочтение отдается B-tree индексам из-за их универсальности: они эффективны как для точечных и диапазонных условий в WHERE, так и для операций JOIN. Индексы на Н_ЛЮДИ.ИД и Н_СЕССИЯ.ЧЛВК_ИД критически важны для 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

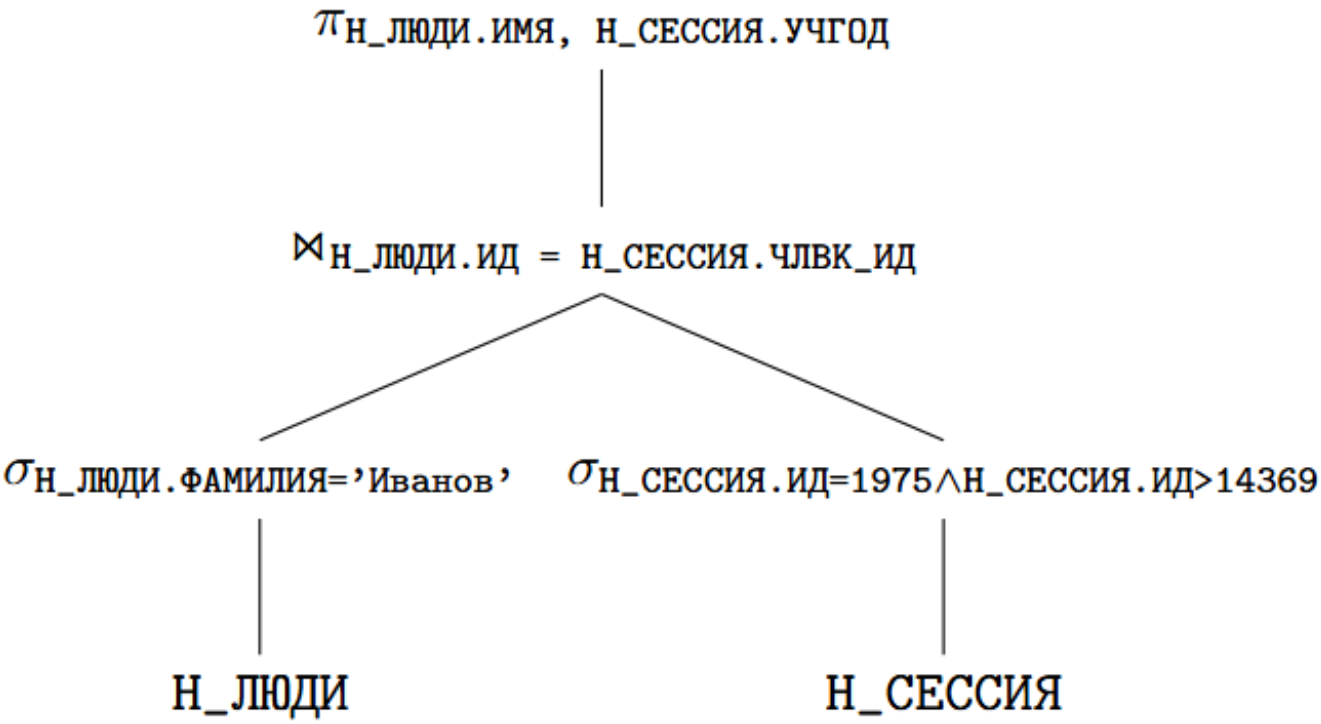
План 1



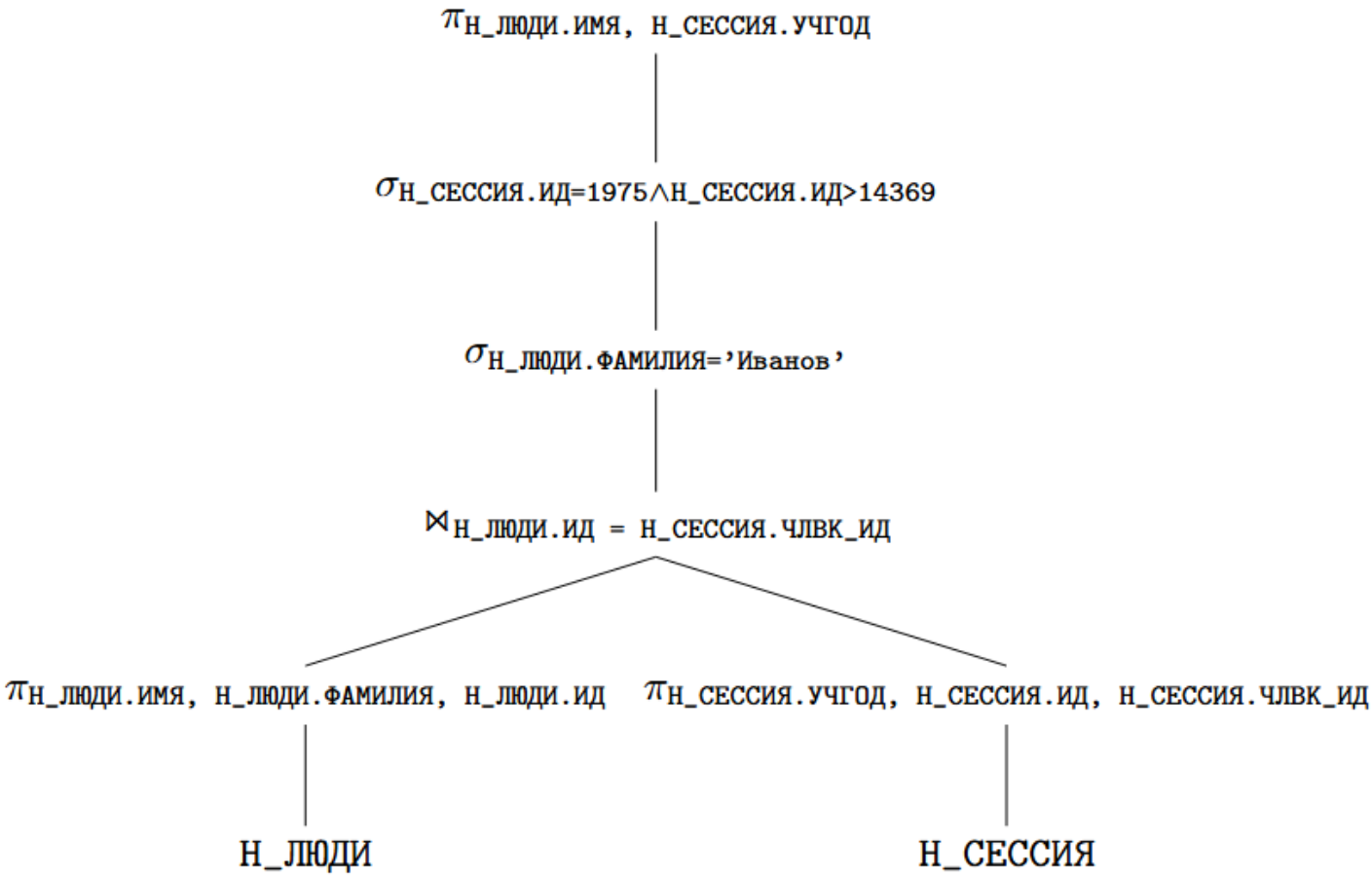
План 2


πН_ЛЮДИ.ИМЯ, Н_СЕССИЯ.УЧГОД
σН_СЕССИЯ.ИД=1975∧Н_СЕССИЯ.ИД>14369
⋈Н_ЛЮДИ.ИД = Н_СЕССИЯ.ЧЛВК_ИД
σН_ЛЮДИ.ФАМИЛИЯ='Иванов'
Н_СЕССИЯ
Н_ЛЮДИ

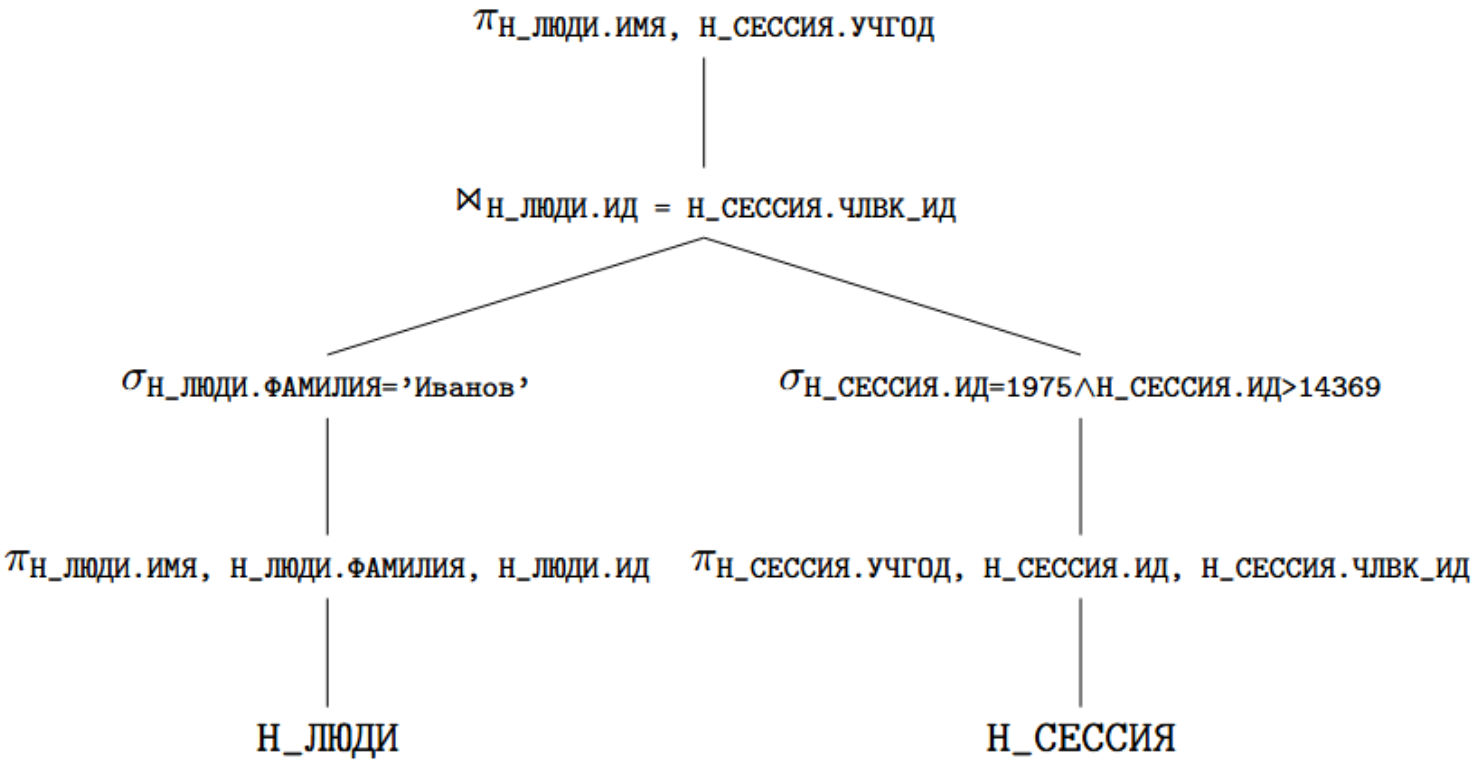
План 3


πН_ЛЮДИ.ИМЯ, Н_СЕССИЯ.УЧГОД
⋈Н_ЛЮДИ.ИД = Н_СЕССИЯ.ЧЛВК_ИД
σН_ЛЮДИ.ФАМИЛИЯ='Иванов'
σН_СЕССИЯ.ИД=1975∧Н_СЕССИЯ.ИД>14369
Н_ЛЮДИ
Н_СЕССИЯ

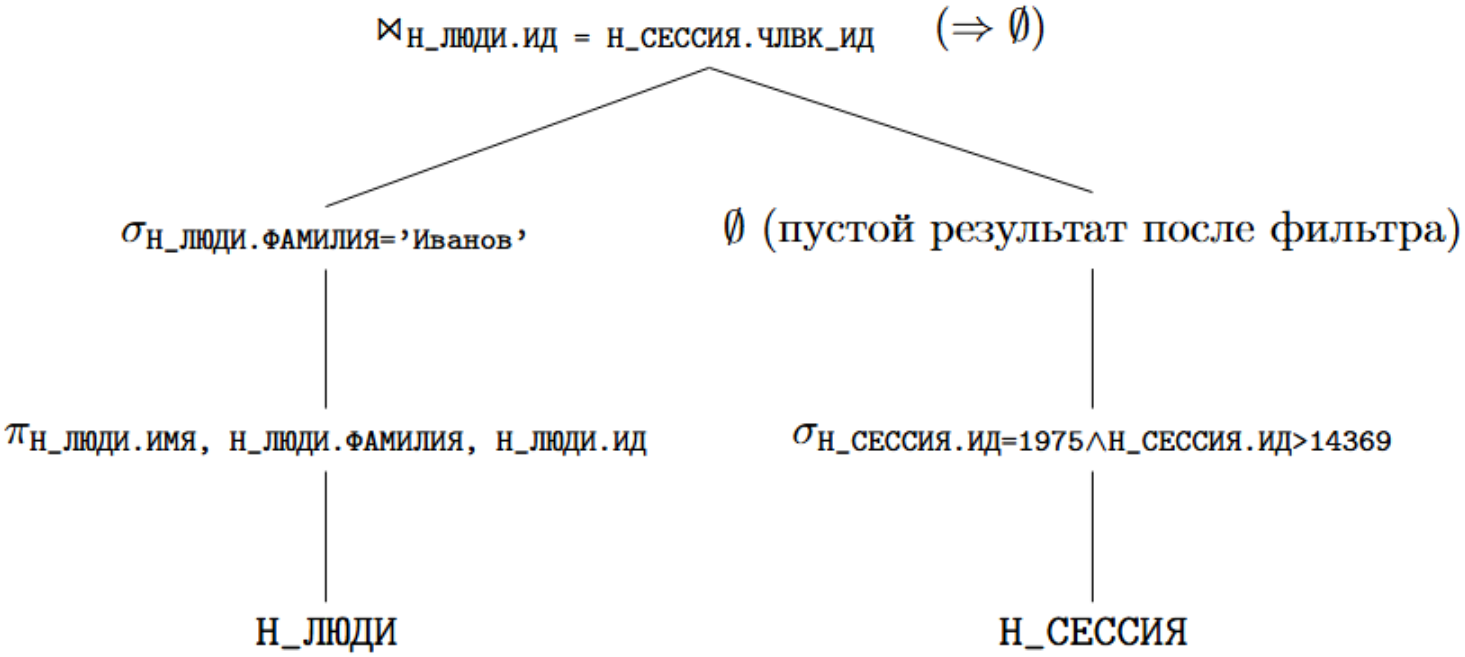
План 4



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

```
                                                    QUERY PLAN
-------------------------------------------------------------------------------------------------------------------
 Nested Loop  (cost=0.28..135.93 rows=1 width=23) (actual time=0.350..0.350 rows=0 loops=1)
   ->  Seq Scan on "Н_СЕССИЯ" ns  (cost=0.00..127.28 rows=1 width=14) (actual time=0.349..0.349 rows=0 loops=1)
         Filter: (("ИД" > 14369) AND ("ИД" = 1975))
         Rows Removed by Filter: 3752
   ->  Index Scan using "ЧЛВК_РК" on "Н_ЛЮДИ" nl  (cost=0.28..8.30 rows=1 width=17) (never executed)
         Index Cond: ("ИД" = ns."ЧЛВК_ИД")
         Filter: (("ФАМИЛИЯ")::text = 'Иванов'::text)
 Planning Time: 0.278 ms
 Execution Time: 0.378 ms
(9 строк)
```

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

```sql
SELECT
    Н_ЛЮДИ.ИД,
    Н_ВЕДОМОСТИ.ЧЛВК_ИД,
    Н_СЕССИЯ.ИД
FROM
    Н_ЛЮДИ
INNER JOIN
    Н_ВЕДОМОСТИ ON Н_ЛЮДИ.ИД = Н_ВЕДОМОСТИ.ЧЛВК_ИД
INNER JOIN
    Н_СЕССИЯ ON Н_ВЕДОМОСТИ.СЭС_ИД = Н_СЕССИЯ.ИД
WHERE
    Н_ЛЮДИ.ОТЧЕСТВО < 'Сергеевич'
    AND Н_ВЕДОМОСТИ.ЧЛВК_ИД = 153285;
```

Рис. 3. Реализация запроса 2 на SQL

Возможные индексы

Таблица 2 – Предлагаемые индексы для атрибутов таблиц запроса 2

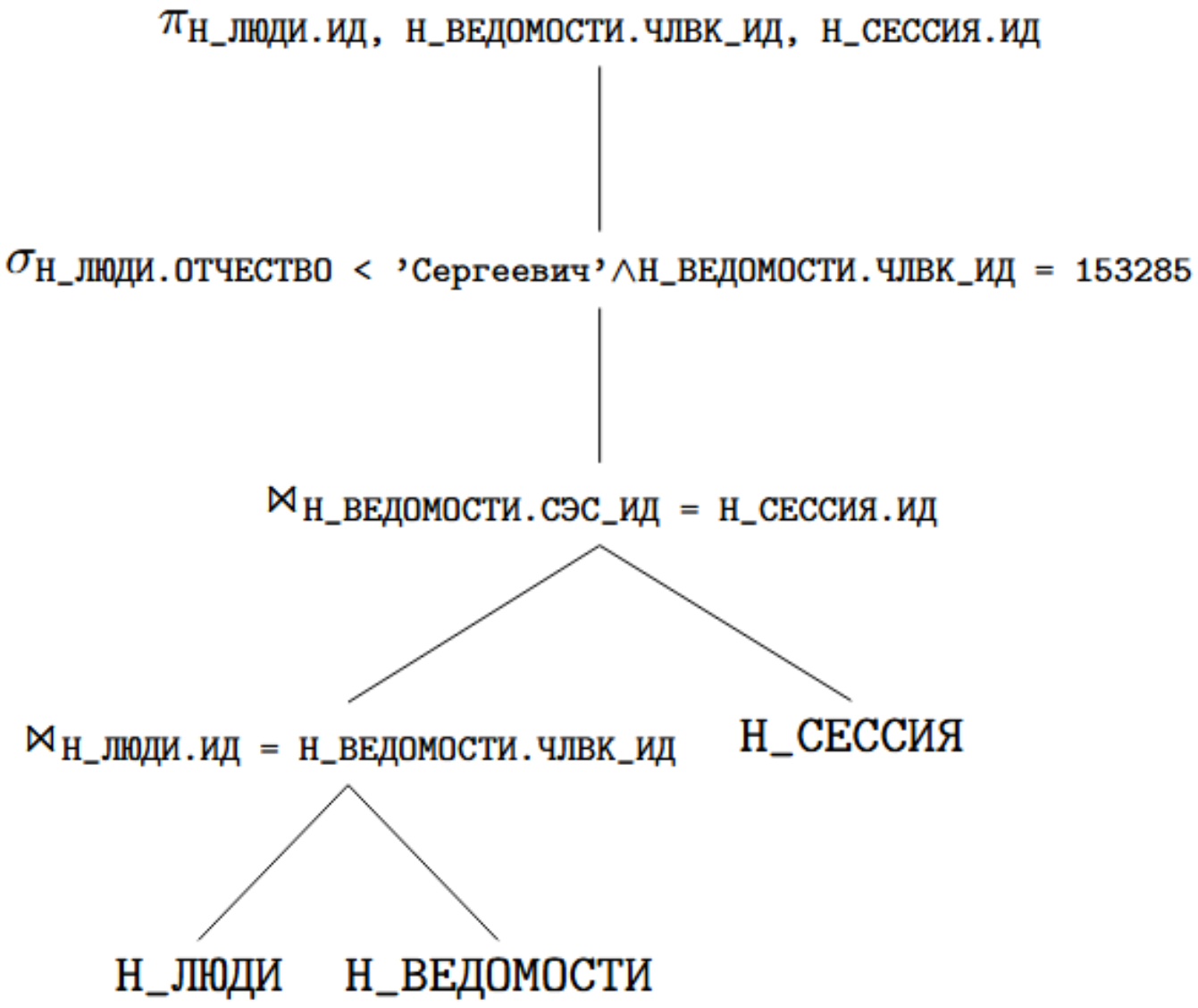
Тип индекса	Таблица.Атрибут	Обоснование
B-tree	Н_ВЕДОМОСТИ. (ЧЛВК_ИД, СЭС_ИД)	Этот индекс для условия WHERE Н_ВЕДОМОСТИ.ЧЛВК_ИД = 153285. Так как это равенство конкретному значению, СУБД сможет очень быстро найти все строки в Н_ВЕДОМОСТИ, удовлетворяющие этому условию.
B-tree	Н_ЛЮДИ.ИД	Этот атрибут используется 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.

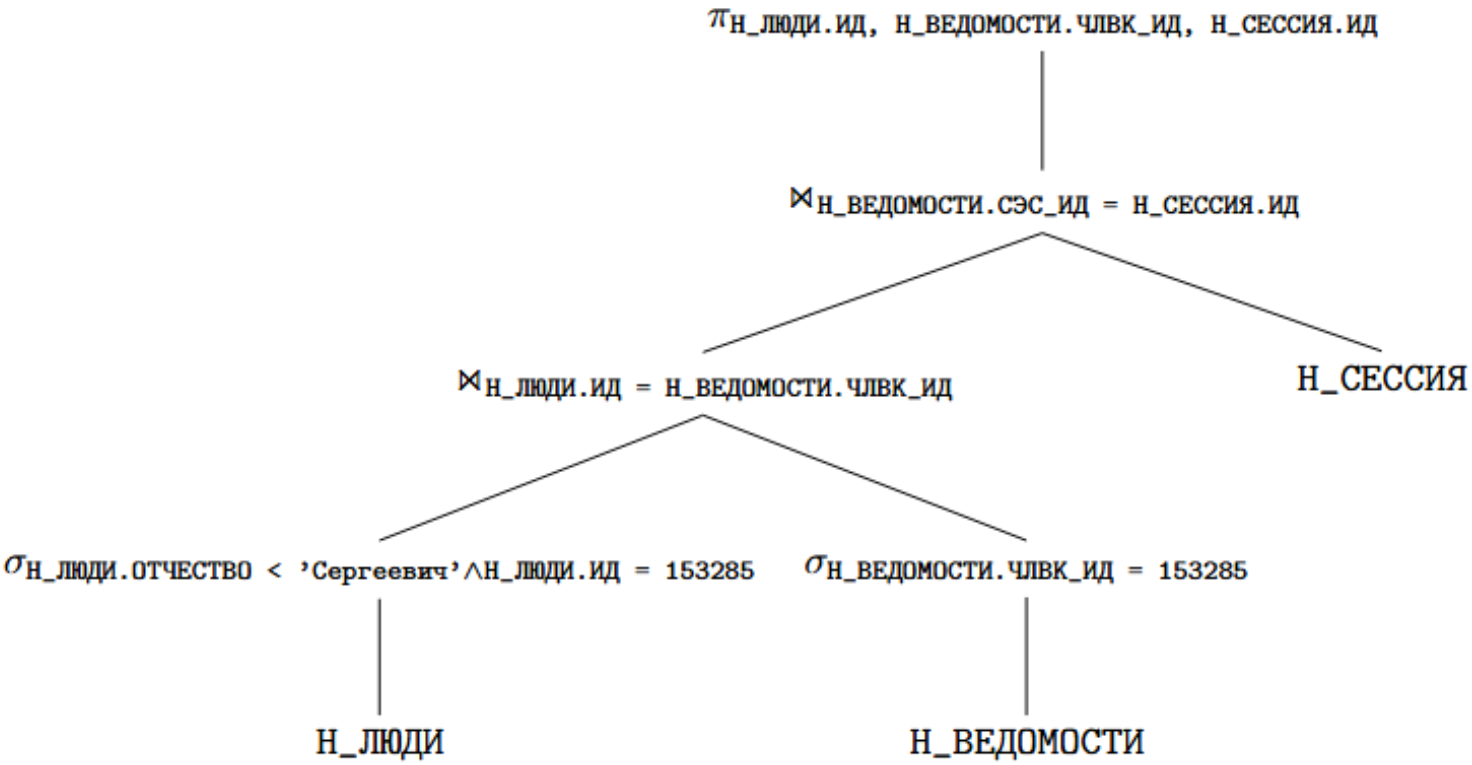
Без этих индексов СУБД, вероятно, пришлось бы выполнять полные сканирования таблиц (Table Scan) для поиска нужных строк, что значительно медленнее, особенно на больших таблицах. Предложенные B-tree индексы позволяют СУБД использовать более эффективные методы доступа к данным, такие как поиск по индексу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

Возможные планы запроса

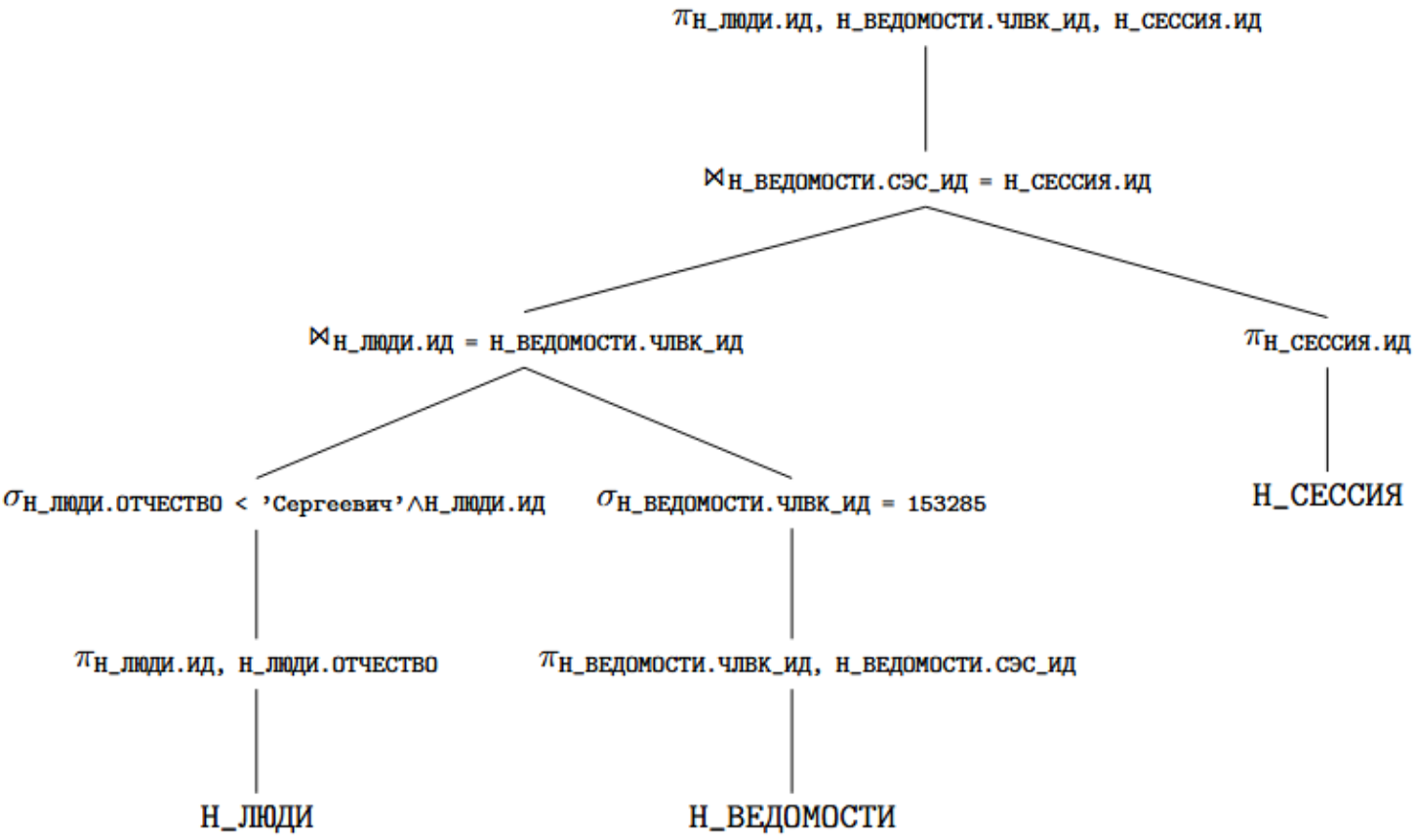
План 1



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

```
                                                              QUERY PLAN
---------------------------------------------------------------------------------------------------------------------------------------
 Nested Loop  (cost=156.00..363.52 rows=37 width=12) (actual time=1.088..1.089 rows=0 loops=1)
   ->  Index Scan using "ЧЛВК_РК" on "Н_ЛЮДИ"  (cost=0.28..8.30 rows=1 width=4) (actual time=0.019..0.019 rows=1 loops=1)
         Index Cond: ("ИД" = 153285)
         Filter: (("ОТЧЕСТВО")::text < 'Сергеевич'::text)
   ->  Hash Join  (cost=155.72..354.84 rows=37 width=8) (actual time=1.066..1.067 rows=0 loops=1)
         Hash Cond: ("Н_ВЕДОМОСТИ"."СЭС_ИД" = "Н_СЕССИЯ"."ИД")
         ->  Index Scan using "ВЕД_ЧЛВК_FK_IFK" on "Н_ВЕДОМОСТИ"  (cost=0.29..198.81 rows=65 width=8) (actual time=0.006..0.026 rows=25 loops=1)
               Index Cond: ("ЧЛВК_ИД" = 153285)
         ->  Hash  (cost=108.52..108.52 rows=3752 width=4) (actual time=1.016..1.016 rows=3752 loops=1)
               Buckets: 4096  Batches: 1  Memory Usage: 164kB
               ->  Seq Scan on "Н_СЕССИЯ"  (cost=0.00..108.52 rows=3752 width=4) (actual time=0.006..0.528 rows=3752 loops=1)
 Planning Time: 0.383 ms
 Execution Time: 1.130 ms
(13 строк)
```

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, …).