Университет ИТМО
Факультет программной инженерии и компьютерной техники

Информационные системы и базы данных

Лабораторная работа №3

Выполнила: Машина Е.А.

Группа P33113

Преподаватель: Николаев В.В.

Санкт-Петербург
2020 г.

Задание

Please, enter your variant: 652385

Внимание! У разных вариантов разный 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?

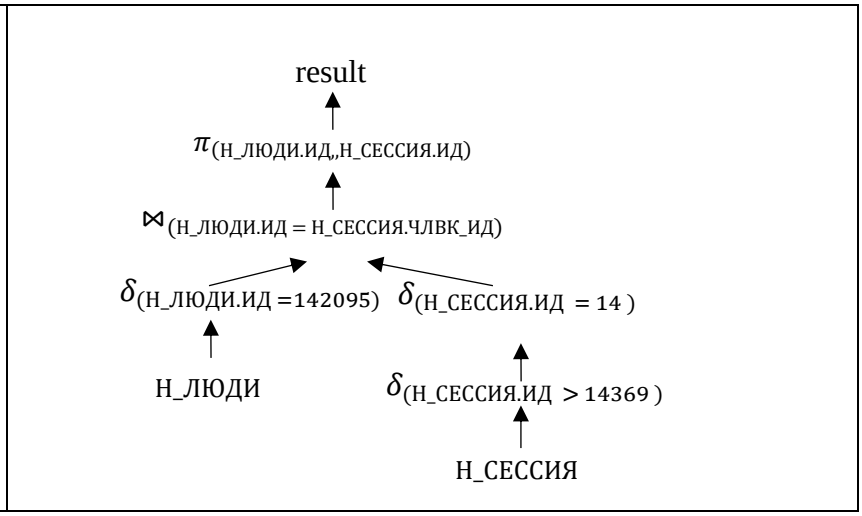
Для запросов 1-2 необходимо добавить в отчет вывод команды EXPLAIN ANALYZE [запрос]

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.ИД, Н_СЕССИЯ.ИД.
   Фильтры (AND):
   a) Н_ЛЮДИ.ИД = 142095.
   b) Н_СЕССИЯ.ИД > 14369.
   c) Н_СЕССИЯ.ИД = 14.
   Вид соединения: RIGHT 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.ОТЧЕСТВО, Н_ОБУЧЕНИЯ.ЧЛВК_ИД, Н_УЧЕНИКИ.ГРУППА.
   Фильтры: (AND)
   a) Н_ЛЮДИ.ИМЯ = Ярослав.
   b) Н_ОБУЧЕНИЯ.НЗК = 933232.
   Вид соединения: INNER JOIN.

Выполнение

```
    --1--
1 SELECT "Н_ЛЮДИ"."ИД", "Н_СЕССИЯ"."ИД"
2 FROM "Н_ЛЮДИ"
3          RIGHT JOIN "Н_СЕССИЯ" ON "Н_ЛЮДИ"."ИД" = "Н_СЕССИЯ"."ЧЛВК_ИД"
4 WHERE "Н_ЛЮДИ"."ИД" = 142095
5   AND "Н_СЕССИЯ"."ИД" > 14369
6   AND "Н_СЕССИЯ"."ИД" = 14;
    /*
     ФАМИЛИЯ | ИД
    ---------+----
    (0 rows)
    */
```



Оптимальным планом является второй, т.к. мы объединяем не целые сущности, а только нужные нам выборки из них, следовательно, размер промежуточного отношения меньше.

Индексы:

```
CREATE INDEX "ЛЮДИ_ИД" ON "Н_ЛЮДИ" USING HASH ("ИД");
CREATE INDEX "ИД_ИНД" ON "Н_СЕССИЯ" USING BTREE ("ИД");
```

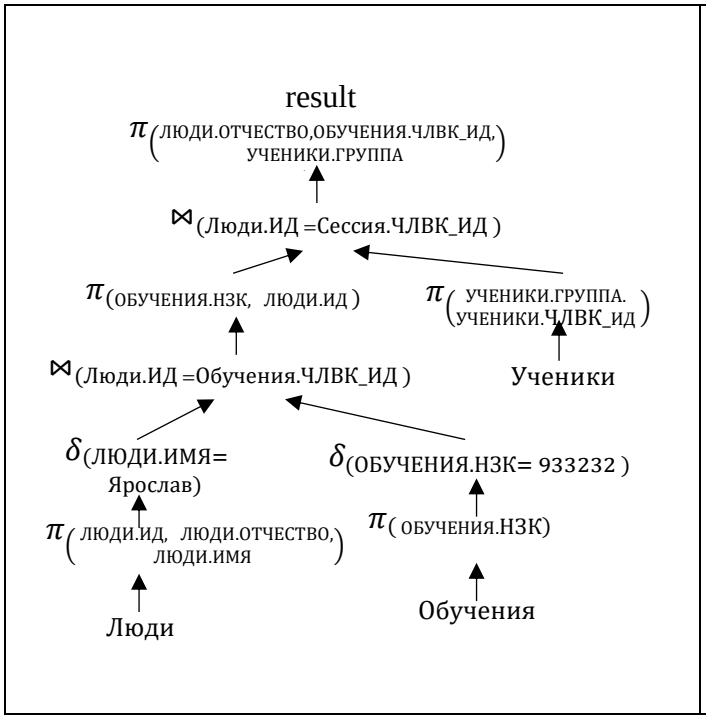
Добавление этих индексов может существенно ускорить запросы, т.к. по данным атрибутам идет выборка с использованием операторов = и >; соединение таблиц

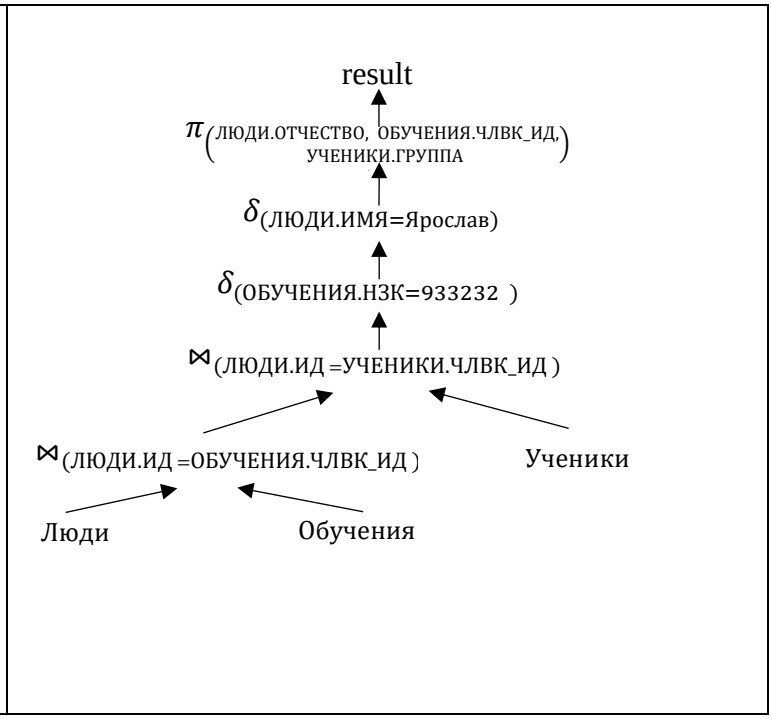
```
                                QUERY PLAN
     Nested Loop  (cost=4.62..34.17 rows=1 width=8) (actual time=0.074..0.074 rows=0
     loops=1)
     "  ->  Index Only Scan using ""ЧЛВК_PK"" on ""Н_ЛЮДИ""  (cost=0.28..8.30 rows=1
     width=4) (actual time=0.046..0.047 rows=1 loops=1)"
     "        Index Cond: (""ИД"" = 142095)"
            Heap Fetches: 1
     "  ->  Bitmap Heap Scan on ""Н_СЕССИЯ""  (cost=4.33..25.86 rows=1 width=8) (actual
     time=0.023..0.023 rows=0 loops=1)"
     "        Recheck Cond: (""ЧЛВК_ИД"" = 142095)"
     "        Filter: ((""ИД"" > 14369) AND (""ИД"" = 14))"
     "        ->  Bitmap Index Scan on ""SYS_C003500_IFK""  (cost=0.00..4.33 rows=7
     width=0) (actual time=0.022..0.022 rows=0 loops=1)"
     "              Index Cond: (""ЧЛВК_ИД"" = 142095)"
     Planning time: 0.324 ms
     Execution time: 0.107 ms
```

```
--2--
```

```
1 SELECT "Н_ЛЮДИ"."ОТЧЕСТВО", "Н_ОБУЧЕНИЯ"."ЧЛВК_ИД", "Н_УЧЕНИКИ"."ГРУППА"
2 FROM "Н_ЛЮДИ"
3          INNER JOIN "Н_ОБУЧЕНИЯ" ON "Н_ЛЮДИ"."ИД" = "Н_ОБУЧЕНИЯ"."ЧЛВК_ИД"
4          INNER JOIN "Н_УЧЕНИКИ" ON "Н_ЛЮДИ"."ИД" = "Н_УЧЕНИКИ"."ЧЛВК_ИД"
5 WHERE "Н_ЛЮДИ"."ИМЯ" = 'Ярослав'
6   AND "Н_ОБУЧЕНИЯ"."НЗК" = '933232';
```

```
/*
 ИМЯ | ЧЛВК_ИД | ГРУППА
---+---------+--------
(0 rows)
*/
```





Оптимальным является первый план. За счет раннего использования выборки и проекции происходит соединение не целых отношений, а только нужных нам атрибутов. Следовательно, промежуточные данные меньше.

Индексы:

```
CREATE INDEX "ЧЛВК_РК" ON "Н_ЛЮДИ" USING HASH ("ИД");
CREATE INDEX "ИМЯ_ИНД" ON "Н_ЛЮДИ" USING HASH ("ИМЯ");
CREATE INDEX "ЧЛВК_ИД_ИНД" ON "Н_ОБУЧЕНИЯ" USING HASH ("НЗК");
```

Добавление этих индексов может существенно ускорить запросы, т.к. по данным атрибутам идет выборка с использованием оператора =; соединение таблиц.

```
                                QUERY PLAN
    Unique  (cost=129.03..129.08 rows=5 width=27) (actual time=1.435..1.438 rows=3
loops=1)
      ->  Sort  (cost=129.03..129.04 rows=5 width=27) (actual time=1.434..1.434 rows=3
loops=1)
    "        Sort Key: ""Н_ЛЮДИ"".""ИМЯ"", ""Н_ОБУЧЕНИЯ"".""ЧЛВК_ИД"",
""Н_УЧЕНИКИ"".""ГРУППА"""
            Sort Method: quicksort  Memory: 25kB
            ->  Nested Loop  (cost=0.57..128.97 rows=5 width=27) (actual
time=0.112..1.324 rows=3 loops=1)
                  ->  Nested Loop  (cost=0.28..128.07 rows=1 width=27) (actual
time=0.076..1.253 rows=1 loops=1)
    "                    ->  Seq Scan on ""Н_ОБУЧЕНИЯ""  (cost=0.00..119.76 rows=1
width=10) (actual time=0.029..1.204 rows=1 loops=1)"
    "                          Filter: (""НЗК""::text = '933232'::text)"
                              Rows Removed by Filter: 5020
    "                    ->  Index Scan using ""ЧЛВК_РК"" on ""Н_ЛЮДИ""
(cost=0.28..8.30 rows=1 width=17) (actual time=0.039..0.039 rows=1 loops=1)"
    "                          Index Cond: (""ИД"" = ""Н_ОБУЧЕНИЯ"".""ЧЛВК_ИД"")"
    "              ->  Index Scan using ""УЧЕН_ОБУЧ_FK_I"" on ""Н_УЧЕНИКИ""
(cost=0.29..0.85 rows=5 width=8) (actual time=0.034..0.064 rows=3 loops=1)"
    "                    Index Cond: (""ЧЛВК_ИД"" = ""Н_ЛЮДИ"".""ИД"")"
    Planning time: 2.659 ms
    Execution time: 1.543 ms
```

Вывод

В ходе выполнения данной лабораторной работы я более подробно изучила язык SQL, совершила запросы к нескольким таблицам путем склеивания таблиц, познакомилась с понятием «индекс» и написала свои индексы.

Также поняла, что создание индексов не всегда ускоряет взаимодействие с базами данных, так как при изменении или удалении содержимого индексированного столбца (при добавлении новой строки) индекс необходимо обновлять. Эти действия замедляют операции. Также неэффективно использовать индексы в том случае, если в таблице мало строк, так как мы занимаем лишнюю память хранением индексов в тот момент, когда простой перебор сработает быстрее.

В контексте данной лабораторной работы я познакомилась с такими типами индексов, как btree и hash. Первый тип - это индекс, сгруппированный по листьям дерева поиска, эффективен при использовании совместно с =, >, >=, <=, BETWEEN и подобными операторами. Второй тип основан на построении хеш-функции и подходит в том случае, если у нас есть операция прямого сравнения.