Санкт-Петербургский Национальный Исследовательский Университет ИТМО
Факультет Программной Инженерии и Компьютерной Техники

Вариант №4132
Лабораторная работа №4
По дисциплине
Базы Данных

Выполнил студент группы P3117:
Изаак Герман Константинович

Преподаватель:
Чупанов Аликылыч Алибекович

Санкт-Петербург 2025 г.

1. 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.ИД.
   Фильтры (AND):
   a) H_ОЦЕНКИ.ПРИМЕЧАНИЕ = отлично.
   b) H_ВЕДОМОСТИ.ЧЛВК_ИД < 105590.
   c) H_ВЕДОМОСТИ.ЧЛВК_ИД < 163249.
   Вид соединения: RIGHT 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.ИМЯ, H_ОБУЧЕНИЯ.НЗК, H_УЧЕНИКИ.НАЧАЛО.
   Фильтры: (AND)
   a) H_ЛЮДИ.ОТЧЕСТВО < Георгиевич.
   b) H_ОБУЧЕНИЯ.ЧЛВК_ИД > 105590.
   Вид соединения: LEFT JOIN.

2. Реализация запросов на SQL

```sql
-- first query
SELECT  "Н_ОЦЕНКИ"."КОД", "Н_ВЕДОМОСТИ"."ИД"
FROM    "Н_ОЦЕНКИ"
RIGHT JOIN "Н_ВЕДОМОСТИ"
ON "Н_ОЦЕНКИ"."КОД" = "Н_ВЕДОМОСТИ"."ОЦЕНКА"
WHERE   "Н_ОЦЕНКИ"."ПРИМЕЧАНИЕ" = 'отлично'
  AND   "Н_ВЕДОМОСТИ"."ЧЛВК_ИД" < 105590
  AND   "Н_ВЕДОМОСТИ"."ЧЛВК_ИД" < 163249;


-- second query
SELECT  "Н_ЛЮДИ"."ИМЯ",
        "Н_ОБУЧЕНИЯ"."НЗК",
        "Н_УЧЕНИКИ"."НАЧАЛО"
FROM    "Н_ЛЮДИ"
LEFT JOIN "Н_ОБУЧЕНИЯ"
       ON "Н_ОБУЧЕНИЯ"."ЧЛВК_ИД" = "Н_ЛЮДИ"."ИД"
LEFT JOIN "Н_УЧЕНИКИ"
       ON "Н_УЧЕНИКИ"."ЧЛВК_ИД"    = "Н_ОБУЧЕНИЯ"."ЧЛВК_ИД"
      AND "Н_УЧЕНИКИ"."ВИД_ОБУЧ_ИД" = "Н_ОБУЧЕНИЯ"."ВИД_ОБУЧ_ИД"
WHERE   "Н_ЛЮДИ"."ОТЧЕСТВО" < 'Георгиевич'
  AND   "Н_ОБУЧЕНИЯ"."ЧЛВК_ИД" > 105590;

```

3. Первый запрос

Индексы, добавление которых уменьшит время выполнения запроса:

«Н_ОЦЕНКИ»:

- Индекс на атрибутах **(ПРИМЕЧАНИЕ, КОД)** *(Hash)*: первый столбец — фильтр ПРИМЕЧАНИЕ = 'отлично', поэтому план сможет сразу отобрать только нужные строки. Второй столбец КОД входит в условие соединения, что позволит выполнить «Index Only Scan» без обращения к таблице.

«Н_ВЕДОМОСТИ»:

- Индекс на атрибутах **(ЧЛВК_ИД, ОЦЕНКА)** *(B-tree)*: ведущий столбец ускоряет диапазонный фильтр ЧЛВК_ИД < 105 590, а включённый ОЦЕНКА совпадает с полем JOIN.

Добавление указанных B-tree-индексов снизит объём просматриваемых страниц и ускорит фильтрацию и соединение в запросе.

Возможные планы выполнения запросов без индексов:

План 1:
- Полный скан таблицы **Н_ОЦЕНКИ** (без фильтра).
- Полный скан таблицы **Н_ВЕДОМОСТИ** с применением фильтра по условию ЧЛВК_ИД < 105 590
- RIGHT JOIN по полю КОД = ОЦЕНКА
- После соединения выполняется фильтрация результата: ПРИМЕЧАНИЕ = 'отлично'

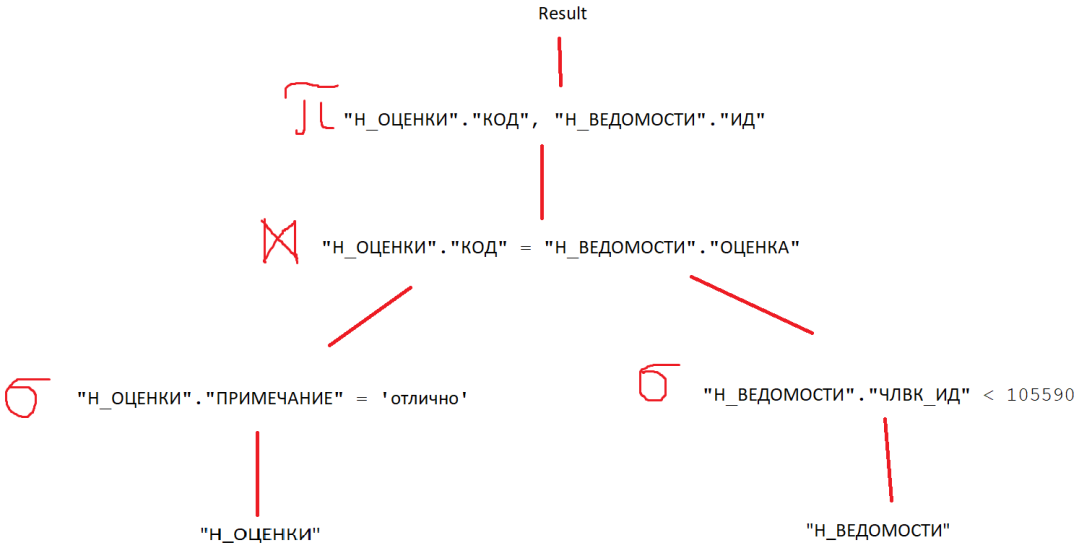
План 2:
- Полный скан **Н_ОЦЕНКИ** + сразу фильтр ПРИМЕЧАНИЕ = 'отлично'
- Полный скан **Н_ВЕДОМОСТИ** + сразу фильтр ЧЛВК_ИД < 105 590.
- RIGHT JOIN по КОД = ОЦЕНКА уже над уменьшенными наборами строк.

Оптимальный план — План 2. Фильтрация выполняется до соединения, поэтому в JOIN участвует гораздо меньше строк; уменьшаются сравнения и объём промежуточных данных.

При добавлении индексов планы выполнения запросов изменятся:

- Вместо полного сканирования «Н_ОЦЕНКИ» выполняется **Index Only Scan** — сразу берутся только коды с пометкой *«отлично»*.
- Соединение по-прежнему реализуется **Nested Loop Join**, но теперь каждая итерация обращается к точному диапазону в индексе, а не просматривает всю таблицу.
- За счёт индексных обращений объём читаемых страниц резко уменьшается, и время выполнения запроса снижается в разы.

План выполнения запроса




```
QUERY PLAN
Nested Loop  (cost=0.29..8.39 rows=1 width=38) (actual time=0.024..0.025 rows=0 loops=1)
  Join Filter: (("Н_ОЦЕНКИ"."КОД")::text = ("Н_ВЕДОМОСТИ"."ОЦЕНКА")::text)
  ->  Seq Scan on "Н_ОЦЕНКИ"  (cost=0.00..1.11 rows=1 width=34) (actual time=0.015..0.016 rows=1 loops=1)
        Filter: (("ПРИМЕЧАНИЕ")::text = 'отлично'::text)
        Rows Removed by Filter: 8
  ->  Index Scan using "ВЕД_ЧЛВК_FK_IFK" on "Н_ВЕДОМОСТИ"  (cost=0.29..7.27 rows=1 width=10) (actual time=0.005..0.005 rows=0 loops=1)
        Index Cond: (("ЧЛВК_ИД" < 105590) AND ("ЧЛВК_ИД" < 163249))
Planning Time: 0.307 ms
Execution Time: 0.058 ms
```

4. Второй запрос

Индексы, добавление которых уменьшит время выполнения запроса:

«Н_ЛЮДИ»

- Индекс на атрибутах (**ОТЧЕСТВО, ИД**) (*B-tree*): Фильтр ОТЧЕСТВО < 'Георгиевич' использует ведущее поле; ИД сразу доступен для присоединения к «Н_ОБУЧЕНИЯ», что уменьшит число обращений к таблице «Н_ЛЮДИ».

«Н_ОБУЧЕНИЯ»

- Индекс на атрибутах (**ЧЛВК_ИД, ВИД_ОБУЧ_ИД, НЗК**) (*B-tree*): Первые два столбца совпадают с парой, по которой Н_УЧЕНИКИ присоединяется к Н_ОБУЧЕНИЯ, значит, SQL сможет взять строку

«обучения» по точному ключу. Третий столбец (НЗК) выводится в SELECT, поэтому планер сможет выполнить Index-Only Scan без обращения к таблице.

«Н_УЧЕНИКИ»

- Индекс на атрибутах (**ЧЛВК_ИД, ВИД_ОБУЧ_ИД, НАЧАЛО**) (*B-tree*): Первые два столбца — точный ключ соединения с Н_ОБУЧЕНИЯ. В индекс включён столбец НАЧАЛО, который нужен в выходном наборе, поэтому чтение тоже может быть «index-only».

Таблицы больше не читаются полностью: каждая операция соединения задействует точечные обращения по составному ключу, а выборка нужных полей выполняется прямо из индексных страниц.

Возможные планы выполнения запросов без индексов:

План 1:

- Полный скан таблицы «Н_ЛЮДИ» без фильтров
- Левое соединение (**LEFT JOIN**) с **Н_ОБУЧЕНИЯ** — полный скан, соединение по ИД = ЧЛВК_ИД
- Левое соединение с **Н_УЧЕНИКИ** — полный скан, соединение по паре (ЧЛВК_ИД,ВИД_ОБУЧ_ИД).
- После всех соединений выполняется фильтрация результата: `"ОТЧЕСТВО" < 'Георгиевич'` и `"ЧЛВК_ИД" > 105 590`.

План 2:

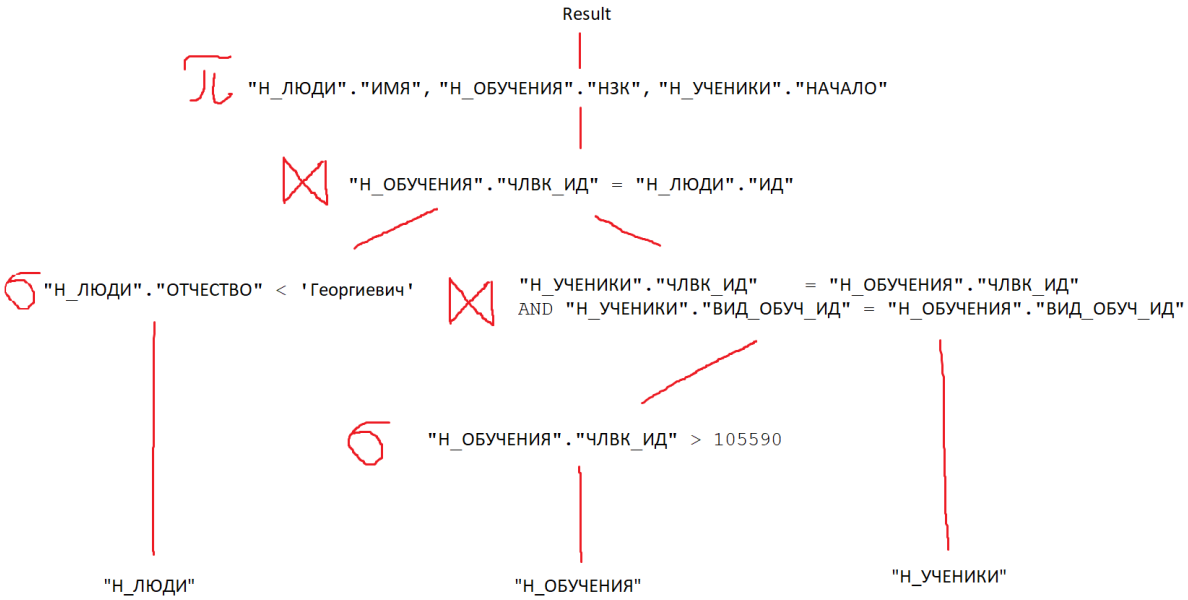
- Полный скан таблицы «**Н_ЛЮДИ**» и применение фильтра ОТЧЕСТВО < 'Георгиевич'
- Полный скан таблицы «**Н_ОБУЧЕНИЯ**» и применения фильтра ЧЛВК_ИД > 105 590
- LEFT JOIN (Nested Loop) «отфильтрованных» **Н_ЛЮДИ ⟶ Н_ОБУЧЕНИЯ**.
- LEFT JOIN (Nested Loop) результата с отфильтрованной подмножеством Н_УЧЕНИКИ

Оптимальный план — План 2. Фильтрация выполняется до соединений, поэтому каждая последующая операция обрабатывает существенно меньше строк; объём промежуточных данных и число сравнений минимальны.

При добавлении индексов планы выполнения запросов изменятся:

- Соединения выполняются тем же **Nested Loop**, но каждая итерация обращается к строкам «Н_ОБУЧЕНИЯ» и «Н_УЧЕНИКИ» по точному ключу, без повторного прохода по таблице.
- Промежуточные наборы резко сокращаются; общее время запроса уменьшается

План выполнения запроса




```
QUERY PLAN
Hash Join  (cost=394.51..1387.20 rows=12903 width=27) (actual time=6.103..22.399 rows=13102 loops=1)
  Hash Cond: ("Н_ОБУЧЕНИЯ"."ЧЛВК_ИД" = "Н_ЛЮДИ"."ИД")
  ->  Hash Right Join  (cost=195.06..1126.62 rows=23266 width=18) (actual time=2.061..13.856 rows=23368 loops=1)
        Hash Cond: (("Н_УЧЕНИКИ"."ЧЛВК_ИД" = "Н_ОБУЧЕНИЯ"."ЧЛВК_ИД") AND ("Н_УЧЕНИКИ"."ВИД_ОБУЧ_ИД" = "Н_ОБУЧЕНИЯ"."ВИД_ОБУЧ_ИД"))
        ->  Seq Scan on "Н_УЧЕНИКИ"  (cost=0.00..809.11 rows=23311 width=16) (actual time=0.006..2.501 rows=23311 loops=1)
        ->  Hash  (cost=119.76..119.76 rows=5020 width=14) (actual time=2.004..2.005 rows=5020 loops=1)
              Buckets: 8192  Batches: 1  Memory Usage: 295kB
              ->  Seq Scan on "Н_ОБУЧЕНИЯ"  (cost=0.00..119.76 rows=5020 width=14) (actual time=0.006..1.043 rows=5020 loops=1)
                    Filter: ("ЧЛВК_ИД" > 105590)
                    Rows Removed by Filter: 1
  ->  Hash  (cost=163.97..163.97 rows=2838 width=17) (actual time=3.996..3.996 rows=2841 loops=1)
        Buckets: 4096  Batches: 1  Memory Usage: 174kB
        ->  Seq Scan on "Н_ЛЮДИ"  (cost=0.00..163.97 rows=2838 width=17) (actual time=0.014..3.392 rows=2841 loops=1)
              Filter: (("ОТЧЕСТВО")::text < 'Георгиевич'::text)
              Rows Removed by Filter: 2277
Planning Time: 1.774 ms
Execution Time: 23.059 ms
```

5. Вывод по работе

В ходе выполнения лабораторной работы я освоил работу с реляционной алгеброй и научился строить планы выполнения запросов, а также их диаграммы. Я изучил различные виды индексов и узнал, как использовать их для оптимизации скорости выполнения запросов. Теперь я могу применять полученные знания для эффективной работы с базами данных и повышения производительности SQL-запросов.