Федеральное государственное автономное
образовательное учреждение высшего образования
«Научно-образовательная корпорация ИТМО»

Факультет программной инженерии и компьютерной техники
Направление подготовки 09.03.04 Программная инженерия

Отчёт по лабораторной работе №4

По дисциплине «Базы данных» (второй семестр)

Студент:
Дениченко Александр P3112
Практик:
Лисицина В.В

Санкт-Петербург
2023 г.

1 Задание

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: H_ЛЮДИ, H_СЕССИЯ.
Вывести атрибуты: H_ЛЮДИ.ФАМИЛИЯ, H_СЕССИЯ.ДАТА.
Фильтры (AND):
a) H_ЛЮДИ.ФАМИЛИЯ < Ёлкин.
b) H_СЕССИЯ.УЧГОД > 2001/2002.
Вид соединения: LEFT JOIN.

2. Сделать запрос для получения атрибутов из указанных таблиц, применив фильтры по указанным условиям:
Таблицы: H_ЛЮДИ, H_ОБУЧЕНИЯ, H_УЧЕНИКИ.
Вывести атрибуты: H_ЛЮДИ.ФАМИЛИЯ, H_ОБУЧЕНИЯ.НЗК, H_УЧЕНИКИ.ИД.
Фильтры: (AND)
a) H_ЛЮДИ.ОТЧЕСТВО = Сергеевич.
b) H_ОБУЧЕНИЯ.ЧЛВК_ИД < 105590.
c) H_УЧЕНИКИ.НАЧАЛО > 2008-09-01.
Вид соединения: LEFT JOIN.

2 Запросы и индексы

1)Первый запрос:

```
select H_ЛЮДИ.ФАМИЛИЯ, H_СЕССИЯ.ДАТА from H_ЛЮДИ
left outer join H_СЕССИЯ on H_ЛЮДИ.ИД = H_СЕССИЯ.ЧЛВК_ИД
where (H_ЛЮДИ.ФАМИЛИЯ < 'Ёлкин' and H_СЕССИЯ.УЧГОД > '2001/2002');
```

Planning Time: 0.461 ms
Execution Time: 3.804 ms

Для ускорения запроса можно предложить следующий вариант:

1.1 Для таблицы H_ЛЮДИ создать индекс, индексирующий столбец ФАМИЛИЯ, так как в этом отношении ФАМИЛИЯ не отсортирована, а также будем использовать кластеризованную индексацию, так как нам нужно, чтобы индекс хранил в листьях строки данных:

```
CREATE CLUSTERED INDEX idx_фамилия ON H_ЛЮДИ USING btree(ФАМИЛИЯ);
```

1.2 Так как кластеризованной индексации нет в найшей базе данных, будем использовать только в-дерево:

```
CREATE INDEX idx_фамилия ON H_ЛЮДИ USING btree(ФАМИЛИЯ);
```

1.3 Для таблицы Н_СЕССИЯ никаких индексов создавать не нужно, так как количество объектов в этом отношении не слишком большое – индекс лишь замедлит процесс выборки.

2)Второй запрос:

select Н_ЛЮДИ.ФАМИЛИЯ, Н_ОБУЧЕНИЯ.НЗК, Н_УЧЕНИКИ.ИД from Н_ЛЮДИ
left outer join Н_ОБУЧЕНИЯ on Н_ЛЮДИ.ИД = Н_ОБУЧЕНИЯ.ЧЛВК_ИД
left outer join Н_УЧЕНИКИ on Н_ЛЮДИ.ИД = Н_УЧЕНИКИ.ЧЛВК_ИД
where (Н_ЛЮДИ.ОТЧЕСТВО like 'Сергеевич' and Н_ОБУЧЕНИЯ.ЧЛВК_ИД < 105590 and Н_УЧЕНИКИ.НАЧАЛО >'2008-09-01');

Planning Time: 0.766 ms
Execution Time: 0.054 ms

Для ускорения запроса можно предложить следующий вариант:

2.1 Для таблицы Н_ЛЮДИ можно проиндексировать атрибут ОТЧЕСТВО:

CREATE INDEX idx_отчество ON Н_ЛЮДИ USING btree(ОТЧЕСТВО);

2.2 Для таблицы Н_ОБУЧЕНИЯ можно проиндексировать атрибут ЧЛВК_ИД, так как в условии WHERE используется неравенство, то выборку мы можем ускорить при помощи B-дерева:

CREATE INDEX idx_ид_обуч ON Н_ОБУЧЕНИЯ USING btree(ЧЛВК_ИД);

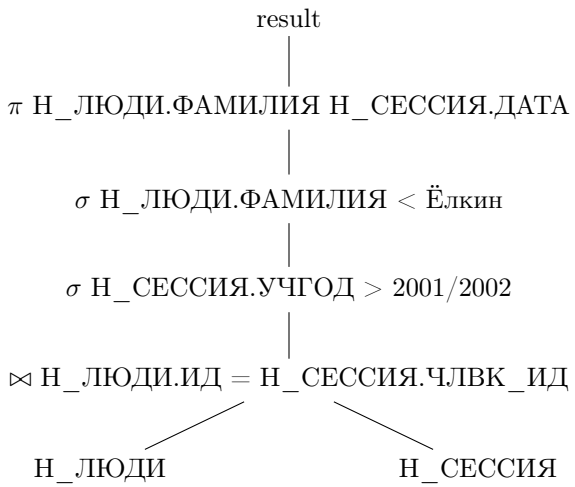
2.3 Для таблицы Н_УЧЕНИКИ можно проиндексировать атрибут НАЧАЛО, так как в условии WHERE используется неравенство, то выборку мы можем ускорить при помощи B-дерева:

CREATE INDEX idx_начало ON Н_УЧЕНИКИ USING btree(НАЧАЛО);

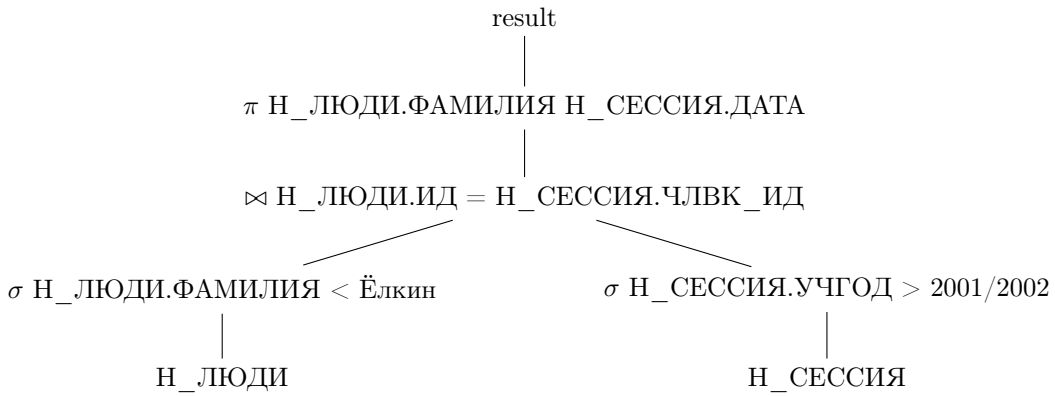
3 Планы выполнения

1)Первый запрос:

3.1 Неоптимальный план выполнения:



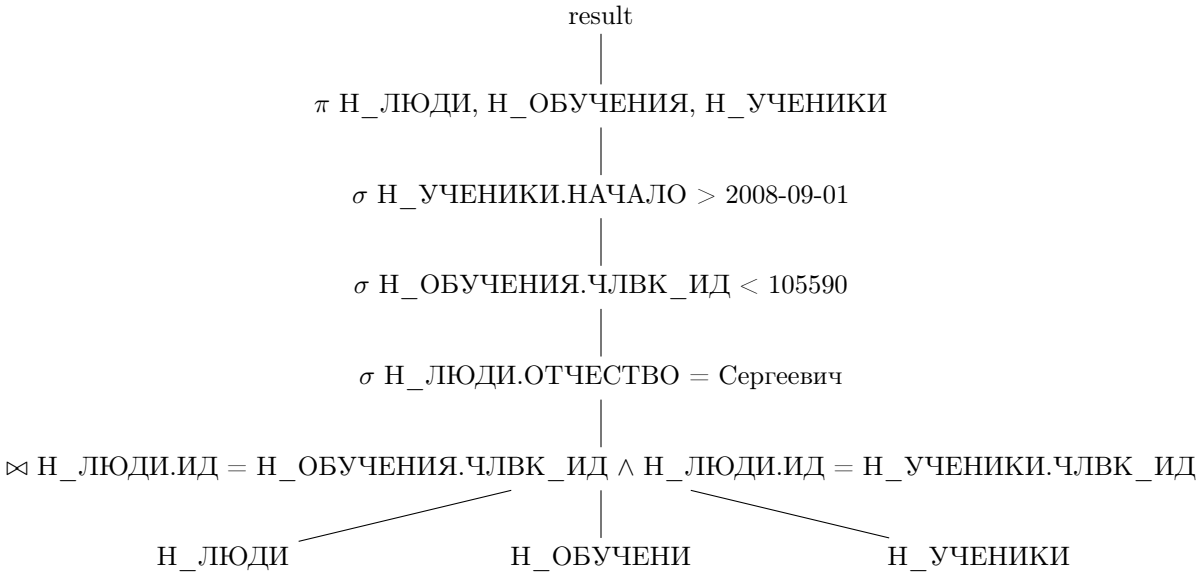
3.2 Оптимальный план выполнения:
Целью оптимизации является уменьшение размеров промежуточных данных -> уменьшение числа операций чтения записи во внешнюю память



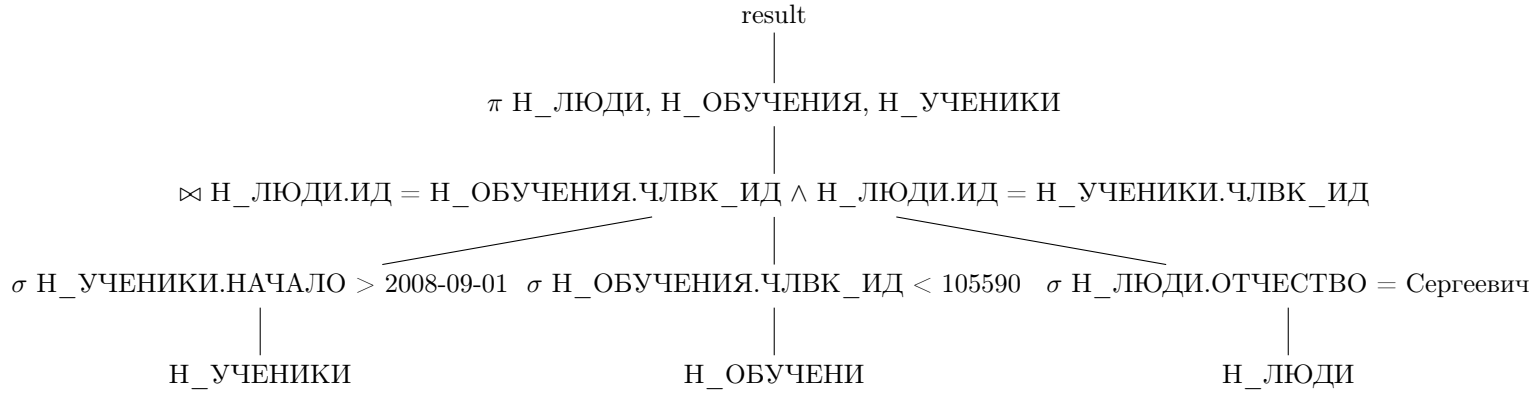
Этот план выполнения оптимален так как мы достигли цели. На момент соединения строк перебор будет минимальным за счёт отдельных процессов выборки, получилось разделение процессов.

2)Второй запрос:

4.1 Неоптимальный план выполнения:

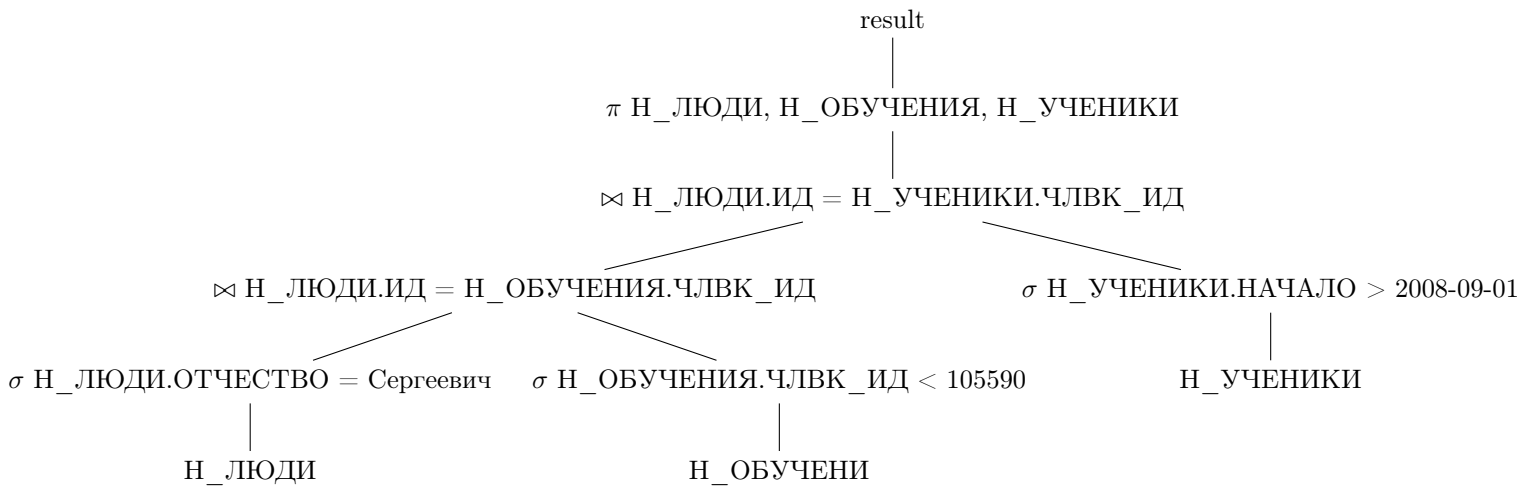


4.2 Средне - оптимальный план выполнения:
Целью оптимизации является уменьшение размеров промежуточных данных -> уменьшение числа операций чтения записи во внешнюю память



Этот план выполнения средне - оптимальный так как мы достигли цели, но для подобных планов лучше использовать левостороннее дерево, где выборка будет происходить перед каждым соединением, что упростит поиск подходящих объектов.

4.2 Оптимальный план выполнения:



5.1 Влияние на план выборки индексами:

5.2 Если запрос содержит условие, которое может быть поддержано созданным индексом, это может привести к более быстрому выполнению операции поиска. База данных может использовать индекс для быстрого нахождения соответствующих строк, что улучшит производительность запроса.

5.3 Вместо сортировки всей таблицы, база данных может использовать индекс для получения данных уже отсортированными, что может значительно ускорить выполнение запроса.

5.4 При выполнении операции объединения таблиц, наличие соответствующих индексов может улучшить производительность. База данных может использовать индексы для быстрого нахождения соответствующих строк и выполнения объединения.

5.5 Добавление индекса может изменить стоимость выполнения различных операций в базе данных. Планировщик запросов может выбрать новый план выполнения, учитывая наличие индексов и их стоимость. В некоторых случаях это может привести к изменению порядка выполнения операций или выбору других типов доступа к данным.

4 EXPLAIN ANALYZE

Запрос первый:

EXPLAIN ANALYZE select Н_ЛЮДИ.ФАМИЛИЯ, Н_СЕССИЯ.ДАТА from Н_ЛЮДИ
left outer join Н_СЕССИЯ on Н_ЛЮДИ.ИД = Н_СЕССИЯ.ЧЛВК_ИД
where (Н_ЛЮДИ.ФАМИЛИЯ < 'Ёлкин' and Н_СЕССИЯ.УЧГОД > '2001/2002');

Answer:

Nested Loop (cost=0.29..286.49 rows=826 width=24) (actual time=0.067..3.782 rows=937 loops=1)
-> Seq Scan on "Н_СЕССИЯ"(cost=0.00..117.90 rows=3630 width=12) (actual time=0.014..1.887 rows=3630 loops=1)
Filter: (("УЧГОД")::text > '2001/2002'::text)
Rows Removed by Filter: 122
-> Memoize (cost=0.29..0.44 rows=1 width=20) (actual time=0.000..0.000 rows=0 loops=3630)
Cache Key: "Н_СЕССИЯ"."ЧЛВК_ИД"

Cache Mode: logical
Hits: 3452 Misses: 178 Evictions: 0 Overflows: 0 Memory Usage: 15kB
-> Index Scan using "ЧЛВК_РК"on "Н_ЛЮДИ"(cost=0.28..0.43 rows=1 width=20) (actual time=0.003..0.003 rows=0 loops=178)
Index Cond: ("ИД- "Н_СЕССИЯ"."ЧЛВК_ИД")
Filter: (("ФАМИЛИЯ")::text < 'Ёлкин'::text)
Rows Removed by Filter: 1
Planning Time: 0.398 ms
Execution Time: 3.870 ms (14 строк)

Запрос второй:

explain analyze select Н_ЛЮДИ.ФАМИЛИЯ, Н_ОБУЧЕНИЯ.НЗК, Н_УЧЕНИКИ.ИД from Н_ЛЮДИ
left outer join Н_ОБУЧЕНИЯ on Н_ЛЮДИ.ИД = Н_ОБУЧЕНИЯ.ЧЛВК_ИД
left outer join Н_УЧЕНИКИ on Н_ЛЮДИ.ИД = Н_УЧЕНИКИ.ЧЛВК_ИД
where (Н_ЛЮДИ.ОТЧЕСТВО like 'Сергеевич' and Н_ОБУЧЕНИЯ.ЧЛВК_ИД < 105590 and Н_УЧЕНИКИ.НАЧАЛО
>'2008-09-01');

Answer:

Nested Loop (cost=0.85..21.42 rows=1 width=26) (actual time=0.006..0.006 rows=0 loops=1)
-> Nested Loop (cost=0.56..15.22 rows=1 width=30) (actual time=0.005..0.006 rows=0 loops=1)
-> Index Scan using "ОБУЧ_ЧЛВК_FK_I"on "Н_ОБУЧЕНИЯ"(cost=0.28..6.87 rows=1 width=10) (actual time=0.005..0.005
rows=0 loops=1)
Index Cond: ("ЧЛВК_ИД« 105590)
-> Index Scan using "ЧЛВК_РК"on "Н_ЛЮДИ"(cost=0.28..8.30 rows=1 width=20) (never executed)
Index Cond: ("ИД- "Н_ОБУЧЕНИЯ"."ЧЛВК_ИД")
Filter: (("ОТЧЕСТВО")::text 'Сергеевич'::text)
-> Index Scan using "УЧЕН_ОБУЧ_FK_I"on "Н_УЧЕНИКИ"(cost=0.29..6.20 rows=1 width=8) (never executed)
Index Cond: ("ЧЛВК_ИД- "Н_ЛЮДИ"."ИД")
Filter: ("НАЧАЛО» '2008-09-01 00:00:00'::timestamp without time zone)
Planning Time: 0.687 ms
Execution Time: 0.055 ms
(12 строк)

5 Вывод

1. Индексы играют важную роль в оптимизации запросов, ускоряя операции поиска, сортировки и объединения таблиц.
2. При добавлении индексов могут измениться планы выполнения запросов, что может улучшить производительность.
3. Выбор типа индекса зависит от структуры таблицы и операций над данными.
4. Необходимо проводить анализ производительности и тестирование изменений с индексами.
5. Оптимизация производительности базы данных - сложная задача, требующая учета множества факторов.