УНИВЕРСИТЕТ ИТМО

Факультет программной инженерии и компьютерной техники
Кафедра вычислительной техники

Отчёт по лабораторной работе №4

Дисциплина: "Информационные системы и базы данных"

Преподаватель: Гаврилов Антон

Студент: Закиров Бобур

Группа: P33312

Санкт-Петербург
2021 г.

Задание

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]`

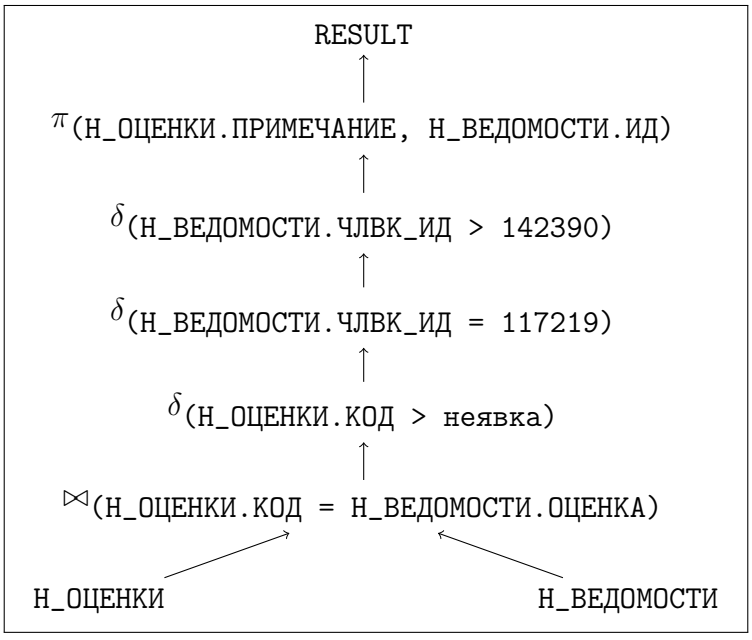
Выполнение

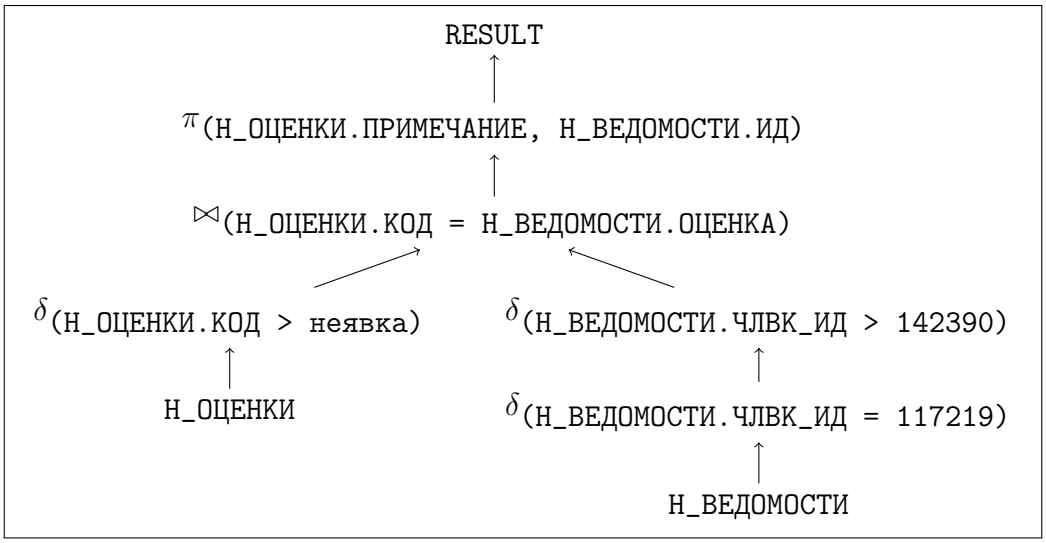
Запрос 1.

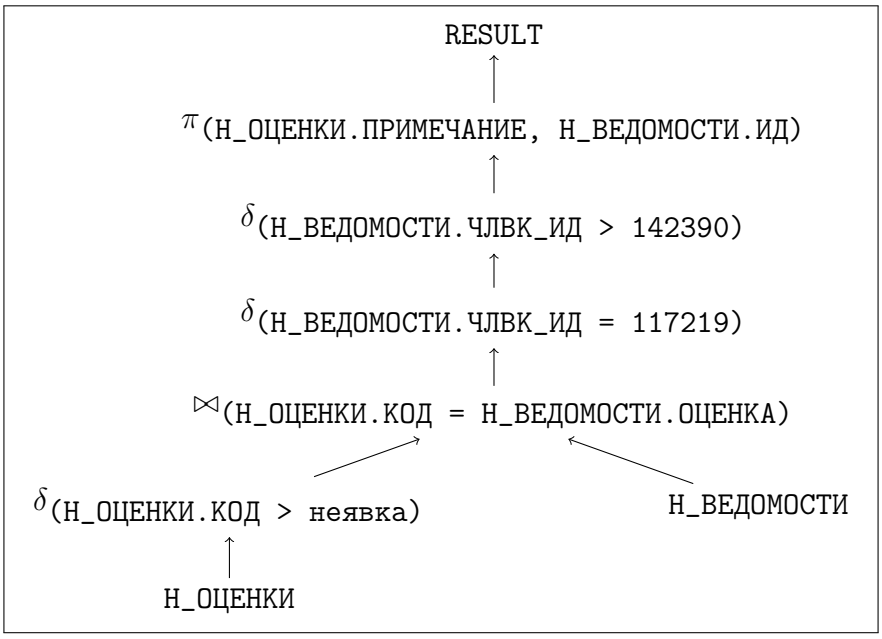
Сделать запрос для получения атрибутов из указанных таблиц, применив фильтры по указанным условиям:

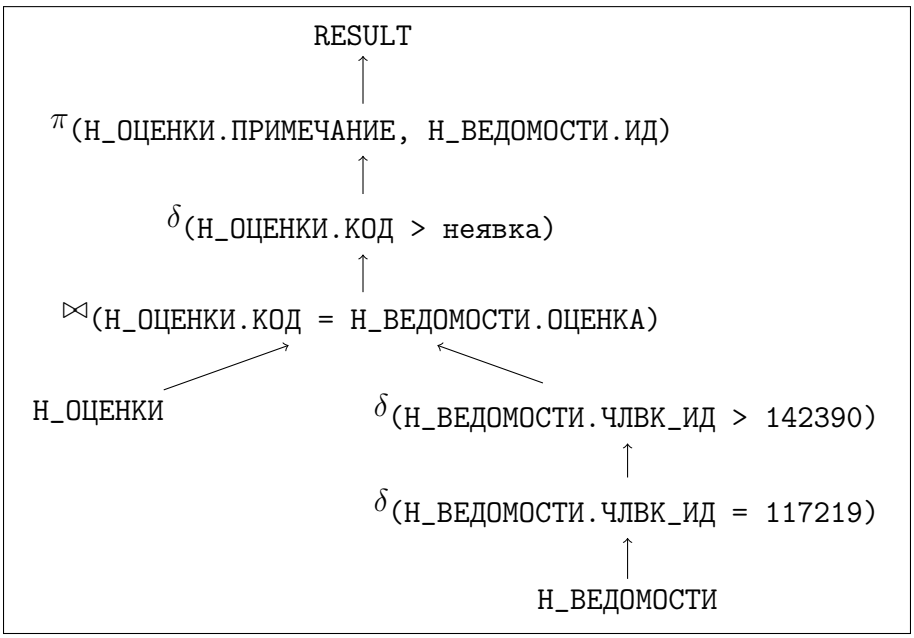
- Таблицы: `Н_ОЦЕНКИ`, `Н_ВЕДОМОСТИ`.
- Вывести атрибуты: `Н_ОЦЕНКИ.ПРИМЕЧАНИЕ`, `Н_ВЕДОМОСТИ.ИД`
- Фильтры `AND`:
  - `Н_ОЦЕНКИ.КОД > неявка`.
  - `Н_ВЕДОМОСТИ.ЧЛВК_ИД > 142390`.
  - `Н_ВЕДОМОСТИ.ЧЛВК_ИД = 117219`.
- Вид соединения: `RIGHT JOIN`.

Планы выполнения запроса




RESULT
π(Н_ОЦЕНКИ.ПРИМЕЧАНИЕ, Н_ВЕДОМОСТИ.ИД)
⋈(Н_ОЦЕНКИ.КОД = Н_ВЕДОМОСТИ.ОЦЕНКА)
δ(Н_ОЦЕНКИ.КОД > неявка)
δ(Н_ВЕДОМОСТИ.ЧЛВК_ИД > 142390)
Н_ОЦЕНКИ
δ(Н_ВЕДОМОСТИ.ЧЛВК_ИД = 117219)
Н_ВЕДОМОСТИ


RESULT
π(Н_ОЦЕНКИ.ПРИМЕЧАНИЕ, Н_ВЕДОМОСТИ.ИД)
δ(Н_ВЕДОМОСТИ.ЧЛВК_ИД > 142390)
δ(Н_ВЕДОМОСТИ.ЧЛВК_ИД = 117219)
⋈(Н_ОЦЕНКИ.КОД = Н_ВЕДОМОСТИ.ОЦЕНКА)
δ(Н_ОЦЕНКИ.КОД > неявка)
Н_ВЕДОМОСТИ
Н_ОЦЕНКИ


RESULT
π(Н_ОЦЕНКИ.ПРИМЕЧАНИЕ, Н_ВЕДОМОСТИ.ИД)
δ(Н_ОЦЕНКИ.КОД > неявка)
⋈(Н_ОЦЕНКИ.КОД = Н_ВЕДОМОСТИ.ОЦЕНКА)
Н_ОЦЕНКИ
δ(Н_ВЕДОМОСТИ.ЧЛВК_ИД > 142390)
δ(Н_ВЕДОМОСТИ.ЧЛВК_ИД = 117219)
Н_ВЕДОМОСТИ

Я привел четыре возможных плана выполнения для данного запроса:

- сначала идет процесс объединения таблиц, а затем выборка по их столбцам.
- сначала идет выборка для каждой таблицы отдельно, а затем процесс объединения результатов этих выборок.
- сначала идет процесс объединения результата выборки первой таблицы со второй таблицей, а затем выборка по столбцам второй таблицы.
- сначала идет процесс объединения первой таблицы с результатом выборки второй, а затем выборка по столбцам первой.

Я считаю, что среди этих планов оптимальном планом является второй, так как мы объединяем не все строки, а только нужные нам выборки из них, следовательно, размер промежуточного отношения меньше.

```
   EXPLAIN ANALYZE
    SELECT "ОЦЕНКИ"."ПРИМЕЧАНИЕ", "ВЕДОМОСТИ"."ИД"
      FROM (select * from "Н_ОЦЕНКИ" where "КОД" > 'неявка') AS "ОЦЕНКИ"
RIGHT JOIN (select * from "Н_ВЕДОМОСТИ" where "ЧЛВК_ИД" = 117219 and "ЧЛВК_ИД" > 142390) AS "ВЕДОМОСТИ"
        ON "ВЕДОМОСТИ"."ОЦЕНКА" = "ОЦЕНКИ"."КОД";
```

```
                                                    QUERY PLAN
-------------------------------------------------------------------------------------------------------------------------------
 Hash Left Join  (cost=1.57..45.40 rows=13 width=422) (actual time=0.005..0.005 rows=0 loops=1)
   Hash Cond: (("Н_ВЕДОМОСТИ"."ОЦЕНКА")::text = ("Н_ОЦЕНКИ"."КОД")::text)
   ->  Index Scan using "ВЕД_ЧЛВК_FK_IFK" on "Н_ВЕДОМОСТИ"  (cost=0.42..44.17 rows=13 width=10)
                                                              (actual time=0.004..0.004 rows=0 loops=1)
         Index Cond: (("ЧЛВК_ИД" > 142390) AND ("ЧЛВК_ИД" = 117219))
   ->  Hash  (cost=1.11..1.11 rows=3 width=452) (never executed)
         ->  Seq Scan on "Н_ОЦЕНКИ"  (cost=0.00..1.11 rows=3 width=452) (never executed)
               Filter: (("КОД")::text > 'неявка'::text)
 Planning time: 0.263 ms
 Execution time: 0.045 ms
(9 rows)
```

Индексы

```
CREATE INDEX "ИД_ВЕДОМОСТИ_ЧЛВК_ИД" ON "Н_ВЕДОМОСТИ" USING btree("ЧЛВК_ИД");
```

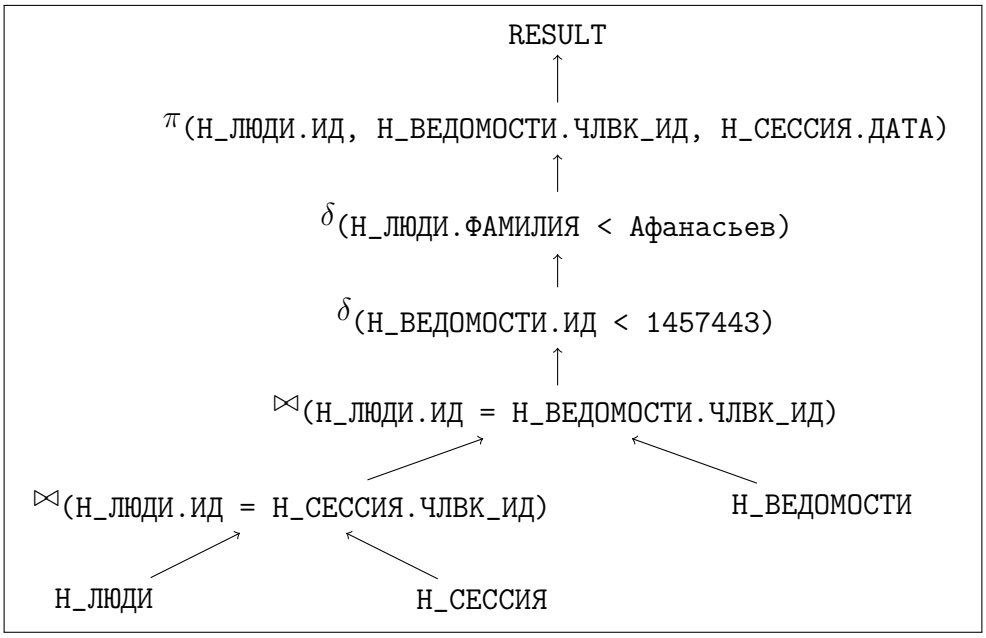
Добавление этого индекса может существенно ускорить запросы, т.к. по столбец `ЧЛВК_ИД` идет выборка с использованием операторов `=` и `>`, и данный индекс предоставляет возможность выполнять операции поиска за $O(\log n)$.

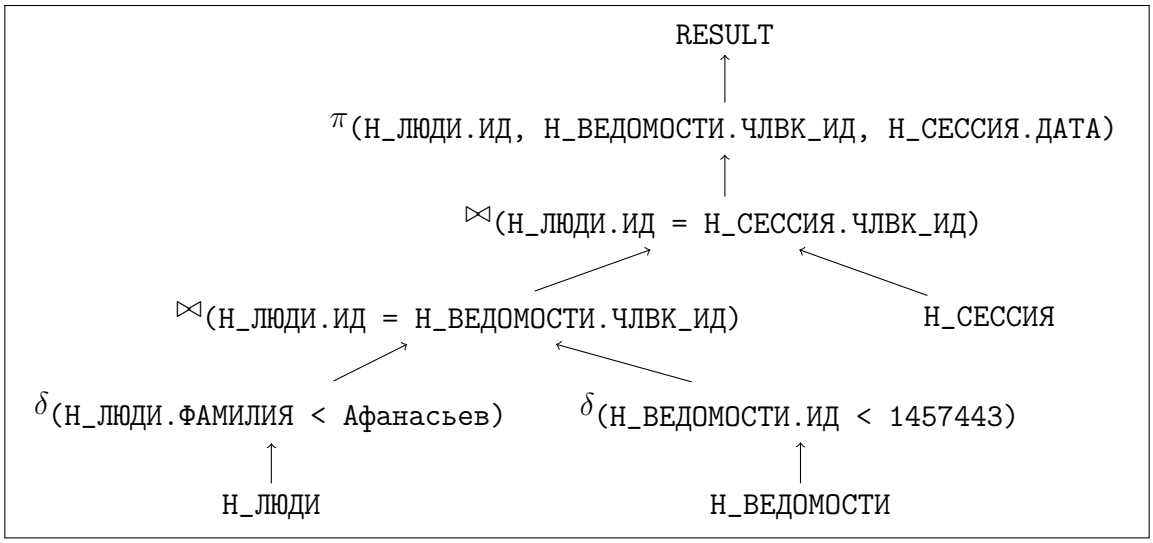
Запрос 2.

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

- Таблицы: `Н_ЛЮДИ`, `Н_ВЕДОМОСТИ`, `Н_СЕССИЯ`.
- Вывести атрибуты: `Н_ЛЮДИ.ИД`, `Н_ВЕДОМОСТИ.ЧЛВК_ИД`, `Н_СЕССИЯ.ДАТА`.
- Фильтры `AND`:
  - `Н_ЛЮДИ.ФАМИЛИЯ < Афанасьев`.
  - `Н_ВЕДОМОСТИ.ИД < 1457443`.
- Вид соединения: `RIGHT JOIN`.

Планы выполнения запроса





Оптимальным планом выполнения запроса является второй, так как происходит объединение только после необходимой выборки, вместо полного объединения таблиц.

```
   EXPLAIN ANALYZE
    SELECT "ЛЮДИ"."ИД", "ВЕДОМОСТИ"."ЧЛВК_ИД", "СЕССИЯ"."ДАТА"
      FROM (select * from "Н_ЛЮДИ" where "ФАМИЛИЯ" < 'Афанасьев') AS "ЛЮДИ"
RIGHT JOIN (select * from "Н_ВЕДОМОСТИ" where "ИД" < 1457443) AS "ВЕДОМОСТИ"
        ON "ЛЮДИ"."ИД" = "ВЕДОМОСТИ"."ЧЛВК_ИД"
RIGHT JOIN "Н_СЕССИЯ" AS "СЕССИЯ"
        ON "ЛЮДИ"."ИД" = "СЕССИЯ"."ЧЛВК_ИД";
```

```
                                                  QUERY PLAN
-----------------------------------------------------------------------------------------------------------------------
 Hash Left Join  (cost=7994.54..8373.23 rows=5912 width=16) (actual time=154.683..158.157 rows=3752 loops=1)
   Hash Cond: ("СЕССИЯ"."ЧЛВК_ИД" = "Н_ЛЮДИ"."ИД")
   ->  Seq Scan on "Н_СЕССИЯ" "СЕССИЯ"  (cost=0.00..108.52 rows=3752 width=12)
                                         (actual time=0.007..1.224 rows=3752 loops=1)
   ->  Hash  (cost=7877.76..7877.76 rows=9342 width=8) (actual time=154.653..154.653 rows=8450 loops=1)
         Buckets: 16384  Batches: 1  Memory Usage: 459kB
         ->  Hash Join  (cost=115.42..7877.76 rows=9342 width=8) (actual time=0.367..151.419 rows=8450 loops=1)
               Hash Cond: ("Н_ВЕДОМОСТИ"."ЧЛВК_ИД" = "Н_ЛЮДИ"."ИД")
               ->  Seq Scan on "Н_ВЕДОМОСТИ"  (cost=0.00..6846.50 rows=219312 width=4)
                                               (actual time=0.008..82.072 rows=219050 loops=1)
                     Filter: ("ИД" < 1457443)
                     Rows Removed by Filter: 3390
               ->  Hash  (cost=112.70..112.70 rows=218 width=4) (actual time=0.342..0.342 rows=214 loops=1)
                     Buckets: 1024  Batches: 1  Memory Usage: 16kB
                     ->  Bitmap Heap Scan on "Н_ЛЮДИ"  (cost=9.97..112.70 rows=218 width=4)
                                                        (actual time=0.084..0.261 rows=214 loops=1)
                           Recheck Cond: (("ФАМИЛИЯ")::text < 'Афанасьев'::text)
                           Heap Blocks: exact=78
                           ->  Bitmap Index Scan on "ФАМ_ЛЮД"  (cost=0.00..9.92 rows=218 width=0)
                                                                (actual time=0.074..0.074 rows=214 loops=1)
                                 Index Cond: (("ФАМИЛИЯ")::text < 'Афанасьев'::text)
 Planning time: 0.530 ms
 Execution time: 59.145 ms
(19 rows)
```

Индексы

```
CREATE INDEX "ИД_ЛЮДИ_ИД" ON "Н_ЛЮДИ" USING hash("ИД");
CREATE INDEX "ИД_ЛЮДИ_ФАМИЛИЯ" ON "Н_ЛЮДИ" USING btree("ФАМИЛИЯ");
CREATE INDEX "ИД_ВЕДОМОСТИ_ИД" ON "Н_ВЕДОМОСТИ" USING btree("ИД");
```

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

Вывод

В ходе выполнения данной лабораторной работы я более подробно изучал язык `SQL` и познакомился с понятием "индекс"и написал свои индексы.

Также понял, что создание индексов не всегда ускоряет взаимодействие с базами данных, так как при инменение или удалении содержимого индексированного стольбца (при добавлении новой строки) индекс необходимо обновлять. Эти действия замедляют операции. Также лишнюю память хранением инлексов в тот момент, когда простой перебор сработает быстрее.

В контексте данной лабораторной работы я познакомился с такими типами индексов, как `btree` и `hash`.