Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное учреждение высшего образования

«Национальный исследовательский университет ИТМО»

Факультет Программной инженерии и компьютерной техники

Лабораторная работа №4

по «Информационным системам и базам данных»

Выполнил: Группа P33312 Хайкин О. И.

Преподаватель:

`Наумова Н. А.

Санкт-Петербург, 2023

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: H_ОЦЕНКИ, H_ВЕДОМОСТИ.
   Вывести атрибуты: H_ОЦЕНКИ.КОД, H_ВЕДОМОСТИ.ДАТА.
   Фильтры (AND):
   a) H_ОЦЕНКИ.КОД = 5.
   b) H_ВЕДОМОСТИ.ИД > 1457443.
   c) H_ВЕДОМОСТИ.ИД > 1457443.
   Вид соединения: INNER 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.ИД, H_ОБУЧЕНИЯ.НЗК, H_УЧЕНИКИ.НАЧАЛО.
   Фильтры: (AND)
   a) H_ЛЮДИ.ИМЯ = Владимир.
   b) H_ОБУЧЕНИЯ.НЗК = 001000.
   Вид соединения: INNER JOIN.

Первый запрос

Реализация запроса

```
SELECT оценки.КОД, ведомости.ДАТА
FROM Н_ОЦЕНКИ оценки
INNER JOIN Н_ВЕДОМОСТИ ведомости
ON оценки.КОД = ведомости.ОЦЕНКА
WHERE
    оценки.КОД = '5'
    AND ведомости.ИД > 1457443
    AND ведомости.ИД > 1457443;
```

Полезные для запроса индексы

Hash-индекс для столбца “КОД” в таблице Н_ОЦЕНКИ

Уменьшит время фильтрации по условию оценки.КОД = ‘5’

Hash-индекс для столбца “ОЦЕНКА” в таблице Н_ВЕДОМОСТИ

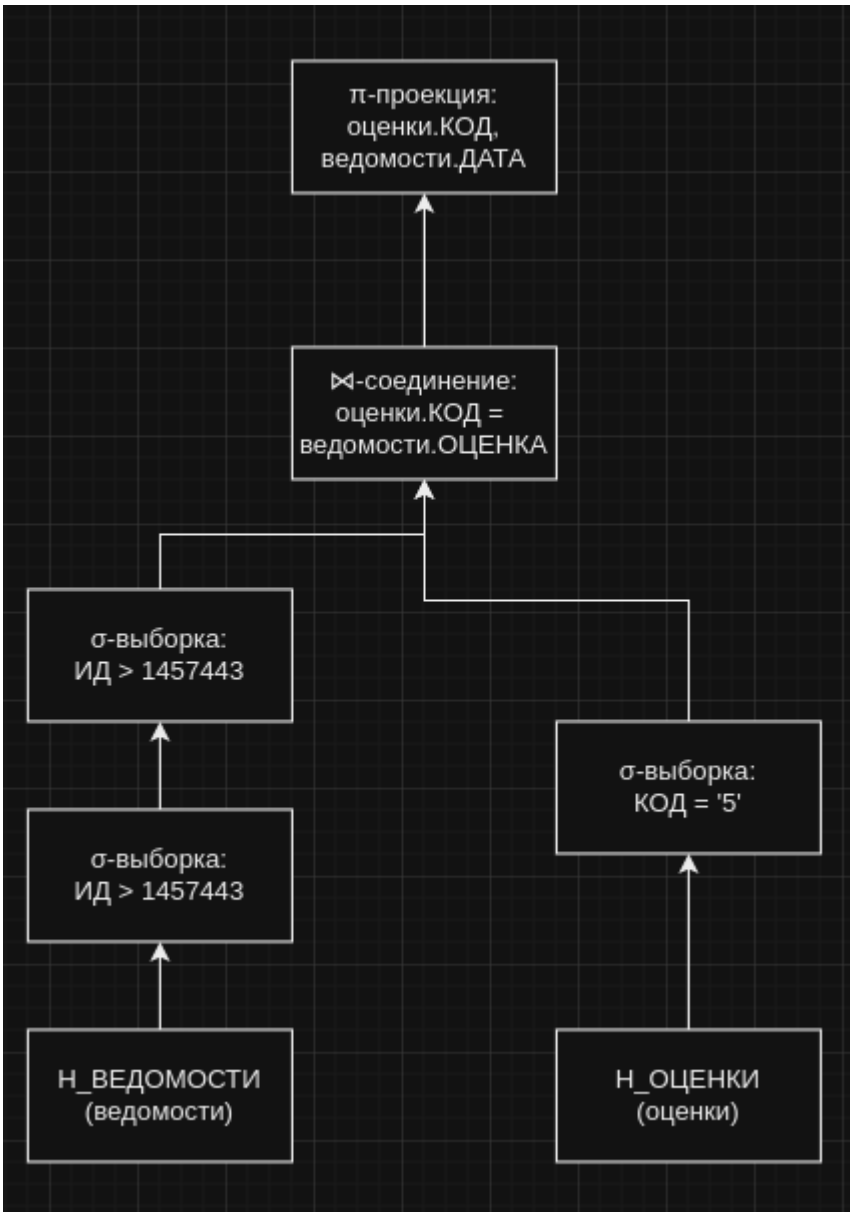
Уменьшит время выполнения Join’а

BTree-индекс для столбца “ИД” в таблице Н_ВЕДОМОСТИ

Уменьшит время фильтрации по условиям ведомости.ИД > 1457443

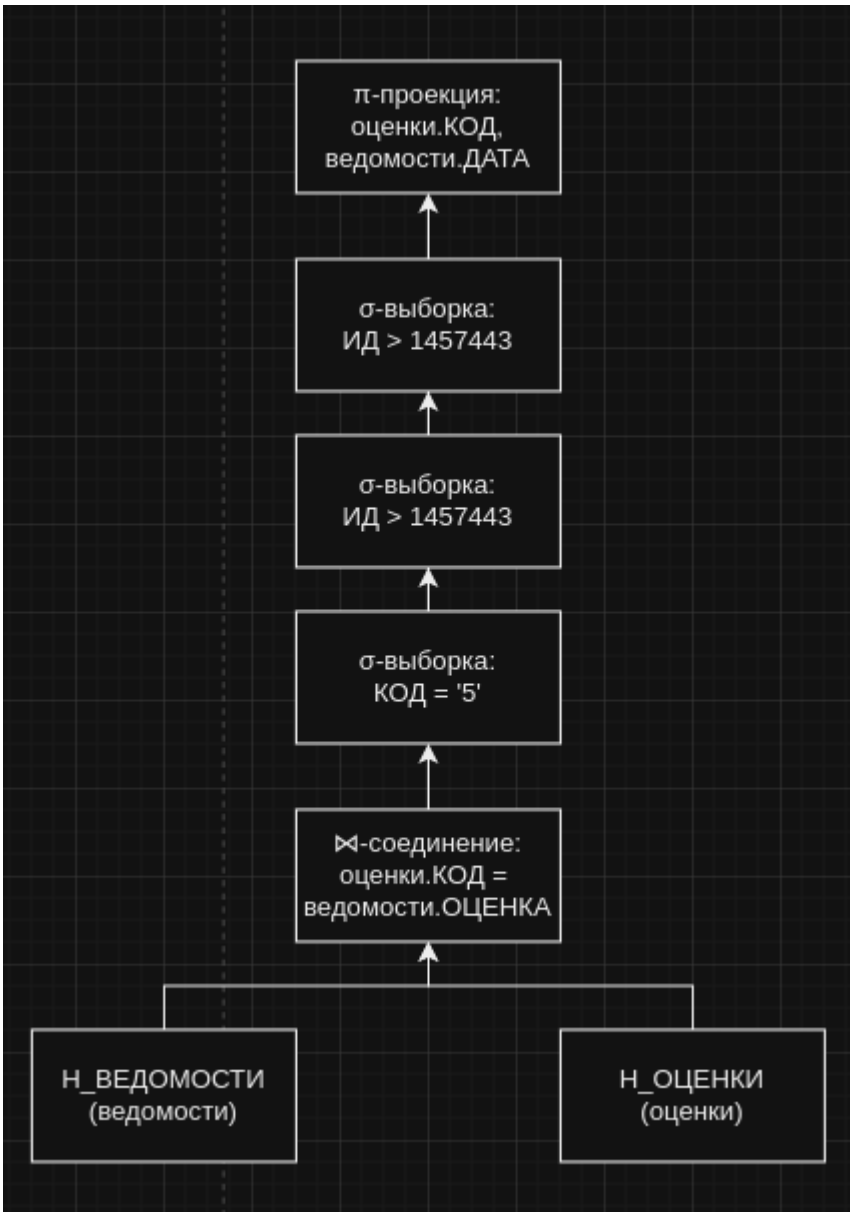
Возможные планы выполнения запроса

1



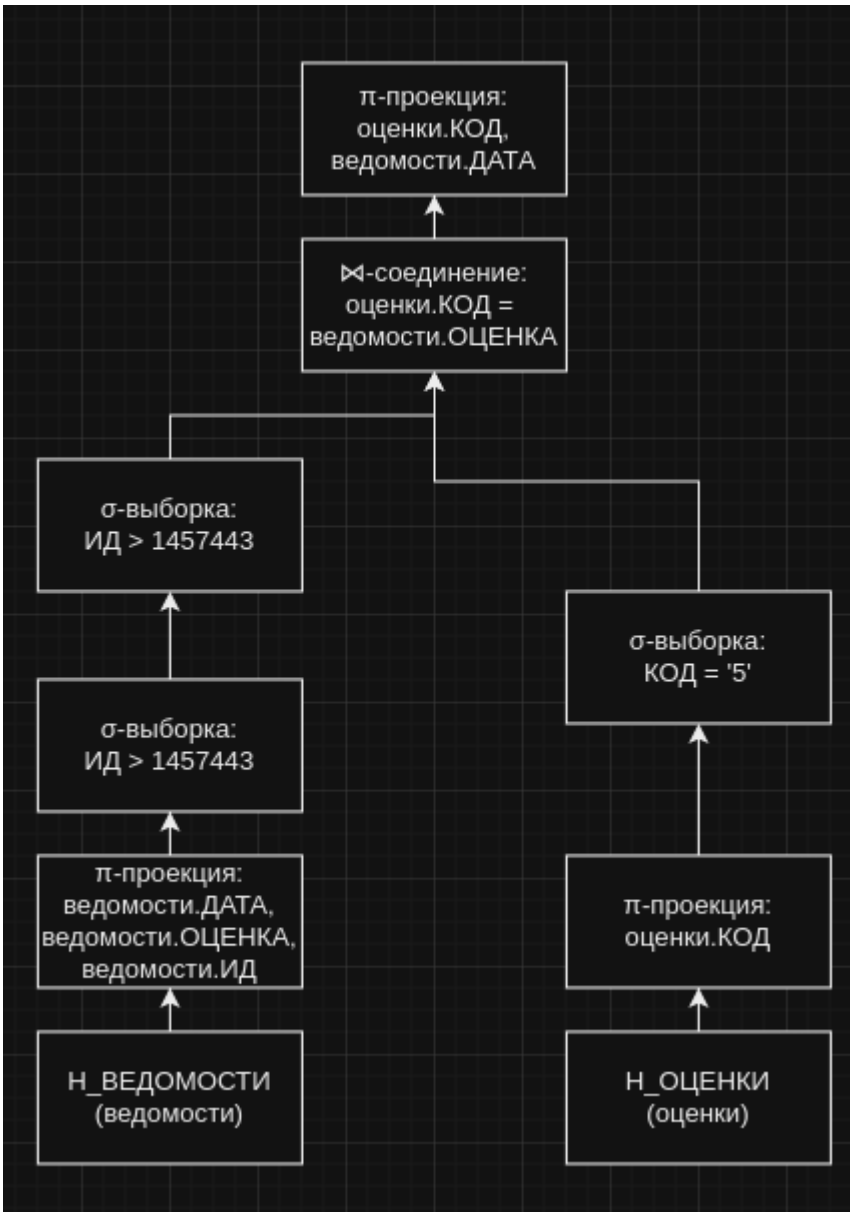
В этом варианте плана сначала выполняется фильтрация, а затем соединение и финальная проекция для получения результата.

2



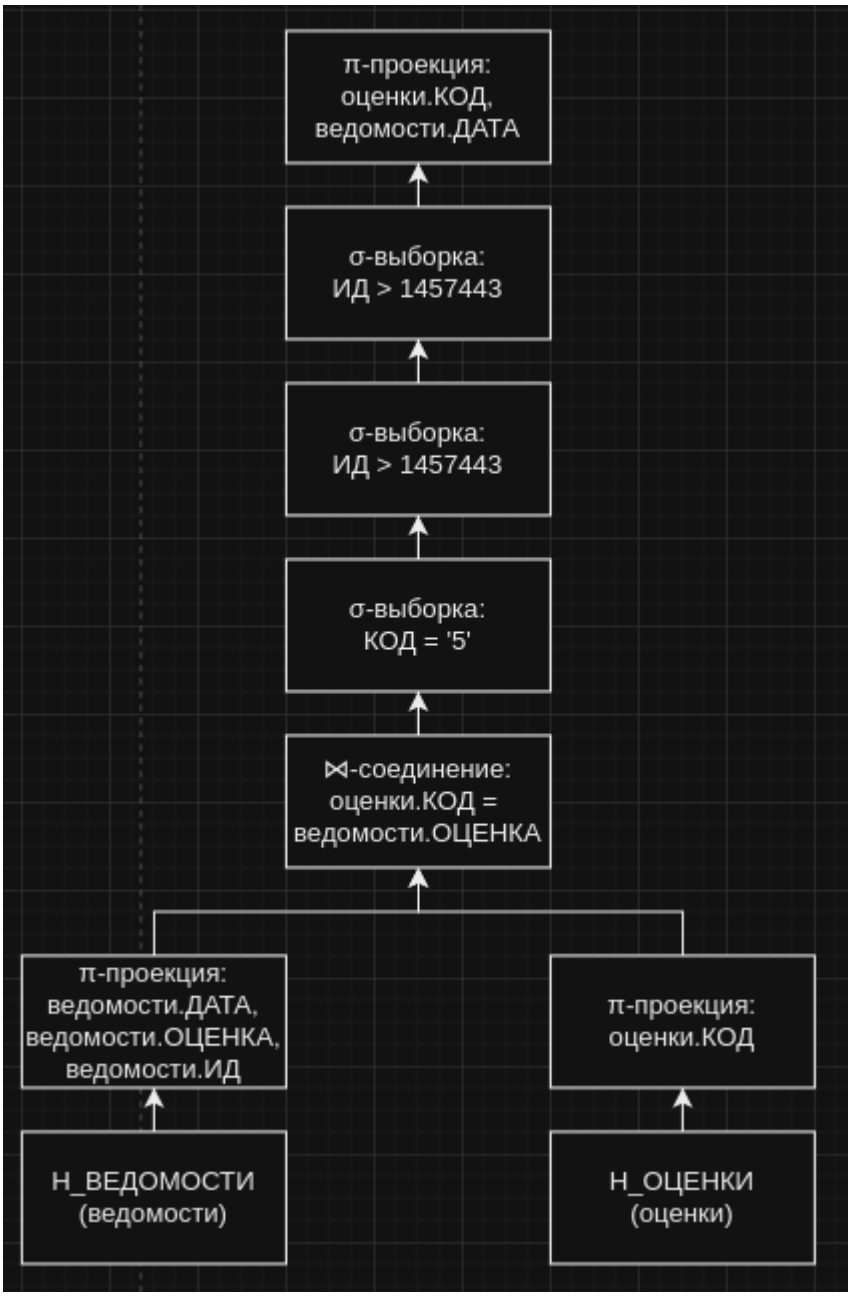
В этом варианте плана сначала выполняется соединение, а затем фильтрация и финальная проекция для получения результата.

3



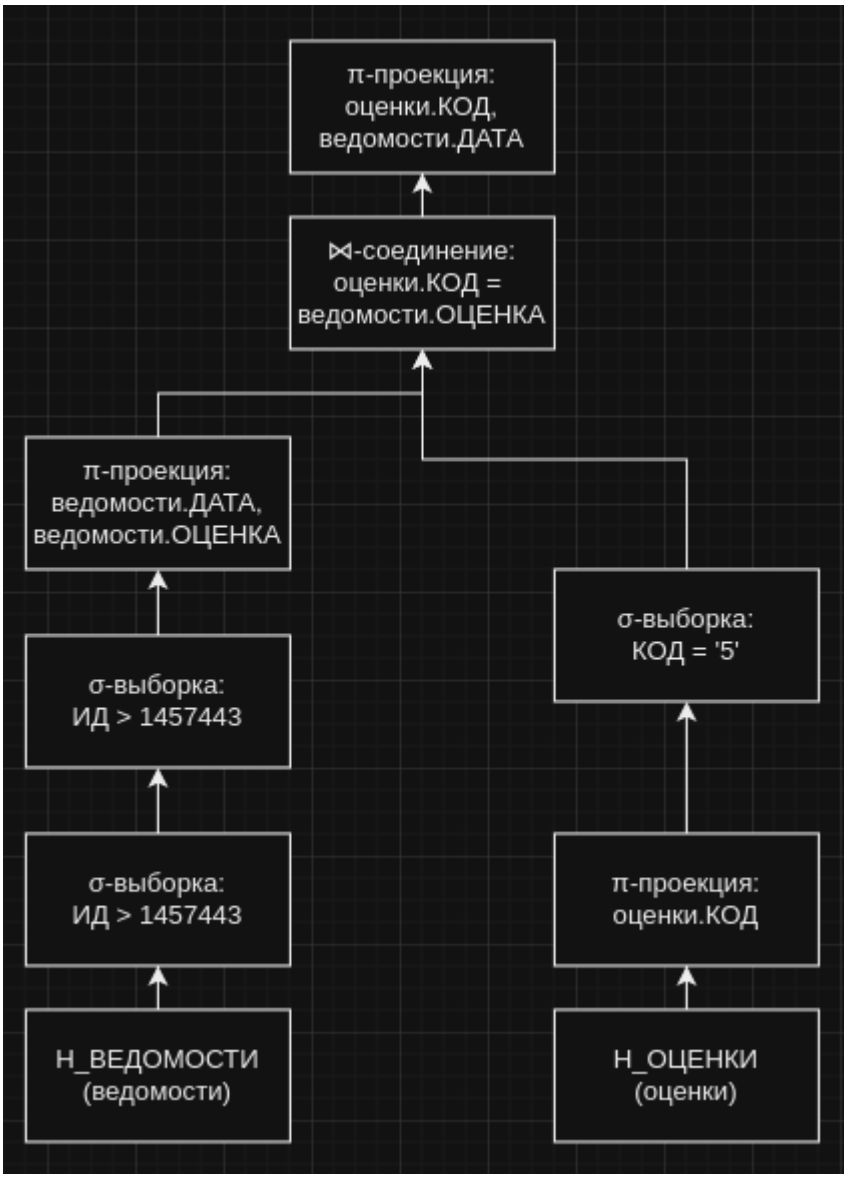
В этом варианте плана сначала выполняются проекции, избавляющиеся от “лишних” атрибутов, потом фильтрация, а затем соединение и финальная проекция для получения результата.

4



В этом варианте плана сначала выполняются проекции, избавляющиеся от “лишних” атрибутов, потом соединение, а затем фильтрация и финальная проекция для получения результата.

5



Оптимальный план и влияние индексов

Из всех предложенных планов я бы посчитал оптимальным план №3 - план, в котором сначала выполняются проекции, после чего фильтрация, а в конце соединение. При таком плане фильтрация происходит перед соединением, что позволяет избежать “лишних” соединений, от которых потом избавятся. Кроме этого, предварительное использование проекций позволяет не “нести” за собой ненужные данные при выполнении запроса.

При использовании предложенных ранее индексов этот план не изменится, и лишь сократит время выполнения операций фильтрации и соединения.

Вывод EXPLAIN ANALYZE

```
                                                     QUERY PLAN
------------------------------------------------------------------------------------------------------------------------------------------
 Nested Loop  (cost=485.52..1989.95 rows=539 width=13) (actual time=1.814..2.080 rows=461 loops=1)
   ->  Seq Scan on "Н_ОЦЕНКИ" "оценки"  (cost=0.00..1.11 rows=1 width=5) (actual time=0.011..0.013 rows=1
loops=1)
         Filter: (("КОД")::text = '5'::text)
         Rows Removed by Filter: 8
   ->  Bitmap Heap Scan on "Н_ВЕДОМОСТИ" "ведомости"  (cost=485.52..1983.44 rows=539 width=14) (actual
time=1.796..1.983 rows=461 loops=1)
         Recheck Cond: (("ИД" > 1457443) AND (("ОЦЕНКА")::text = '5'::text))
         Heap Blocks: exact=93
         ->  BitmapAnd  (cost=485.52..485.52 rows=539 width=0) (actual time=1.781..1.781 rows=0 loops=1)
               ->  Bitmap Index Scan on "ВЕД_РК"  (cost=0.00..68.02 rows=3160 width=0) (actual time=0.155..0.156
rows=3389 loops=1)
                     Index Cond: (("ИД" > 1457443) AND ("ИД" > 1457443))
               ->  Bitmap Index Scan on "ВЕД_ОЦЕНКА_I"  (cost=0.00..416.98 rows=37941 width=0) (actual
time=1.615..1.615 rows=37825 loops=1)
                     Index Cond: (("ОЦЕНКА")::text = '5'::text)
 Planning Time: 0.250 ms
 Execution Time: 2.141 ms
```

Второй запрос

Реализация запроса

```sql
SELECT люди.ИД, обучения.НЗК, ученики.НАЧАЛО
FROM Н_ЛЮДИ люди
INNER JOIN Н_УЧЕНИКИ ученики ON люди.ИД = ученики.ЧЛВК_ИД
INNER JOIN Н_ОБУЧЕНИЯ обучения ON люди.ИД = обучения.ЧЛВК_ИД
WHERE
люди.ИМЯ = 'Владимир'
AND обучения.НЗК = '001000';
```

Полезные для запроса индексы

Hash-индекс для столбца ИМЯ в таблице Н_ЛЮДИ

Уменьшит время фильтрации по условию люди.ИМЯ = ‘ВЛАДИМИР’

Hash-индекс для столбца НЗК в таблице Н_ОБУЧЕНИЯ

Уменьшит время фильтрации по условию обучения.НЗК = ‘ВЛАДИМИР’

Hash-индекс для столбца ЧЛВК_ИД в таблице Н_УЧЕНИКИ

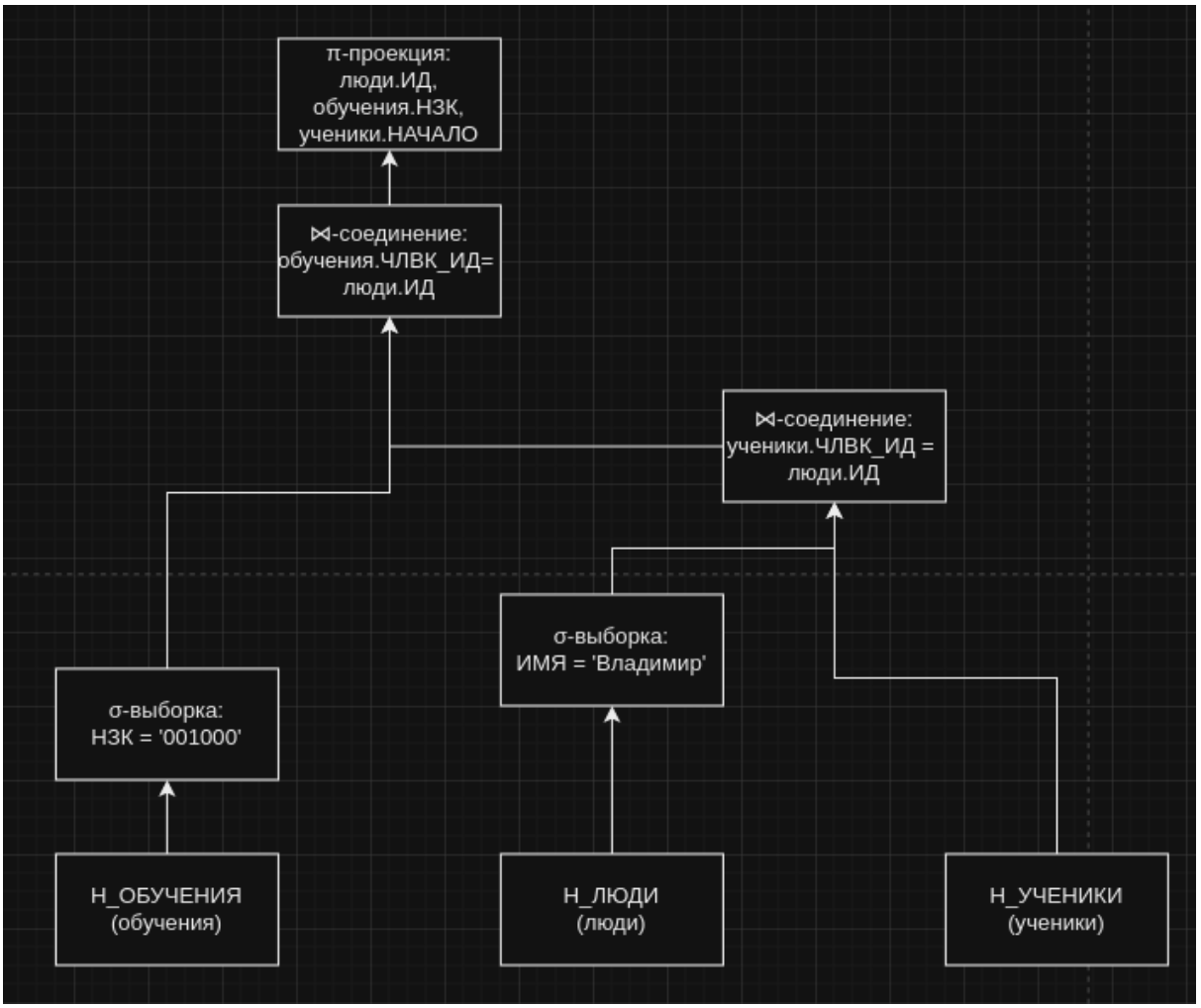
Уменьшит время выполнения Join’а

Hash-индекс для столбца ЧЛВК_ИД в таблице Н_ОБУЧЕНИЯ

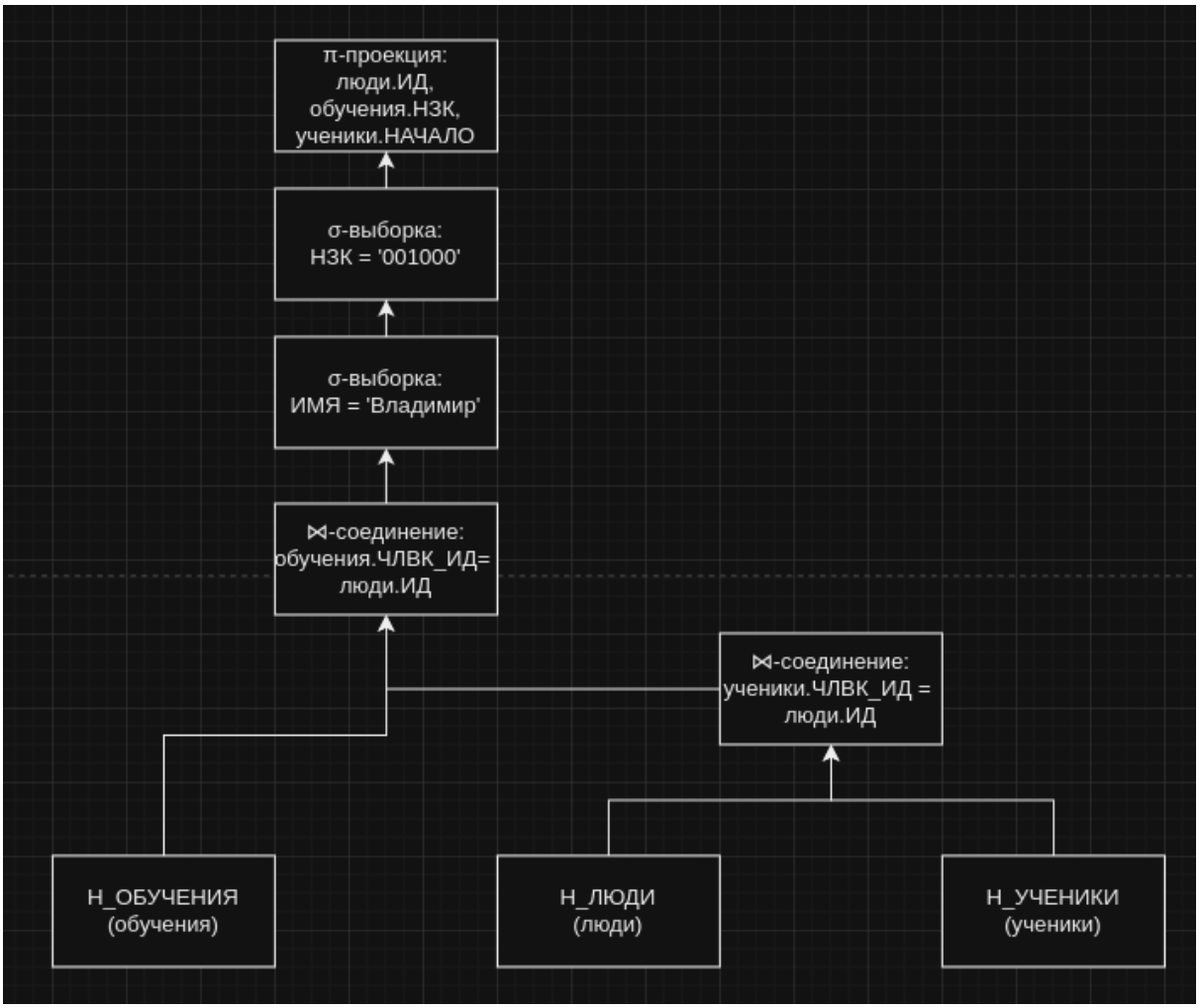
Уменьшит время выполнения Join’а

Возможные планы выполнения запроса

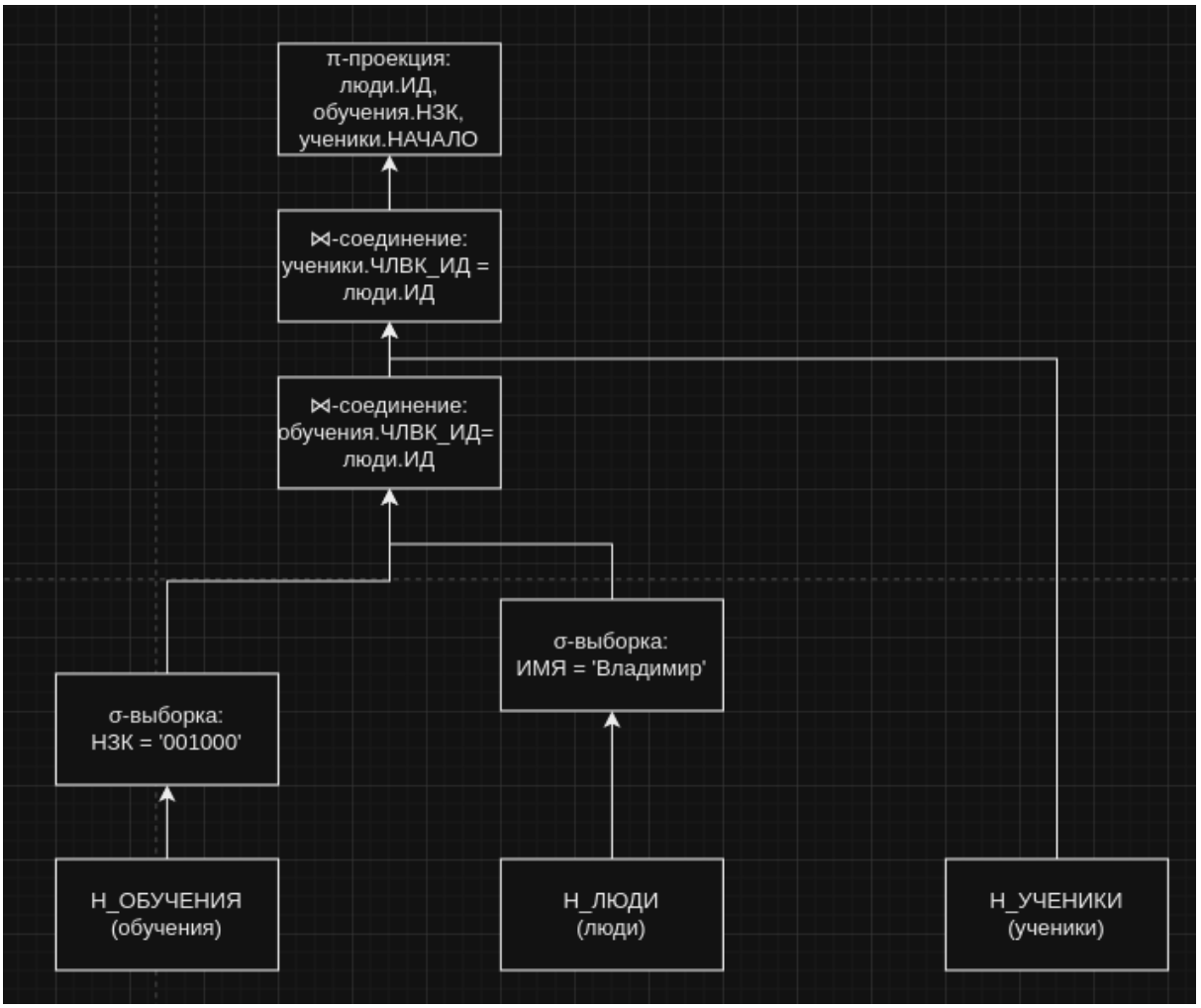
1



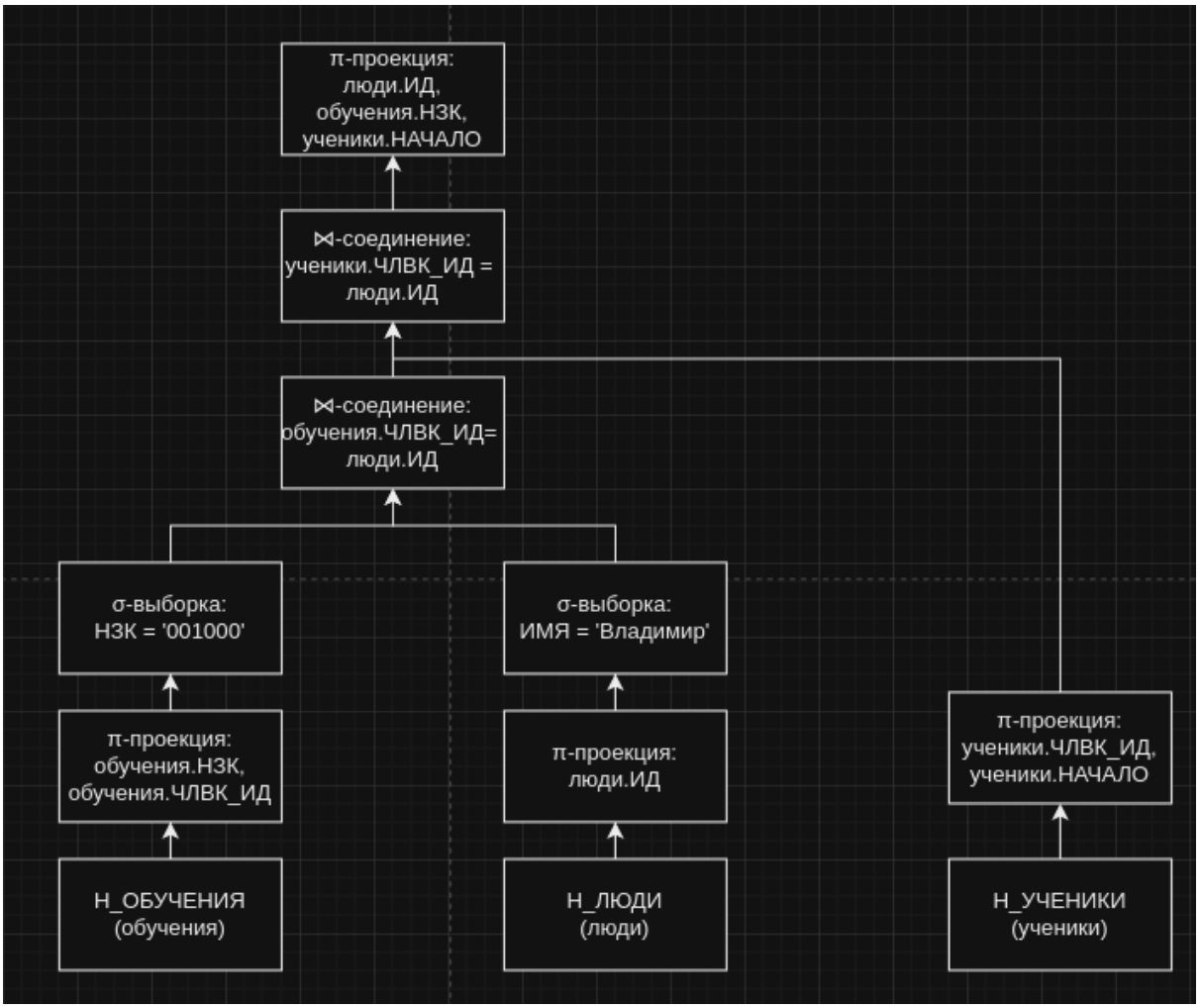
2


π-проекция:
люди.ИД,
обучения.НЗК,
ученики.НАЧАЛО
σ-выборка:
НЗК = '001000'
σ-выборка:
ИМЯ = 'Владимир'
⋈-соединение:
обучения.ЧЛВК_ИД=
люди.ИД
⋈-соединение:
ученики.ЧЛВК_ИД =
люди.ИД
Н_ОБУЧЕНИЯ
(обучения)
Н_ЛЮДИ
(люди)
Н_УЧЕНИКИ
(ученики)

3


π-проекция:
люди.ИД,
обучения.НЗК,
ученики.НАЧАЛО
⋈-соединение:
ученики.ЧЛВК_ИД =
люди.ИД
⋈-соединение:
обучения.ЧЛВК_ИД=
люди.ИД
σ-выборка:
ИМЯ = 'Владимир'
σ-выборка:
НЗК = '001000'
Н_ОБУЧЕНИЯ
(обучения)
Н_ЛЮДИ
(люди)
Н_УЧЕНИКИ
(ученики)

4



Оптимальный план и влияние индексов

Из всех предложенных планов я бы посчитал оптимальным план №4 - план, в котором сначала выполняются проекции, после чего фильтрация, а в конце соединение. При таком плане фильтрация происходит перед соединением, что позволяет избежать “лишних” соединений, от которых потом избавятся. Кроме этого, предварительное использование проекций позволяет не “нести” за собой ненужные данные при выполнении запроса.

При использовании предложенных ранее индексов этот план не изменится, и лишь сократит время выполнения операций фильтрации и соединения.

Вывод EXPLAIN ANALYZE

QUERY PLAN

```
 Nested Loop  (cost=0.57..143.10 rows=1 width=18) (actual time=0.613..0.614 rows=0 loops=1)
   ->  Nested Loop  (cost=0.28..128.21 rows=1 width=14) (actual time=0.612..0.613 rows=0 loops=1)
         ->  Seq Scan on "Н_ОБУЧЕНИЯ" "обучения"  (cost=0.00..119.76 rows=1 width=10) (actual
time=0.103..0.600 rows=1 loops=1)
               Filter: (("НЗК")::text = '001000'::text)
               Rows Removed by Filter: 5020
         ->  Index Scan using "ЧЛВК_РК" on "Н_ЛЮДИ" "люди"  (cost=0.28..8.30 rows=1 width=4)
(actual time=0.008..0.008 rows=0 loops=1)
               Index Cond: ("ИД" = "обучения"."ЧЛВК_ИД")
               Filter: (("ИМЯ")::text = 'Владимир'::text)
               Rows Removed by Filter: 1
   ->  Index Scan using "УЧЕН_ОБУЧ_FK_I" on "Н_УЧЕНИКИ" "ученики"  (cost=0.29..14.84 rows=5
width=12) (never executed)
         Index Cond: ("ЧЛВК_ИД" = "люди"."ИД")
 Planning Time: 2.047 ms
 Execution Time: 0.701 ms
```

Вывод

В ходе выполнения данной лабораторной работы я изучил планы выполнения запросов в Postgresql