Университет ИТМО

Факультет ПИиКТ

Дисциплина: Информационные системы и базы данных

Лабораторная работа №4.

Вариант 8435

Выполнила: Мозговая Лариса Андреевна,

группа P33311

Преподаватель: Николаев Владимир Вячеславович

г. Санкт-Петербург, 2023 год

Задание к лабораторной работе:

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_ОЦЕНКИ, Н_ВЕДОМОСТИ.
   Вывести атрибуты: Н_ОЦЕНКИ.КОД, Н_ВЕДОМОСТИ.ИД.
   Фильтры (AND):
   a) Н_ОЦЕНКИ.КОД < неявка.
   b) Н_ВЕДОМОСТИ.ЧЛВК_ИД < 105590.
   c) Н_ВЕДОМОСТИ.ЧЛВК_ИД > 163249.
   Вид соединения: LEFT JOIN.
2. Сделать запрос для получения атрибутов из указанных таблиц, применив фильтры по указанным условиям:
   Таблицы: Н_ЛЮДИ, Н_ОБУЧЕНИЯ, Н_УЧЕНИКИ.
   Вывести атрибуты: Н_ЛЮДИ.ИМЯ, Н_ОБУЧЕНИЯ.НЗК, Н_УЧЕНИКИ.НАЧАЛО.
   Фильтры: (AND)
   a) Н_ЛЮДИ.ИД > 152862.
   b) Н_ОБУЧЕНИЯ.НЗК = 933232.
   Вид соединения: LEFT JOIN.

1.

```
select Н_ОЦЕНКИ.КОД, Н_ВЕДОМОСТИ.ИД
from Н_ОЦЕНКИ
left join Н_ВЕДОМОСТИ on Н_ОЦЕНКИ.КОД = Н_ВЕДОМОСТИ.ОЦЕНКА
where  Н_ОЦЕНКИ.КОД < 'неявка'
     and Н_ВЕДОМОСТИ.ЧЛВК_ИД < 105590
     and Н_ВЕДОМОСТИ.ЧЛВК_ИД > 163249
```

Результат:

```
 КОД | ИД
-----+----
(0 строк)
```

Для первого запроса целесообразно создать следующие индексы:

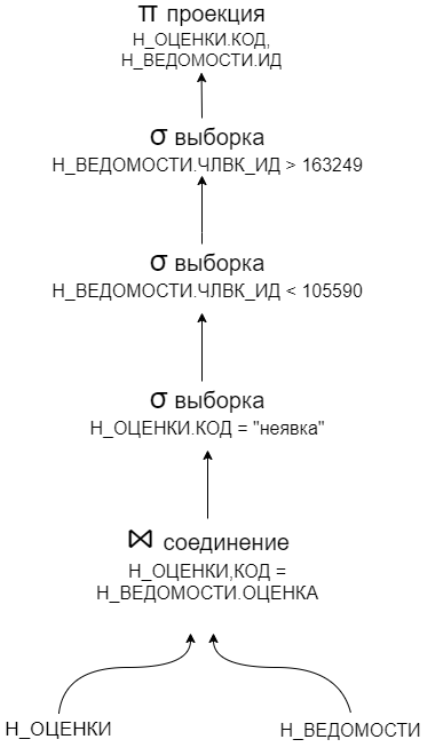
```
create index points on Н_ОЦЕНКИ using btree(КОД);
create index statements on Н_ВЕДОМОСТИ using btree(ЧЛВК_ИД);
```

Для таблицы Н_ОЦЕНКИ для атрибута КОД целесообразно создать индекс B-tree, потому что в запросе используется не только оператор «=», но и «<», который не поддерживается индексом. Добавление индекса ускорит выполнение операций WHERE и ON.

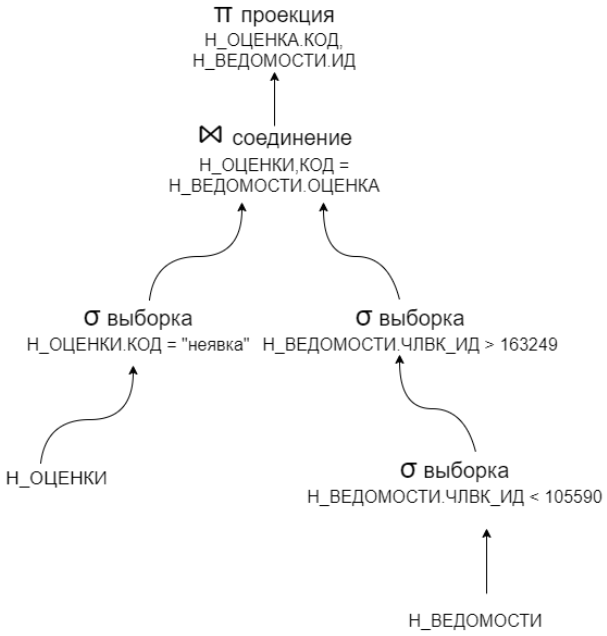
Для таблицы Н_ВЕДОМОСТИ для атрибута ЧЛВК_ИД целесообразно создать индекс B-tree, потому что в запросе используется и оператор «<», который не поддерживается индексом. При этом алгоритмическая сложность будет – O(logN).

План выполнения:

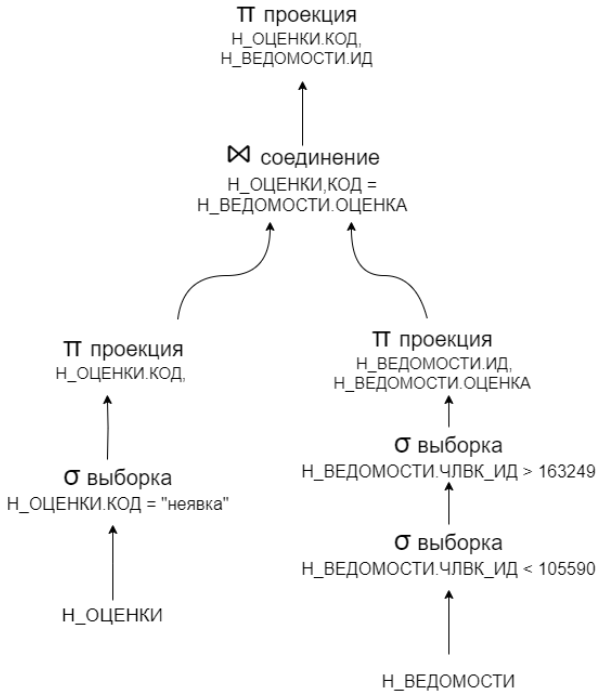
1)



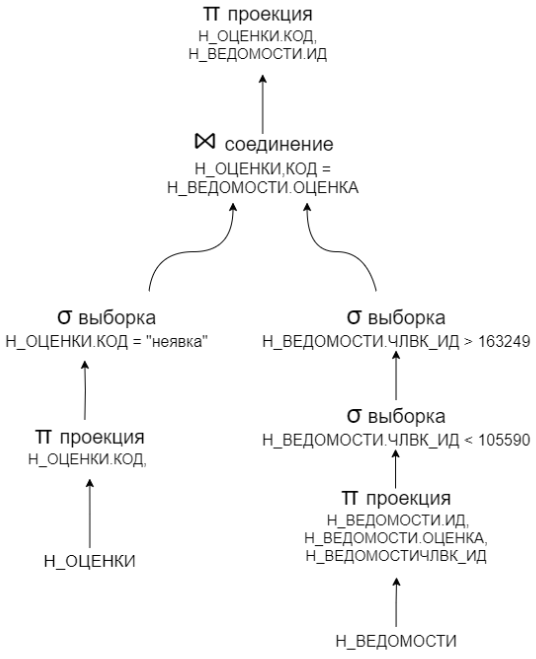
2)



3)



4)



Из составленных возможных планов выполнения запроса лучшим является четвертый, поскольку в нем изначально получаются проекции и производятся выборки, а уже после этого выполняется соединение. Это позволяет уменьшить размер хранимых данных.

При создании индексов четвертый план останется оптимальным, при этом даже ускорится за счёт ускорения поиска. Также довольно эффективным будет третий план. В нем также соединение происходит после выборки, что позволяет ускорить выполнение запроса, но, в отличие от четвертого плана, в нем выборка будет происходить по всей таблице целиком, а не по отдельным проекциям.

```
Nested Loop  (cost=15.84..2535.25 rows=865 width=9) (actual time=0.010..0.011 rows=0 loops=1)
  ->  Bitmap Heap Scan on "Н_ВЕДОМОСТИ"  (cost=15.69..2507.21 rows=1112 width=10) (actual time=0.009..0.009 rows=0 loops=1)
        Recheck Cond: (("ЧЛВК_ИД" < 105590) AND ("ЧЛВК_ИД" > 163249))
        ->  Bitmap Index Scan on "ВЕД_ЧЛВК_FK_IFK"  (cost=0.00..15.42 rows=1112 width=0) (actual time=0.004..0.004 rows=0 loops=1)
              Index Cond: (("ЧЛВК_ИД" < 105590) AND ("ЧЛВК_ИД" > 163249))
  ->  Memoize  (cost=0.15..0.18 rows=1 width=5) (never executed)
        Cache Key: "Н_ВЕДОМОСТИ"."ОЦЕНКА"
        Cache Mode: logical
        ->  Index Only Scan using "ОЦ_РК" on "Н_ОЦЕНКИ"  (cost=0.14..0.17 rows=1 width=5) (never executed)
              Index Cond: (("КОД" = ("Н_ВЕДОМОСТИ"."ОЦЕНКА")::text) AND ("КОД" < 'неявка'::text))
              Heap Fetches: 0
Planning Time: 1.264 ms
Execution Time: 0.090 ms
(13 строк)
```

Изменю запрос, чтобы он хоть что-то выводил

```
select Н_ОЦЕНКИ.КОД, Н_ВЕДОМОСТИ.ИД
from Н_ОЦЕНКИ
left join Н_ВЕДОМОСТИ on Н_ОЦЕНКИ.КОД = Н_ВЕДОМОСТИ.ОЦЕНКА
where  Н_ОЦЕНКИ.КОД < 'неявка'
     and Н_ВЕДОМОСТИ.ЧЛВК_ИД  > 105590;
```

```
                                                                 QUERY PLAN
--------------------------------------------------------------------------------------------------------------------------------------------
 Hash Join  (cost=1.20..7706.31 rows=172958 width=9) (actual time=0.049..101.745 rows=220885 loops=1)
   Hash Cond: (("Н_ВЕДОМОСТИ"."ОЦЕНКА")::text = ("Н_ОЦЕНКИ"."КОД")::text)
   ->  Seq Scan on "Н_ВЕДОМОСТИ"  (cost=0.00..6846.50 rows=222375 width=10) (actual time=0.010..41.250 rows=222413 loops=1)
         Filter: ("ЧЛВК_ИД" > 105590)
         Rows Removed by Filter: 27
   ->  Hash  (cost=1.11..1.11 rows=7 width=5) (actual time=0.017..0.019 rows=7 loops=1)
         Buckets: 1024  Batches: 1  Memory Usage: 9kB
         ->  Seq Scan on "Н_ОЦЕНКИ"  (cost=0.00..1.11 rows=7 width=5) (actual time=0.006..0.009 rows=7 loops=1)
               Filter: (("КОД")::text < 'неявка'::text)
               Rows Removed by Filter: 2
 Planning Time: 1.225 ms
 Execution Time: 112.289 ms
(12 строк)
```

Теперь можно увидеть, что действительно все соответствует плану. Мы выбираем только нужные значения Seq Scan on Н_ВЕДОМОСТИ, затем создается Hash таблица для Н_ОЦЕНКИ по условию фильтрации. Дальше опять Seq Scan выбираем нужное, и затем происходит Hash join, далее выдается результат.

2.

```
select Н_ЛЮДИ.ИМЯ, Н_ОБУЧЕНИЯ.НЗК, Н_УЧЕНИКИ.НАЧАЛО
from Н_ЛЮДИ
    left join Н_ОБУЧЕНИЯ on Н_ЛЮДИ.ИД = Н_ОБУЧЕНИЯ.ЧЛВК_ИД
    left join Н_УЧЕНИКИ on Н_УЧЕНИКИ.ИД = Н_ОБУЧЕНИЯ.ЧЛВК_ИД
where Н_ЛЮДИ.ИД > 153862
    and Н_ОБУЧЕНИЯ.НЗК::integer = 933232;
```

Результат:

```
 ИМЯ | НЗК | НАЧАЛО
-----+-----+--------
(0 строк)
```

Для второго запроса целесообразно создать следующие индексы:

```
create index hlvk_id_index on Н_ОБУЧЕНИЯ using hash(ЧЛВК_ИД);
create index id_index on Н_ЛЮДИ using btree(ИД);
create index nzk_index on Н_ОБУЧЕНИЯ using hash(НЗК);
```

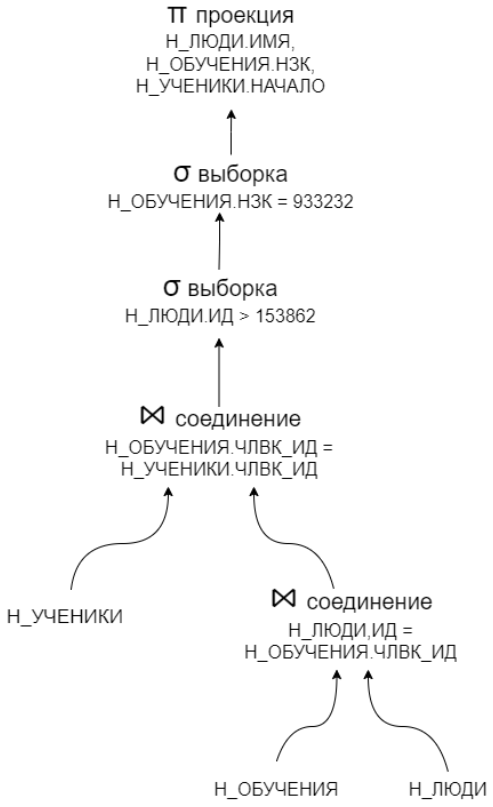
Для таблицы Н_ОБУЧЕНИЯ для атрибута ЧЛВК_ИД целесообразно создать индекс Hash, потому что в запросе используется только оператор «=». Создание данного индекса ускорить операции соединения таблиц.

Для таблицы Н_ЛЮДИ для атрибута ИМЯ целесообразно создать индекс B-tree, потому что помимо оператора «=» в запросе используется и оператор «>». Создание данного индекса ускорить операцию WHERE.

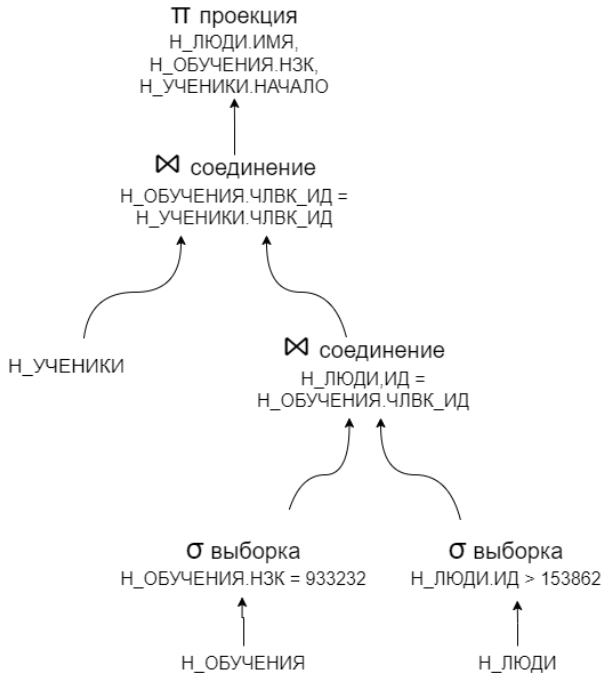
Для таблицы Н_ОБУЧЕНИЯ для атрибута НЗК целесообразно создать индекс Hash, потому что в запросе используется только оператор «=». Создание данного индекса ускорить операцию WHERE.

План выполнения:

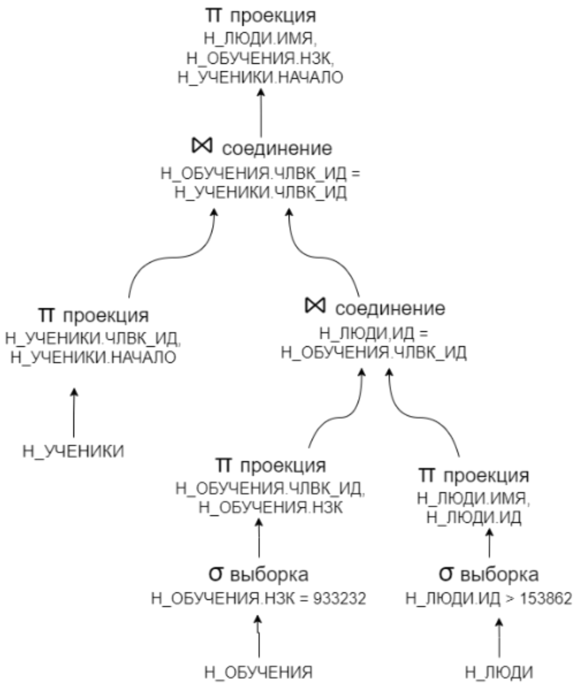
1)



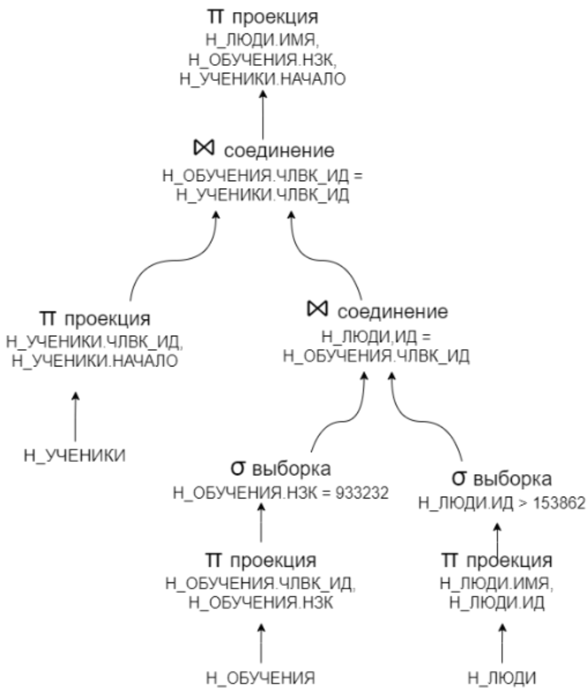
2)



3)



4)



Из составленных возможных планов выполнения запроса лучшим является четвертый, поскольку в нем изначально получаются проекции и производятся выборки, а уже после этого выполняется соединение. Это позволяет уменьшить размер хранимых данных.

При создании индексов четвертый план останется оптимальным, при этом даже ускорится за счёт ускорения поиска. Также довольно эффективным будет третий план. В нем также соединение происходит после выборки, что позволяет ускорить выполнение запроса, но, в отличие от четвертого плана, в нем выборка будет происходить по всей таблице целиком, а не по отдельным проекциям.

```
                                                   QUERY PLAN
------------------------------------------------------------------------------------------------------------------------------------
 Nested Loop Left Join  (cost=158.41..292.14 rows=3 width=27) (actual time=0.974..0.976 rows=0 loops=1)
   ->  Hash Join  (cost=158.12..268.67 rows=3 width=23) (actual time=0.973..0.975 rows=0 loops=1)
         Hash Cond: ("Н_ЛЮДИ"."ИД" = "Н_ОБУЧЕНИЯ"."ЧЛВК_ИД")
         ->  Bitmap Heap Scan on "Н_ЛЮДИ"  (cost=12.94..120.45 rows=601 width=17) (actual time=0.043..0.187 rows=593 loops=1)
               Recheck Cond: ("ИД" > 153862)
               Heap Blocks: exact=35
               ->  Bitmap Index Scan on "ЧЛВК_РК"  (cost=0.00..12.79 rows=601 width=0) (actual time=0.034..0.034 rows=593 loops=1)
                     Index Cond: ("ИД" > 153862)
         ->  Hash  (cost=144.87..144.87 rows=25 width=10) (actual time=0.712..0.713 rows=1 loops=1)
               Buckets: 1024  Batches: 1  Memory Usage: 9kB
               ->  Seq Scan on "Н_ОБУЧЕНИЯ"  (cost=0.00..144.87 rows=25 width=10) (actual time=0.015..0.699 rows=1 loops=1)
                     Filter: (("НЗК")::integer = 933232)
                     Rows Removed by Filter: 5020
   ->  Index Scan using "УЧЕН_РК" on "Н_УЧЕНИКИ"  (cost=0.29..7.82 rows=1 width=12) (never executed)
         Index Cond: ("ИД" = "Н_ОБУЧЕНИЯ"."ЧЛВК_ИД")
 Planning Time: 0.558 ms
 Execution Time: 1.037 ms
(17 строк)
```

Сначала используется битовая карта для определения строк в таблице Н_ЛЮДИ, которые соответствуют условиям ИД > 153862. Затем происходит hash соединение с таблицей Н_ОБУЧЕНИЯ по условию Н_ЛЮДИ.ИД = Н_ОБУЧЕНИЯ.ЧЛВК_ИД. Дальше опять Seq Scan выбираем нужное, и затем происходит Hash join. После этого должно выполниться вложенное левое соединение с таблицей "Н_УЧЕНИКИ".

Вывод:

В ходе выполнения данной лабораторной работы я познакомилась с индексами, тем, как они влияют на нагрузку на систему. Также я познакомилась с планом выполнения запроса, узнала, каким образом СУБД выбирает оптимальный. Узнала, что выполняет команда EXPLAIN ANALYZE.