Федеральное государственное автономное образовательное учреждение высшего образования
"Национальный Исследовательский Университет ИТМО"
Мегафакультет Компьютерных Технологий и Управления
Факультет Программной Инженерии и Компьютерной Техники

Лабораторная работа №3
по дисциплине
'Информационные системы и базы данных'
Вариант 564738291

Выполнил Студент группы P33102
Лапин Алексей Александрович
Преподаватель:
Сагайдак Алина Алексеевна

г. Санкт-Петербург
2023г.

Содержание

1 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 указанн
   Таблицы: Н_ЛЮДИ, Н_ВЕДОМОСТИ.
   Вывести атрибуты: Н_ЛЮДИ.ИМЯ, Н_ВЕДОМОСТИ.ЧЛВК_ИД.
   Фильтры (AND):
   a) Н_ЛЮДИ.ОТЧЕСТВО > Георгиевич.
   b) Н_ВЕДОМОСТИ.ИД = 1490007.
   Вид соединения: RIGHT JOIN.
2. Сделать запрос для получения атрибутов из указанных таблиц, применив фильтры по указанн
   Таблицы: Н_ЛЮДИ, Н_ВЕДОМОСТИ, Н_СЕССИЯ.
   Вывести атрибуты: Н_ЛЮДИ.ОТЧЕСТВО, Н_ВЕДОМОСТИ.ЧЛВК_ИД, Н_СЕССИЯ.ИД.
   Фильтры (AND):
   a) Н_ЛЮДИ.ФАМИЛИЯ = Ёлкин.
   b) Н_ВЕДОМОСТИ.ДАТА > 2022-06-08.
   c) Н_СЕССИЯ.УЧГОД > 2011/2012.
   Вид соединения: INNER JOIN.

2 Запрос 1

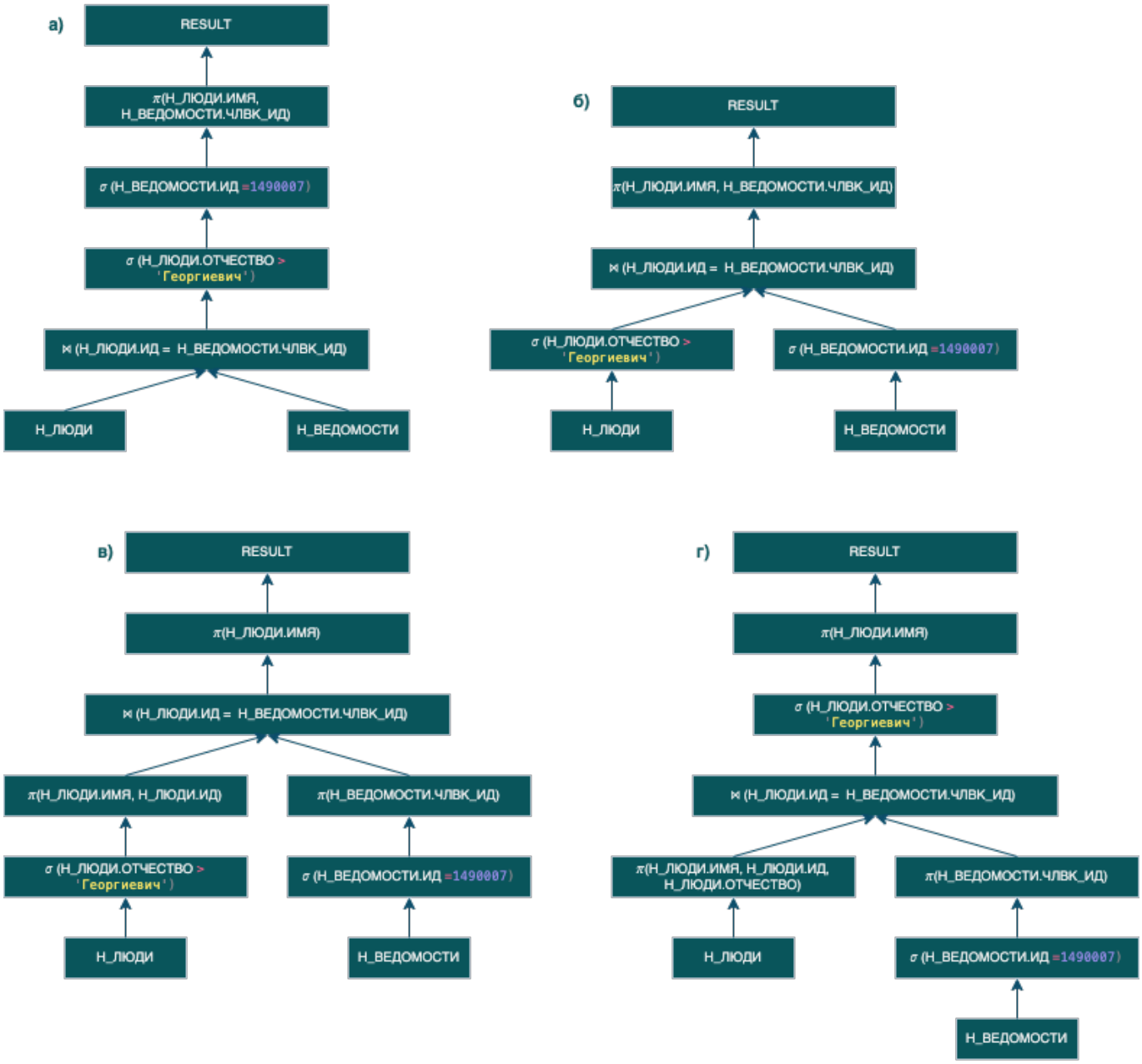
2.1 Реализация запроса на SQL.

```sql
-- Таблицы: Н_ЛЮДИ, Н_ВЕДОМОСТИ.
-- Вывести атрибуты: Н_ЛЮДИ.ИМЯ, Н_ВЕДОМОСТИ.ЧЛВК_ИД.
-- Фильтры (AND):
-- a) Н_ЛЮДИ.ОТЧЕСТВО > Георгиевич.
-- b) Н_ВЕДОМОСТИ.ИД = 1490007.
-- Вид соединения: RIGHT JOIN.

SELECT Н_ЛЮДИ.ИМЯ, Н_ВЕДОМОСТИ.ЧЛВК_ИД
FROM Н_ЛЮДИ
RIGHT JOIN Н_ВЕДОМОСТИ ON Н_ЛЮДИ.ИД = Н_ВЕДОМОСТИ.ЧЛВК_ИД
WHERE Н_ЛЮДИ.ОТЧЕСТВО > 'Георгиевич' AND Н_ВЕДОМОСТИ.ИД = 1490007
```

```
1  -- а) Индекс позволяет эффективно выбирать строки, основываясь на их поло
      жении в дереве, а также так как выражения в WHERE отбираются с использ
      ованием знака >.
2  CREATE INDEX Н_ЛЮДИ_ИМЯ ON Н_ЛЮДИ USING BTREE(ИМЯ)
3  -- б) создавать индекс для Н_ВЕДОМОСТИ.ИД не требуется так как для всех
      PRIMARY KEY по умолчанию создается таблица индексов,
4  -- но если бы потребовалось я бы использовал INDEX HASH, так как он эффек
      тивен на прямое сравнение ИД = HASH.
```

2.2 Планы выполнения запроса.



Я считаю, что в общем случае оптимальным вариантом является план в) так, как объединяются только нужные строки и колонки (после выборки и проекции). Тем самым мы уменьшаем размер промежуточных данных => уменьшаем число операций чтения записи во внешнюю

память.

```
QUERY PLAN
-------------------------------------------
 Nested Loop  (cost=0.70..16.75 rows=1 width=17) (actual
    time=0.041..0.042 rows=0 loops=1)
   ->  Index Scan using "ВЕД_РК" on "Н_ВЕДОМОСТИ"  (cost=0.42..8.44
      rows=1 width=4) (actual time=0.026..0.026 rows=1 loops=1)
         Index Cond: ("ИД" = 1490007)
   ->  Index Scan using "ЧЛВК_РК" on "Н_ЛЮДИ"  (cost=0.28..8.30 rows=1
      width=17) (actual time=0.009..0.009 rows=0 loops=1)
         Index Cond: ("ИД" = "Н_ВЕДОМОСТИ"."ЧЛВК_ИД")
         Filter: (("ОТЧЕСТВО")::text > 'Георгиевич'::text)
         Rows Removed by Filter: 1
 Planning Time: 1.634 ms
 Execution Time: 0.115 ms
(9 строк)
```

Мы видим, что Postgresql выбрал вариант г), так как в данном случае после выборки Н_ВЕДОМОСТИ.ИД осталась всего одна строка. В таком случае меньше операций будет, если мы сначала соединим эту одну строку со строками Н_ЛЮДИ (так как мы делаем RIGHT JOIN) и только потом отфильтруем Н_ЛЮДИ.
В данном случае план не измениться после добавления индекса, но скорость выполнения увеличится.

3 Запрос 2

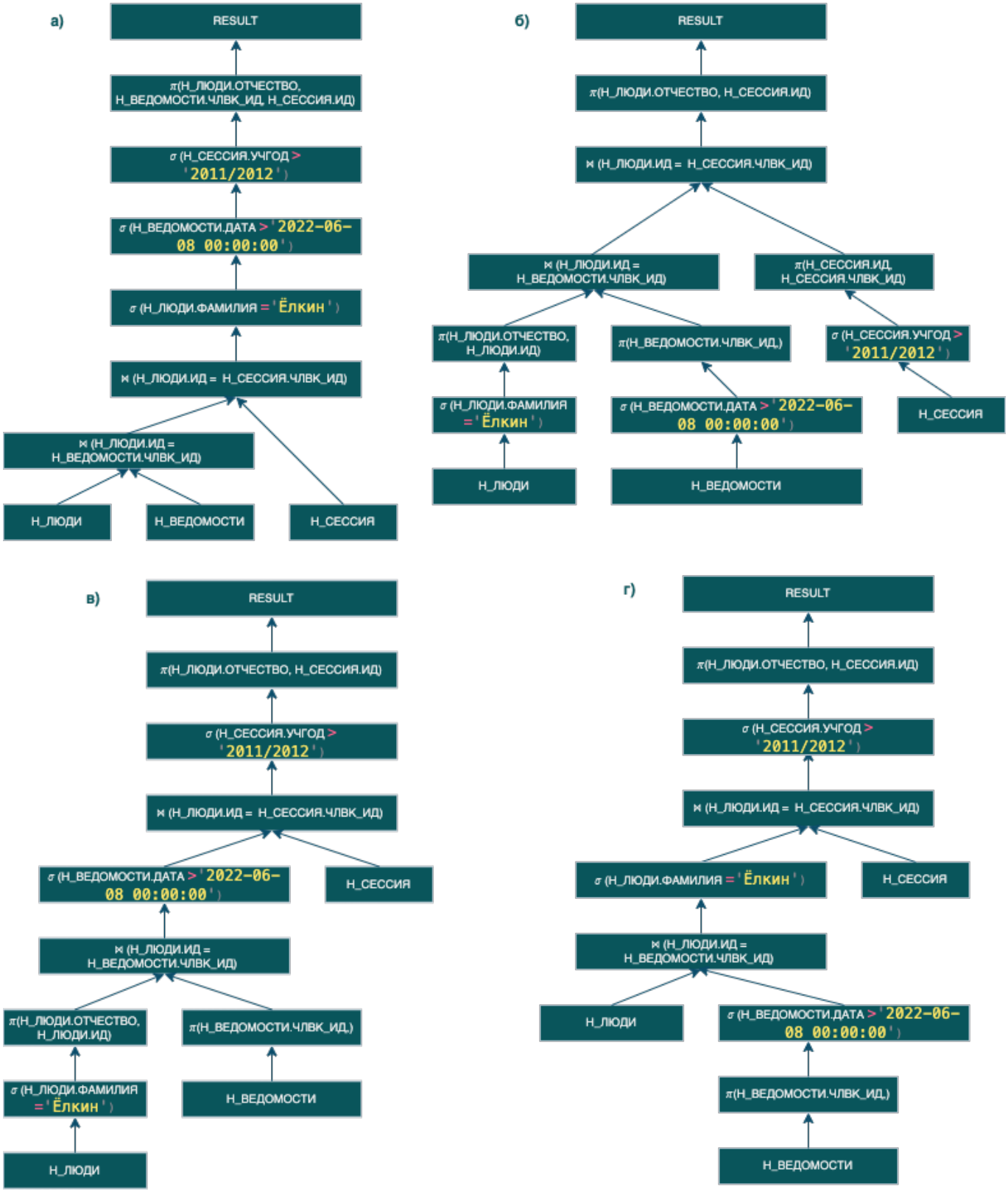
3.1 Реализация запроса на SQL.

```
-- Таблицы: Н_ЛЮДИ, Н_ВЕДОМОСТИ, Н_СЕССИЯ.
-- Вывести атрибуты: Н_ЛЮДИ.ОТЧЕСТВО, Н_ВЕДОМОСТИ.ЧЛВК_ИД, Н_СЕССИЯ.ИД.
-- Фильтры (AND):
-- a) Н_ЛЮДИ.ФАМИЛИЯ = Ёлкин.
-- b) Н_ВЕДОМОСТИ.ДАТА > 2022-06-08.
-- c) Н_СЕССИЯ.УЧГОД > 2011/2012.
-- Вид соединения: INNER JOIN.

SELECT Н_ЛЮДИ.ОТЧЕСТВО, 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_ЛЮДИ.ИД = Н_СЕССИЯ.ЧЛВК_ИД
WHERE
    Н_ЛЮДИ.ФАМИЛИЯ = 'Ёлкин' AND
    Н_ВЕДОМОСТИ.ДАТА > TIMESTAMP '2022-06-08 00:00:00' AND
    Н_СЕССИЯ.УЧГОД > '2011/2012';
```

```
-- a) Индекс позволяет эффективно выбирать строки, основываясь на значени
   и хеша столбца.
CREATE INDEX Н_ЛЮДИ_ФАМИЛИЯ ON Н_ЛЮДИ USING HASH(ФАМИЛИЯ)
```

```
3  -- б) Индекс позволяет эффективно выбирать строки, основываясь на их поло
      жении в дереве, а также так как выражения в WHERE отбираются с использ
      ованием знака >.
4  CREATE INDEX Н_ВЕДОМОСТИ_ДАТА ON Н_ВЕДОМОСТИ USING BTREE(ДАТА)
5  -- в) Индекс позволяет эффективно выбирать строки, основываясь на их поло
      жении в дереве, а также так как выражения в WHERE отбираются с использ
      ованием знака >.
6  CREATE INDEX Н_СЕССИЯ_УЧГОД ON Н_СЕССИЯ USING BTREE(УЧГОД)
```



В общем случае самым оптимальным был бы план б), но в данном случае если использовать план г) то после фильтрации ведомостей мы получим ноль строк и закончим выполнение раньше.

При добавлении предложенных индексов скорость выполнения увеличится и план г) все равно остается самым быстрым.

```
1  QUERY PLAN
```

```
 ----------------------------------------------
  Nested Loop  (cost=4.99..61.42 rows=1 width=28) (actual
     time=0.047..0.048 rows=0 loops=1)
    ->  Nested Loop  (cost=0.58..15.51 rows=1 width=28) (actual
        time=0.046..0.047 rows=0 loops=1)
          Join Filter: ("Н_ЛЮДИ"."ИД" = "Н_ВЕДОМОСТИ"."ЧЛВК_ИД")
          ->  Index Scan using "ФАМ_ЛЮД" on "Н_ЛЮДИ"  (cost=0.28..8.30
              rows=1 width=24) (actual time=0.033..0.033 rows=1 loops=1)
                Index Cond: (("ФАМИЛИЯ")::text = 'Ёлкин'::text)
          ->  Index Scan using "ВЕД_ДАТА_I" on "Н_ВЕДОМОСТИ"
              (cost=0.29..7.20 rows=1 width=4) (actual time=0.009..0.010
              rows=0 loops=1)
                Index Cond: ("ДАТА" > '2022-06-08 00:00:00'::timestamp
                   without time zone)
    ->  Bitmap Heap Scan on "Н_СЕССИЯ"  (cost=4.42..45.90 rows=1 width=8)
        (never executed)
          Recheck Cond: ("ЧЛВК_ИД" = "Н_ЛЮДИ"."ИД")
          Filter: (("УЧГОД")::text > '2011/2012'::text)
          ->  Bitmap Index Scan on "SYS_C003500_IFK"  (cost=0.00..4.42
              rows=18 width=0) (never executed)
                Index Cond: ("ЧЛВК_ИД" = "Н_ЛЮДИ"."ИД")
  Planning Time: 1.960 ms
  Execution Time: 0.142 ms
 (14 строк)
```

4 Выводы по работе.

В результате выполнения лабораторной работы были разработаны и проанализированы два SQL запроса и планы их выполнения. В ходе выполнения были изучены особенности составления и обработки планов СУБД PostgreSQL при использовании и без использования индексов. Были изучены основные виды индексов и стратегии соединения таблиц, применяемых в СУБД.