Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение высшего образования
«Национальный исследовательский университет ИТМО»

Факультет программной инженерии и компьютерной техники

Базы данных

Лабораторная работа № 4

Вариант № 1301

Выполнил:

Сандов Кирилл Алекссевич

Группа:

P3113

Проверил:

преподаватель практики Горбунов Михаил Витальевич

Санкт-Петербург

2023

Оглавление

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).

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.ИД.

Фильтры (AND):

a) Н_ТИПЫ_ВЕДОМОСТЕЙ.НАИМЕНОВАНИЕ < Ведомость.

b) Н_ВЕДОМОСТИ.ДАТА = 2022-06-08.

Вид соединения: LEFT JOIN.

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

Таблицы: Н_ЛЮДИ, Н_ВЕДОМОСТИ, Н_СЕССИЯ.

Вывести атрибуты: Н_ЛЮДИ.ФАМИЛИЯ, Н_ВЕДОМОСТИ.ДАТА, Н_СЕССИЯ.ЧЛВК_ИД.

Фильтры (AND):

a) Н_ЛЮДИ.ФАМИЛИЯ > Ёлкин.

b) Н_ВЕДОМОСТИ.ЧЛВК_ИД = 163249.

c) Н_СЕССИЯ.УЧГОД < 2008/2009.

Вид соединения: INNER JOIN.

Реализация запросов

Запрос 1

```
SELECT Н_ТИПЫ_ВЕДОМОСТЕЙ.ИД, Н_ВЕДОМОСТИ.ИД
FROM Н_ВЕДОМОСТИ
JOIN Н_ТИПЫ_ВЕДОМОСТЕЙ ON Н_ВЕДОМОСТИ.ТВ_ИД =
Н_ТИПЫ_ВЕДОМОСТЕЙ.ИД
WHERE (
    Н_ТИПЫ_ВЕДОМОСТЕЙ.НАИМЕНОВАНИЕ < 'Ведомость'
    AND
    Н_ВЕДОМОСТИ.ДАТА = '2022-06-08'
);
```

Листинг 1

Запрос 2

```
SELECT Н_ЛЮДИ.ФАМИЛИЯ, Н_ВЕДОМОСТИ.ДАТА, Н_СЕССИЯ.ЧЛВК_ИД
FROM Н_ЛЮДИ
JOIN Н_СЕССИЯ ON Н_ЛЮДИ.ИД = Н_СЕССИЯ.ЧЛВК_ИД
JOIN Н_ВЕДОМОСТИ ON Н_ЛЮДИ.ИД = Н_ВЕДОМОСТИ.ЧЛВК_ИД
WHERE (
    Н_ЛЮДИ.ФАМИЛИЯ > 'Ёлкин' AND
    Н_ВЕДОМОСТИ.ЧЛВК_ИД = 163249 AND
    Н_СЕССИЯ.УЧГОД < '2008/2009'
);
```

Листинг 2

Применение индексов

Далее для каждого из запросов будут записаны возможные индексы, мотивация к их применению и оценка их эффективности.

Запрос 1

1) Индекс на атрибут ДАТА таблицы Н_ВЕДОМОСТИ, тип индекса – Hash.

```
CREATE INDEX Н_ВЕДОМОСТИ_ДАТА_I
ON Н_ВЕДОМОСТИ
USING hash(ДАТА)
```

Листинг 3

Посмотрим на статистику данного атрибута в специальной таблице pg_stats:

```
+----------+-----------+
|n_distinct|correlation|
+----------+-----------+
|1836      |0.5155182  |
+----------+-----------+
```

Листинг 4

Мы видим, что значение n_distinct, показывающее, сколько уникальных значений данного атрибута существует в таблице в данный момент, довольно большое, а также correlation ближе к 1, чем к 0, что означает необходимость в применение индекса по данному атрибуту. Ведь это упорядочит значения, что ускорит фильтрацию по ним в блоке WHERE.

Также был выбран тип Hash, потому что он наиболее быстрый для операции равенства.

Причины, почему другие индексы будут бесполезные для данного запроса:

- Индекс в таблице Н_ТИПЫ_ВЕДОМОСТЕЙ на атрибут НАИМЕНОВАНИЕ – он бесполезен, так как в данной таблице всего три строки.

Запрос 2

1) Индекс на атрибут ЧЛВК_ИД таблицы Н_СЕССИЯ, тип индекса – Hash. С указанием того, что применяем на значениях не NULL.

```
CREATE INDEX Н_СЕССИЯ_ЧЛВК_ИД_I
ON Н_СЕССИЯ
USING hash(ЧЛВК_ИД)
WHERE ЧЛВК_ИД IS NOT NULL
```

Листинг 5

Статистика данного атрибута:

```
+----------+----------+-----------+
|null_frac |n_distinct|correlation|
+----------+----------+-----------+
|0.13672708|180       |0.13944401 |
+----------+----------+-----------+
```

Листинг 6

Видим, что n_distinct и correlation не говорят о том, что индекс сильно необходим. Однако вспомним, что данный атрибут участвует в операции соединения с таблицей Н_ЛЮДИ, где строк довольно много. Поэтому к этому атрибуту будет довольного много обращений, что подтверждает полезность индекса. Также заметим, что null_frac (доля значений NULL среди всех значений) имеет некоторое не малое значений. А при соединении вида INNER JOIN к Н_ЛЮДИ значения NULL использоваться не будут. Следовательно, мы можем добавить оптимизацию, исключив из индексации значения NULL.

Был выбран тип индекса Hash, так как в операции соединения постоянно будут использовать сравнения по равенству.

2) Индекс на атрибут ЧЛВК_ИД таблицы Н_ВЕДОМОСТИ, тип индекса – Hash.

```
CREATE INDEX Н_ВЕДОМОСТИ_ЧЛВК_ИД_I
ON Н_ВЕДОМОСТИ
USING hash(ЧЛВК_ИД);
```

Листинг 7

Статистика атрибута:

```
+---------+----------+-----------+
|null_frac|n_distinct|correlation|
+---------+----------+-----------+
|0        |3264      |0.5222814  |
+---------+----------+-----------+
```

Листинг 8

В данном случае мотивация совпадает с предыдущим индексом. Но мы не исключили значения NULL, потому что их нет (null_frac равен 0).

3) Индекс на атрибуты ФАМИЛИЯ, ИД таблицы Н_ЛЮДИ, тип индекса B-tree. С дополнительным условием выборки: ФАМИЛИЯ > 'Ёлкин'.

```
CREATE INDEX Н_ЛЮДИ_ФАМИЛИЯ_ИД_I
ON Н_ЛЮДИ
USING btree(ФАМИЛИЯ, ИД)
WHERE ФАМИЛИЯ > 'Ёлкин';
```

Листинг 9

Статистика атрибута ФАМИЛИЯ:

```
+---------+----------+------------+
|null_frac|n_distinct|correlation |
+---------+----------+------------+
|0        |-0.7317311|-0.000653284|
+---------+----------+------------+
```

Листинг 10

Отрицательное значение n_distinct означает (если взять его по модулю) долю уникальных значений среди всех значений. Это доля довольно большая, поэтому созданный индекс с фильтром отсеет большое количество ненужных запросу значений и ускорит фильтрацию. Также в состав индекса был включён атрибут ИД, поскольку при выполнении запроса нужно будет проводить фильтрацию по обоим атрибутам таблицы.

Был выбран тип B-tree, так как он поддерживает операции сравнения, а у нас как раз таки сравнение по полю ФАМИЛИЯ.

Причины, почему не были созданы другие индексы:

- Индекс в таблице Н_СЕССИЯ на атрибут УЧГОД – при просмотре статистики данной таблицы было выявлено,

что в ней всего 11 уникальных значений, что слишком мало для использования индекса;
- Индекс в таблице `Н_ЛЮДИ` на атрибут `ИД` – он уже создан СУБД автоматически как индекс для основного ключа.

Планы выполнения запросов

Запрос 1

План 1

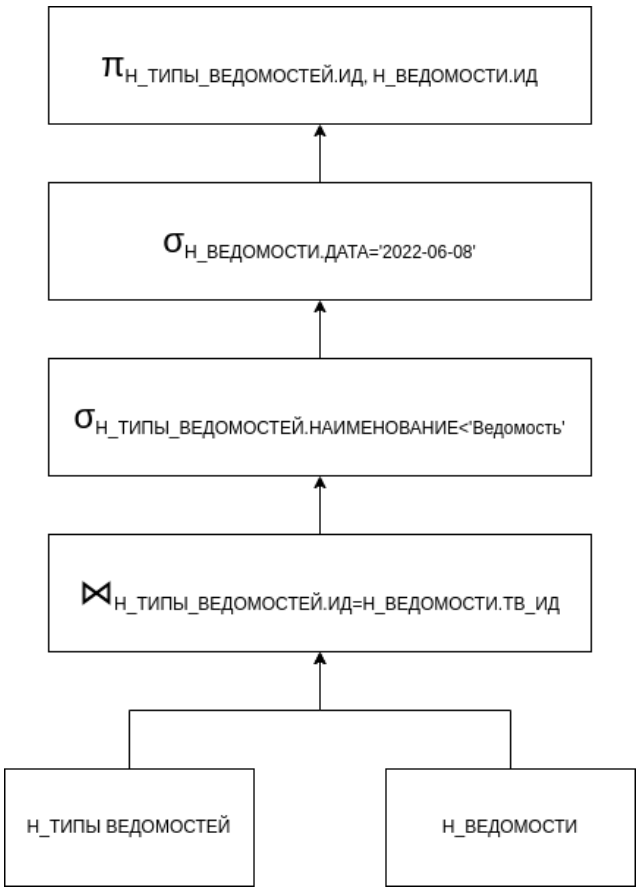

Рисунок 1

План 2

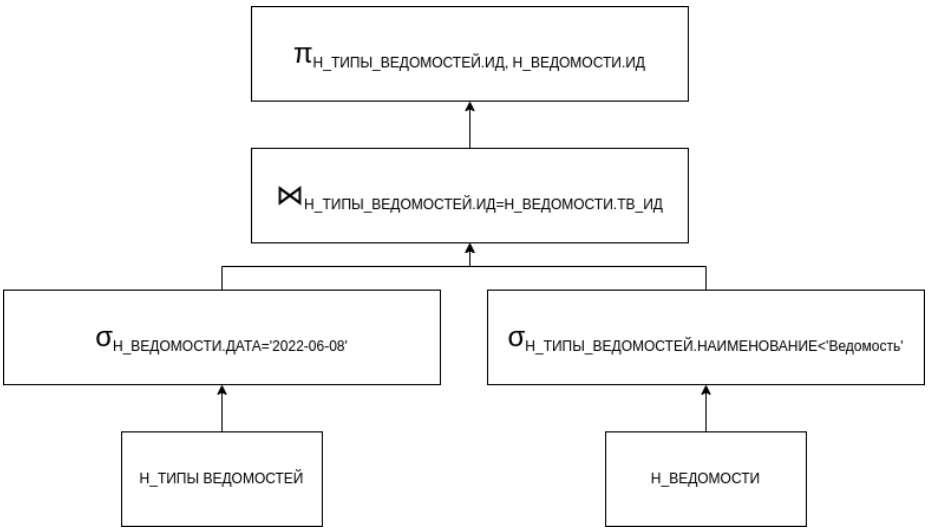

Рисунок 2

План 3

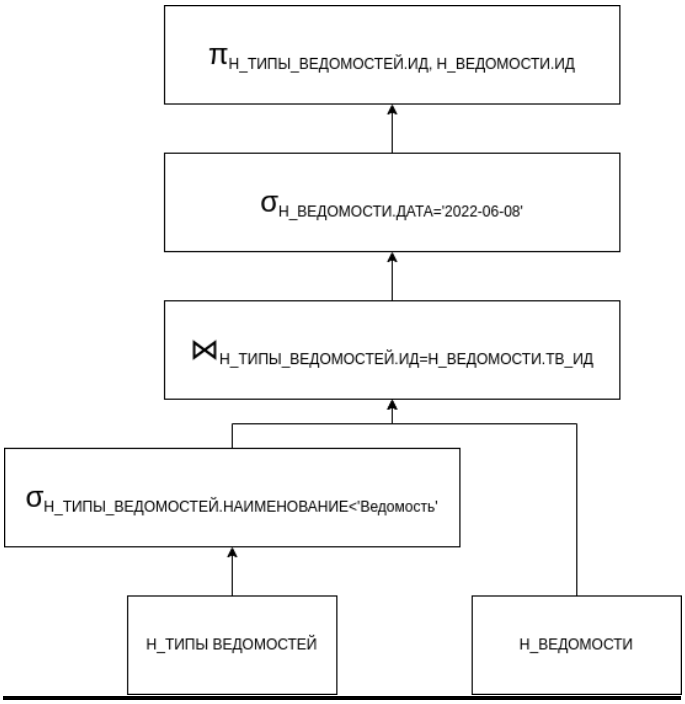

Рисунок 3

План 4

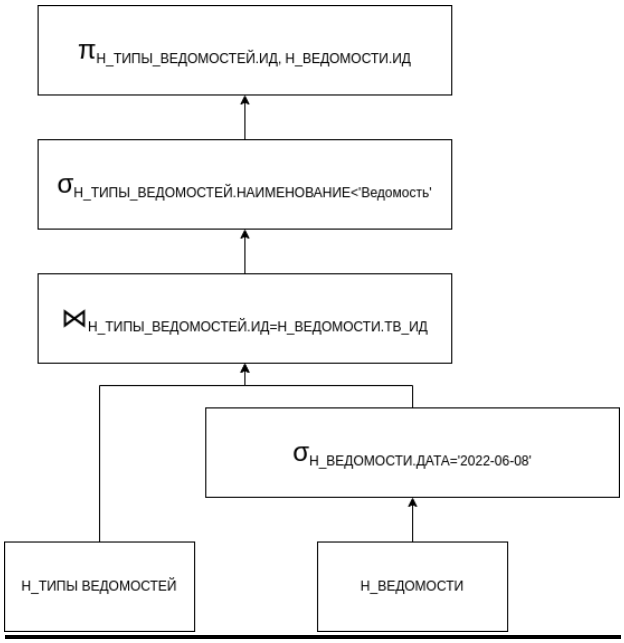

Рисунок 4

Наиболее эффективным является план 2, поскольку в нём мы сначала отфильтруем таблицы, что является довольно быстрой операцией, а потом будет соединять. В результате будет соединяться не так много данных, как могло быть, если бы мы не отфильтровали таблицы перед этим.

При добавлении индекса планы не изменятся, так как он повлияет лишь на эффективность фильтрации данных в таблице H_ВЕДОМОСТИ.

Запрос 2

План 1

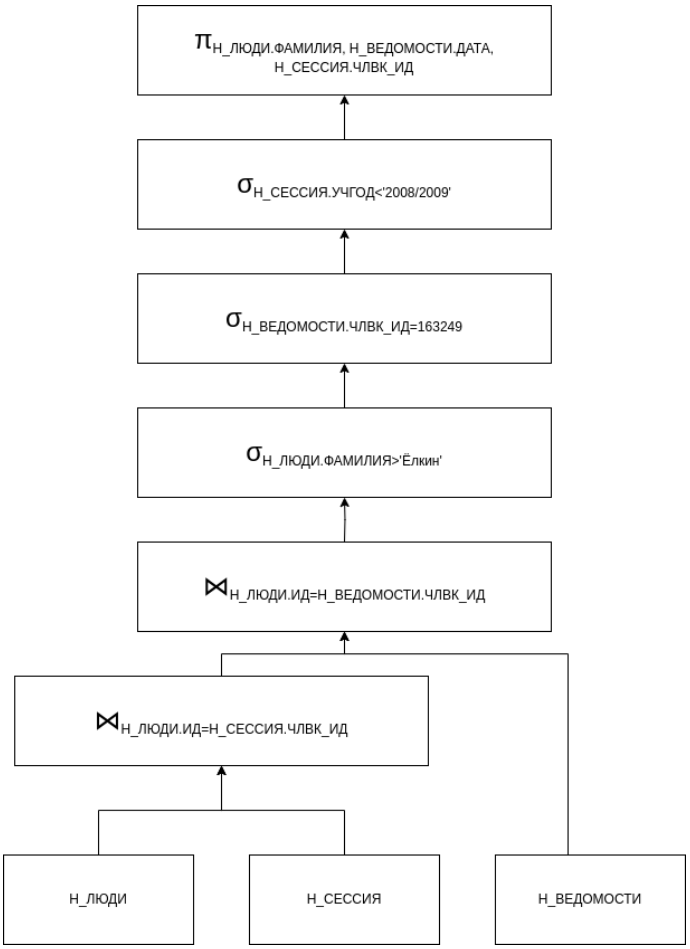

Рисунок 5

План 2

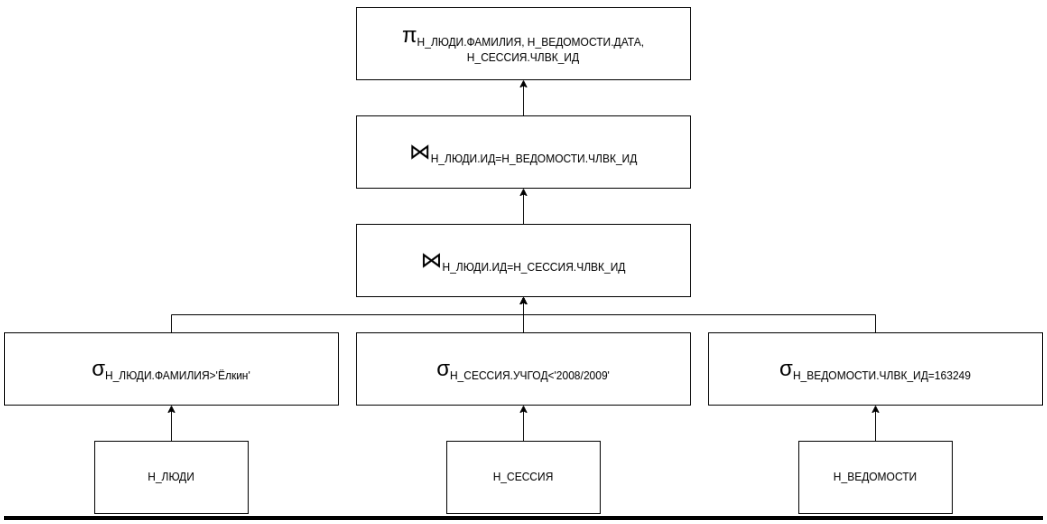

Рисунок 6

Можно составить ещё много вариантов последовательного расположения операций фильтрации и соединений, но все дальнейшие комбинации не будут эффективны, поэтому их можно не изображать. Опять же, оптимальным планом будет план 2, потому что в нём данные сначала фильтруются, а потом используются в соединении.

При добавлении индекса планировщик запроса обнаружит, что благодаря составному индексу по атрибутам `ФАМИЛИЯ`, `ИД` в таблице `Н_ЛЮДИ` выгоднее сделать фильтрацию по нему, взяв за значение `ИД` значение внешнего ключа `Н_ВЕДОМОСТИ.ЧЛВК_ИД`. Также индекс в таблице `Н_СЕССИЯ` по атрибуту `ЧЛВК_ИД` будет использован для фильтрации по этому значению. Тогда план 2 будет выглядеть следующим образом:

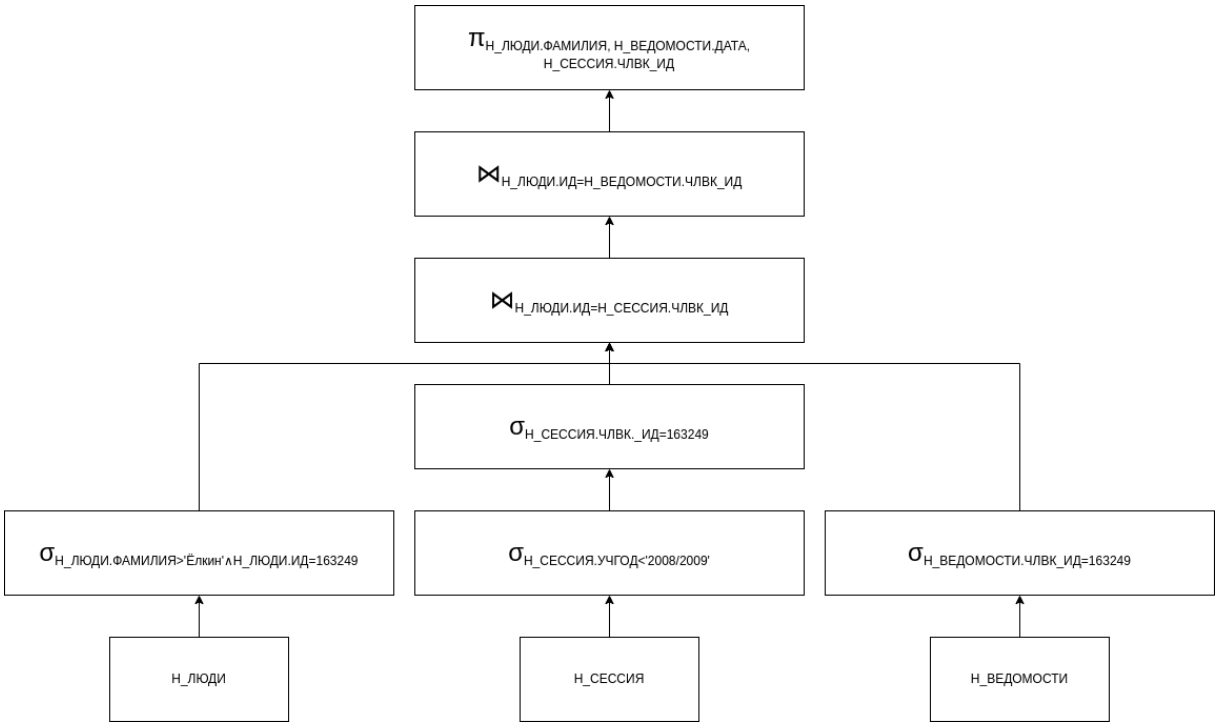

Рисунок 7

Анализ выбора планировщика

Далее будет представлен результат вывода команды `EXPLAIN ANALYZE` для исходных запросов.

Запрос 1

```
Nested Loop  (cost=0.29..217.04 rows=24 width=8) (actual time=0.006..0.007 rows=0 loops=1)
  Join Filter: ("Н_ВЕДОМОСТИ"."ТВ_ИД" = "Н_ТИПЫ_ВЕДОМОСТЕЙ"."ИД")
  →  Seq Scan on "Н_ТИПЫ_ВЕДОМОСТЕЙ"  (cost=0.00..1.04 rows=1 width=4) (actual time=0.006..0.006 rows=0 loops=1)
        Filter: (("НАИМЕНОВАНИЕ")::text < 'Ведомость'::text)
        Rows Removed by Filter: 3
  →  Index Scan using "ВЕД_ДАТА_I" on "Н_ВЕДОМОСТИ"  (cost=0.29..215.10 rows=72 width=8) (never executed)
        Index Cond: ("ДАТА" = '2022-06-08 00:00:00'::timestamp without time zone)
Planning Time: 0.171 ms
Execution Time: 0.020 ms
```

Рисунок 8

Запрос 2

```
Nested Loop  (cost=4.59..320.85 rows=64 width=28) (actual time=0.126..0.127 rows=0 loops=1)
  →  Nested Loop  (cost=4.30..123.67 rows=1 width=24) (actual time=0.126..0.127 rows=0 loops=1)
        →  Index Only Scan using "Н_ЛЮДИ_ФАМИЛИЯ_ИД_i" on "Н_ЛЮДИ"  (cost=0.28..112.62 rows=1 width=20) (actual time=0.037..0.118 rows=1 loops=1)
              Index Cond: ("ИД" = 163249)
              Heap Fetches: 0
        →  Bitmap Heap Scan on "Н_СЕССИЯ"  (cost=4.02..11.04 rows=1 width=4) (actual time=0.006..0.006 rows=0 loops=1)
              Recheck Cond: ("ЧЛВК_ИД" = 163249)
              Filter: (("УЧГОД")::text < '2008/2009'::text)
              →  Bitmap Index Scan on "Н_СЕССИЯ_ЧЛВК_ИД_i"  (cost=0.00..4.01 rows=2 width=0) (actual time=0.004..0.004 rows=0 loops=1)
                    Index Cond: ("ЧЛВК_ИД" = 163249)
  →  Index Scan using "ВЕД_ЧЛВК_FK_IFK" on "Н_ВЕДОМОСТИ"  (cost=0.29..196.53 rows=64 width=12) (never executed)
        Index Cond: ("ЧЛВК_ИД" = 163249)
Planning Time: 0.826 ms
Execution Time: 0.155 ms
```

Рисунок 9

Заключение

В результате выполнения данной лабораторной работы изучена теория оптимизации реляционных баз данных. Получены знания о планировании запросов, а также на примере реальных SQL-запросов проработаны возможные для них планы и выбраны эффективные. Главным объектом изучения стали индексы. Рассмотрены основные алгоритмы, на которых они могут быть построены, рамки их применения, дополнительные свойства. Индексы были применены к атрибутам некоторых таблиц БД, в результате чего исходные запросы начали выполняться быстрее.