Санкт-Петербургский Национальный Исследовательский Университет ИТМО

Факультет Программной Инженерии и Компьютерной Техники

Лабораторная работа №4
По дисциплине
Базы Данных

Выполнил студент группы P3117:

Кудрявцева Руслана Сергеевна

Преподаватель:

Чупанов Аликылыч Алибекович

Санкт-Петербург 2025 г.

Оглавление

Задание

По варианту, выданному преподавателем, составить и выполнить запросы к базе данных "Учебный процесс".

Отчёт по лабораторной работе должен содержать:

1. Текст задания.
2. Реализацию запросов на SQL.
3. Планы выполнения запросов.
4. Ответы на вопросы, представленные в задании.
5. Выводы по работе.

Введите вариант: 34213

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]

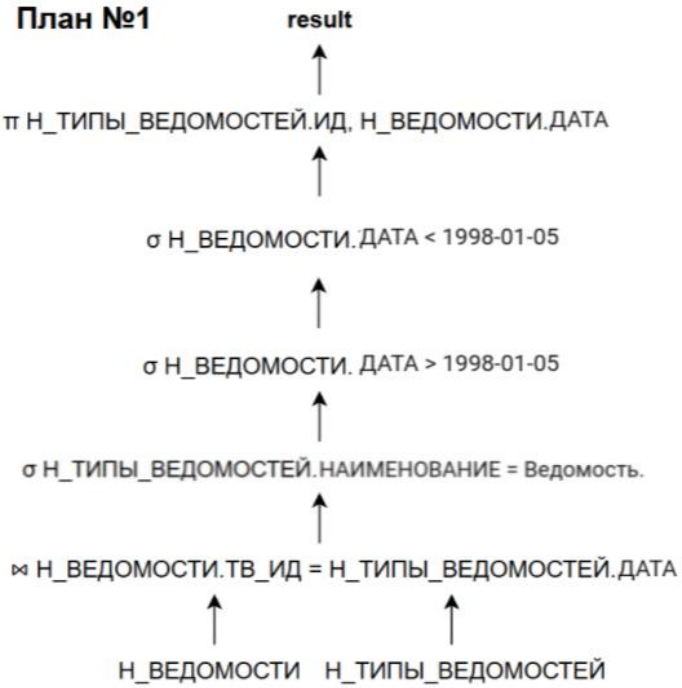
Подробные ответы на все вышеперечисленные вопросы должны присутствовать в отчете (планы выполнения запросов должны быть нарисованы, ответы на вопросы - представлены в текстовом виде).

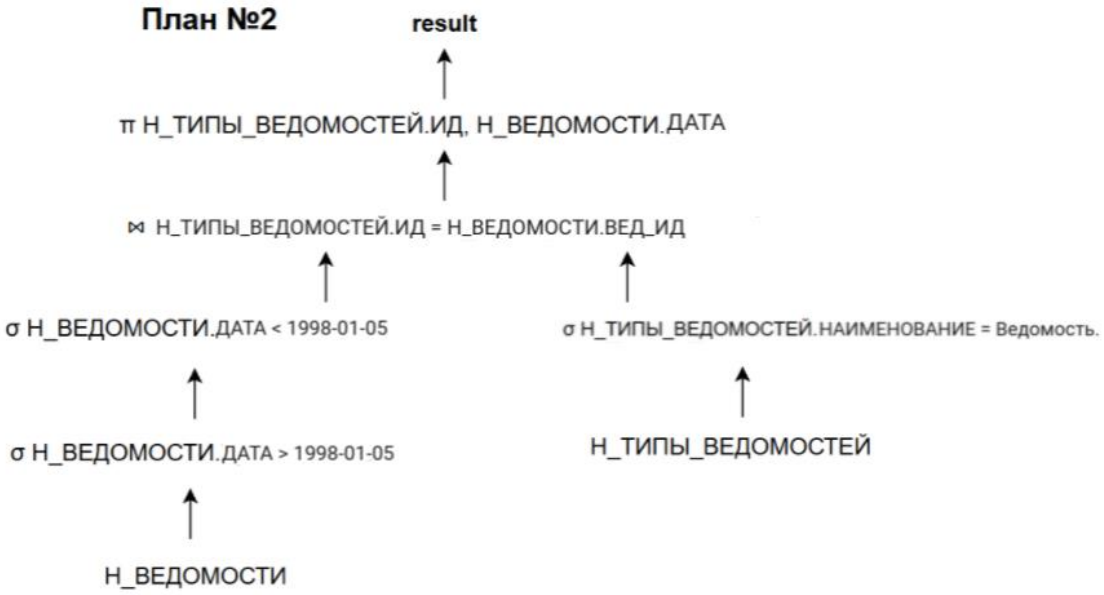
1. Сделать запрос для получения атрибутов из указанных таблиц, применив фильтры по указанным условиям:
   Н_ТИПЫ_ВЕДОМОСТЕЙ, Н_ВЕДОМОСТИ.
   Вывести атрибуты: Н_ТИПЫ_ВЕДОМОСТЕЙ.ИД, Н_ВЕДОМОСТИ.ДАТА.
   Фильтры (AND):
   a) Н_ТИПЫ_ВЕДОМОСТЕЙ.НАИМЕНОВАНИЕ = Ведомость.
   b) Н_ВЕДОМОСТИ.ДАТА < 1998-01-05.
   c) Н_ВЕДОМОСТИ.ДАТА > 1998-01-05.
   Вид соединения: INNER 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.ГРУППА.
   Фильтры: (AND)
   a) Н_ЛЮДИ.ИМЯ < Роман.
   b) Н_ОБУЧЕНИЯ.ЧЛВК_ИД < 113409.
   Вид соединения: RIGHT JOIN.

Запрос №1

Н_ТИПЫ_ВЕДОМОСТЕЙ, Н_ВЕДОМОСТИ.
Вывести атрибуты: Н_ТИПЫ_ВЕДОМОСТЕЙ.ИД, Н_ВЕДОМОСТИ.ДАТА.
Фильтры (AND):
a) Н_ТИПЫ_ВЕДОМОСТЕЙ.НАИМЕНОВАНИЕ = Ведомость.
b) Н_ВЕДОМОСТИ.ДАТА < 1998-01-05.
c) Н_ВЕДОМОСТИ.ДАТА > 1998-01-05.
Вид соединения: INNER JOIN.

```
SELECT "Н_ТИПЫ_ВЕДОМОСТЕЙ"."ИД", "Н_ВЕДОМОСТИ"."ДАТА"
FROM "Н_ТИПЫ_ВЕДОМОСТЕЙ"
INNER JOIN "Н_ВЕДОМОСТИ" ON "Н_ТИПЫ_ВЕДОМОСТЕЙ"."ИД" = "Н_ВЕДОМОСТИ"."ВЕД_ИД"
WHERE "Н_ТИПЫ_ВЕДОМОСТЕЙ"."НАИМЕНОВАНИЕ" = 'Ведомость'
AND "Н_ВЕДОМОСТИ"."ДАТА" > '1998-01-05'
AND "Н_ВЕДОМОСТИ"."ДАТА" < '1998-01-05';
```





Оптимальным является план №2, так как он производит объединение таблиц по ранее выбранным атрибутам, а не по таблицам целиком.

Индексы:

```
CREATE INDEX "ИНДЕКС_Н_ВЕДОМОСТЕЙ_ИД" ON "Н_ВЕДОМОСТЕЙ" USING btree("ВЕД_ИД");
CREATE INDEX "ИНДЕКС_Н_ВЕДОМОСТИ_ДАТА" ON "Н_ВЕДОМОСТИ" USING btree("ДАТА");
CREATE INDEX "ИНДЕКС_Н_ТИПЫ_ВЕДОМОСТЕЙ_НАИМ" ON "Н_ТИПЫ_ВЕДОМОСТЕЙ" USING
hash("НАИМЕНОВАНИЕ");
```

Добавление этих индексов должно ускорить выполнение запросов, так как по перечисленным полям происходит выборка с использованием оператора сравнения. Так же быстрее будет происходить соединение таблиц. Во втором случае используется операторы сравнения „>“ и „<“, так что эффективнее использовать btree. В первом и последнем случаях используется прямое сравнение, так что эффективнее использовать хэш-индекс.

Во всех случаеях ключ создается на поле, по которому идет фильтрация. В первом индексе идет соединение JOIN от Н_ТИПЫ_ВЕДОМОСТЕЙ к Н_ВЕДОМОСТИ, и именно там происходит фильтрация по поиску ВЕД_ИД. Так же и с остальными.

При добавлении индексов планы выполнения запросов изменятся, так как будет происходить индексный скан и Hash Join станет быстрее благодаря индексам.

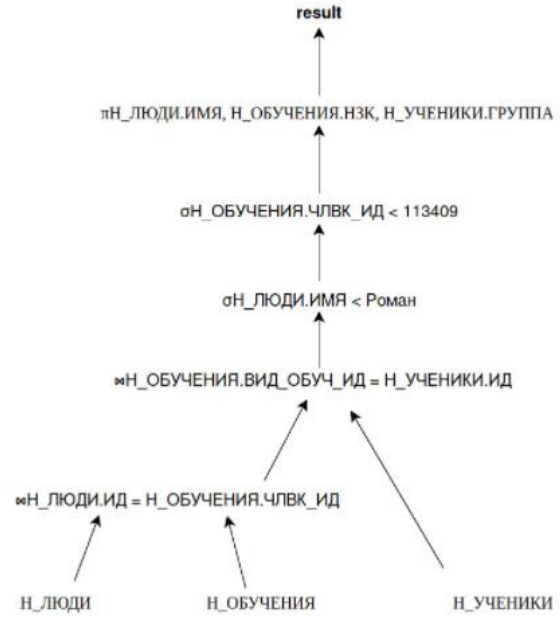
Результат EXPLAIN ANALYSE:

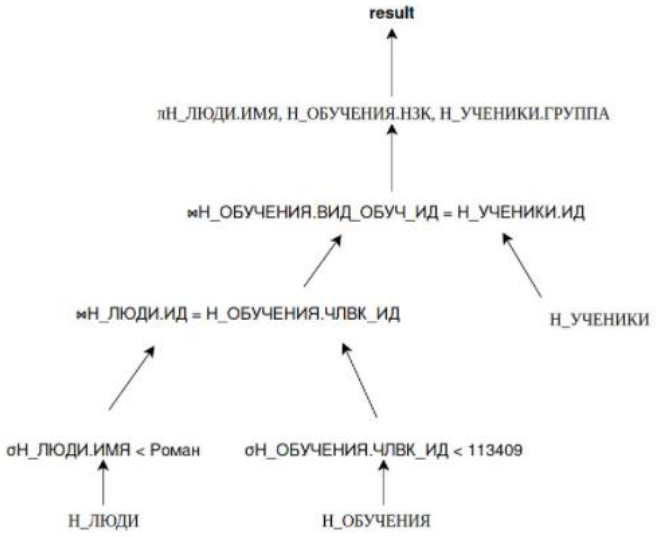
```
Nested Loop  (cost=15.69..2530.64 rows=1 width=12)
  Join Filter: ("Н_ТИПЫ_ВЕДОМОСТЕЙ"."ИД" = "Н_ВЕДОМОСТИ"."ВЕД_ИД")
  ->  Seq Scan on "Н_ТИПЫ_ВЕДОМОСТЕЙ"  (cost=0.00..1.04 rows=1 width=4)
        Filter: (("НАИМЕНОВАНИЕ")::text = 'Ведомость'::text)
  ->  Bitmap Heap Scan on "Н_ВЕДОМОСТИ"  (cost=15.69..2515.70 rows=1112 width=12)
        Recheck Cond: (("ДАТА" > '1998-01-05 00:00:00'::timestamp without time zone) AND ("ДАТА" <
'1998-01-05 00:00:00'::timestamp without time zone))
        ->  Bitmap Index Scan on "ВЕД_ДАТА_I"  (cost=0.00..15.42 rows=1112 width=0)
              Index Cond: (("ДАТА" > '1998-01-05 00:00:00'::timestamp without time zone) AND ("ДАТА" <
'1998-01-05 00:00:00'::timestamp without time zone))
```

Запрос №2

Таблицы: Н_ЛЮДИ, Н_ОБУЧЕНИЯ, Н_УЧЕНИКИ.
Вывести атрибуты: Н_ЛЮДИ.ИМЯ, Н_ОБУЧЕНИЯ.НЗК, Н_УЧЕНИКИ.ГРУППА.
Фильтры: (AND)
a) Н_ЛЮДИ.ИМЯ < Роман.
b) Н_ОБУЧЕНИЯ.ЧЛВК_ИД < 113409.
Вид соединения: RIGHT JOIN.

```
SELECT "Н_ЛЮДИ"."ИМЯ", "Н_ОБУЧЕНИЯ"."НЗК", "Н_УЧЕНИКИ"."ГРУППА"
FROM "Н_ЛЮДИ"
RIGHT JOIN "Н_ОБУЧЕНИЯ" ON "Н_ЛЮДИ"."ИД" = "Н_ОБУЧЕНИЯ"."ЧЛВК_ИД"
RIGHT JOIN "Н_УЧЕНИКИ" ON "Н_ОБУЧЕНИЯ"."ВИД_ОБУЧ_ИД" = "Н_УЧЕНИКИ"."ИД"
WHERE "Н_ЛЮДИ"."ИМЯ" < 'Роман'
AND "Н_ОБУЧЕНИЯ"."ЧЛВК_ИД" < 113409;
```





Оптимальным является план №2, так как он производит объединение таблиц по ранее выбранным атрибутам, а не по таблицам целиком.

Индексы:

```
CREATE INDEX "ИНДЕКС_Н_ЛЮДИ_ИМЯ" ON "Н_ЛЮДИ" USING btree("ИМЯ");
CREATE INDEX "ИНДЕКС_Н_ОБУЧЕНИЯ_ЧЛВК_ИД" ON "Н_ОБУЧЕНИЯ" USING btree("ЧЛВК_ИД");
CREATE INDEX "ИНДЕКС_Н_УЧЕНИКИ_ИД" ON "Н_УЧЕНИКИ" USING hash("ИД");
CREATE INDEX "ИНДЕКС_Н_ОБУЧЕНИЯ_ВИД_ОБУЧЕНИЯ_ИД" ON "Н_ОБУЧЕНИЯ" USING
btree("ВИД_ОБУЧ_ИД");
```

Добавление этих индексов должно ускорить выполнение запроса, так как по перечисленным полям происходит выборка с использованием операторов сравнения. Быстрее будет происходить соединение таблиц и фильтрация.

Для полей ИМЯ и ЧЛВК_ИД используются операторы „<“, поэтому здесь нужен именно btree, так как он поддерживает поиск по диапазонам. Для ВИД_ОБУЧ_ИД также подходит btree, потому что он используется при соединении, а btree универсален и эффективен при равенстве. Для поля ИД в таблице Н_УЧЕНИКИ, где используется точное сравнение,

можно использовать hash, так как в этом случае он будет быстрее. Во всех случаях индексы создаются на поля, по которым идёт фильтрация или соединение в запросе.

При добавлении индексов планы выполнения запросов изменятся, так как будет происходить индексный или хеш-скан. Использование Hash Join и Index Scan ускоряет выполнение за счёт сокращения объема перебираемых строк.

Результат EXPLAIN ANALYSE:

```
Nested Loop  (cost=79.33..288.26 rows=351 width=23)
 ->  Hash Join  (cost=79.04..267.91 rows=351 width=23)
     Hash Cond: ("Н_ЛЮДИ"."ИД" = "Н_ОБУЧЕНИЯ"."ЧЛВК_ИД")
     ->  Seq Scan on "Н_ЛЮДИ"  (cost=0.00..163.97 rows=4277 width=17)
         Filter: (("ИМЯ")::text < 'Роман'::text)
     ->  Hash  (cost=73.79..73.79 rows=420 width=14)
         ->  Bitmap Heap Scan on "Н_ОБУЧЕНИЯ"  (cost=11.54..73.79 rows=420 width=14)
             Recheck Cond: ("ЧЛВК_ИД" < 113409)
             ->  Bitmap Index Scan on "ОБУЧ_ЧЛВК_FK_I"  (cost=0.00..11.43 rows=420 width=0)
                 Index Cond: ("ЧЛВК_ИД" < 113409)
 ->  Memoize  (cost=0.30..3.88 rows=1 width=8)
     Cache Key: "Н_ОБУЧЕНИЯ"."ВИД_ОБУЧ_ИД"
     Cache Mode: logical
     ->  Index Scan using "УЧЕН_PK" on "Н_УЧЕНИКИ"  (cost=0.29..3.87 rows=1 width=8)
         Index Cond: ("ИД" = "Н_ОБУЧЕНИЯ"."ВИД_ОБУЧ_ИД")
```

Вывод: Во время выполнения данной лабораторной работы я научилась оптимизировать запросы, составлять наиболее выгодный план выполнения запросов, используя для этого подходящие виды индексов.