Санкт-Петербургский Национальный Исследовательский Университет ИТМО

Факультет Программной Инженерии и Компьютерной Техники



Вариант № 2025

Лабораторная работа №4

По дисциплине

Базы Данных

Выполнил:

Захра дарабзадех  
студент группы P3130.

Преподаватель:

Перцев Тимофей Сергеевич

Санкт-Петербург 2025 г.

1. **Текст задания**

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: Н\_ЛЮДИ, Н\_СЕССИЯ.  
   Вывести атрибуты: Н\_ЛЮДИ.ФАМИЛИЯ, Н\_СЕССИЯ.УЧГОД.  
   Фильтры (AND):  
   a) Н\_ЛЮДИ.ФАМИЛИЯ < Иванов.  
   b) Н\_СЕССИЯ.УЧГОД > 2008/2009.  
   Вид соединения: LEFT 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.ФАМИЛИЯ, Н\_ОБУЧЕНИЯ.ЧЛВК\_ИД, Н\_УЧЕНИКИ.НАЧАЛО.  
   Фильтры: (AND)  
   a) Н\_ЛЮДИ.ФАМИЛИЯ < Иванов.  
   b) Н\_ОБУЧЕНИЯ.НЗК = 001000.  
   c) Н\_УЧЕНИКИ.НАЧАЛО < 1996-09-01.  
   Вид соединения: RIGHT JOIN.
3. **Реализация запросов на SQL**

--1.Сделать запрос для получения атрибутов из указанных таблиц, применив фильтры по --указанным условиям:  
--Таблицы: Н\_ЛЮДИ, Н\_СЕССИЯ.  
--Вывести атрибуты: Н\_ЛЮДИ.ФАМИЛИЯ, Н\_СЕССИЯ.УЧГОД.  
--Фильтры (AND):  
--a) Н\_ЛЮДИ.ФАМИЛИЯ < Иванов.  
--b) Н\_СЕССИЯ.УЧГОД > 2008/2009.  
--Вид соединения: LEFT JOIN.

EXPLAIN ANALYZE

SELECT Н\_ЛЮДИ.ФАМИЛИЯ, Н\_СЕССИЯ.УЧГОД

FROM Н\_ЛЮДИ

LEFT JOIN Н\_СЕССИЯ ON Н\_ЛЮДИ.ИД = Н\_СЕССИЯ.ЧЛВК\_ИД

WHERE Н\_ЛЮДИ.ФАМИЛИЯ < 'Иванов'

AND Н\_СЕССИЯ.УЧГОД > '2008/2009';

|  |
| --- |
| Plan №1 :  Result  │  Π(люди.фамилия, сессия.учгод)  │  σ (сессия.учгод > '2008/2009')  │  σ (люди.фамилия < 'Иванов')  │  ⋈(люди.ид = сессия.члвк\_ид)    │ │  Н\_ЛЮДИ Н\_СЕССИЯ  Plan №2:  Result  │  Π (люди.фамилия, сессия.учгод)  │  ⋈(люди.ид = сессия.члвк\_ид)  │ │  σ(люди.фамилия < 'Иванов') σ(сессия.учгод > '2008/2009') |

|  |
| --- |
| **Выбираем План №2**, потому что сначала происходит уменьшение объёма данных за счёт применения операций селекции (σ) к каждому из входных отношений, что позволяет JOIN работать с уже отфильтрованными и сокращёнными подмножествами. Это значительно снижает число кортежей на входе в объединение и экономит ресурсы, особенно при работе с большими таблицами, за счёт уменьшения объёмов операций соединения и сокращения I/O  **Индексы**  Чтобы ускорить выполнение этого запроса, рекомендуем создать:  **btree**-индексы на ФАМИЛИЯ и УЧГОД позволяют быстро находить подходящие записи при операциях < и >.  **btree**-индекс на ЧЛВК\_ИД ускоряет поиск строк при соединении таблиц.  -- 1) Для фильтрации по фамилии:  CREATE INDEX idx\_люди\_фамилия ON Н\_ЛЮДИ(ФАМИЛИЯ);  -- 2) Для фильтрации по учёбному году:  CREATE INDEX idx\_сессия\_учгод ON Н\_СЕССИЯ(УЧГОД);  -- 3) Для ускорения JOIN по внешнему ключу:  CREATE INDEX idx\_сессия\_члвк\_ид ON Н\_СЕССИЯ(ЧЛВК\_ИД);  **Влияние на план выполнения:**  После добавления индексов PostgreSQL заменит последовательные сканирования на **индексные сканирования(Index Scan)**, что существенно уменьшит количество обрабатываемых строк:   * Условия фильтрации будут обрабатываться с использованием индексов; * Могут использоваться более эффективные алгоритмы соединения (например, Index Nested Loop Join); * План может измениться с точки зрения порядка соединения таблиц; * Общее время выполнения запроса уменьшится за счет снижения количества операций ввода-вывода и размера промежуточных результатов. |

1. **Реализация запросов на SQL**

--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.ФАМИЛИЯ, Н\_ОБУЧЕНИЯ.ЧЛВК\_ИД, Н\_УЧЕНИКИ.НАЧАЛО.  
--Фильтры: (AND)  
--a) Н\_ЛЮДИ.ФАМИЛИЯ < Иванов.  
--b) Н\_ОБУЧЕНИЯ.НЗК = 001000.  
--c) Н\_УЧЕНИКИ.НАЧАЛО < 1996-09-01.  
--Вид соединения: RIGHT JOIN.

EXPLAIN ANALYZE

SELECT Н\_ЛЮДИ.ФАМИЛИЯ, Н\_ОБУЧЕНИЯ.ЧЛВК\_ИД, Н\_УЧЕНИКИ.НАЧАЛО

FROM Н\_ЛЮДИ

RIGHT JOIN Н\_ОБУЧЕНИЯ ON Н\_ЛЮДИ.ИД = Н\_ОБУЧЕНИЯ.ЧЛВК\_ИД

RIGHT JOIN Н\_УЧЕНИКИ ON Н\_ОБУЧЕНИЯ.ЧЛВК\_ИД = Н\_УЧЕНИКИ.ЧЛВК\_ИД

WHERE Н\_ЛЮДИ.ФАМИЛИЯ < 'Иванов'

AND Н\_ОБУЧЕНИЯ.НЗК = '001000'

AND Н\_УЧЕНИКИ.НАЧАЛО < '1996-09-01';

|  |
| --- |
| Plan №1 :  Result  │  Π (Н\_ЛЮДИ.ФАМИЛИЯ, Н\_ОБУЧЕНИЯ.ЧЛВК\_ИД, Н\_УЧЕНИКИ.НАЧАЛО)  │  Н\_ЛЮДИ ⨝ (Н\_ЛЮДИ.ИД = Н\_ОБУЧЕНИЯ.ЧЛВК\_ИД)  │  ⨝ (Н\_ОБУЧЕНИЯ.ЧЛВК\_ИД = Н\_УЧЕНИКИ.ЧЛВК\_ИД)  │  σ(НЗК = '001000') σ(НАЧАЛО < '1996-09-01')        Н\_ОБУЧЕНИЯ Н\_УЧЕНИКИ    Plan №2 :  Result  │  Π (Н\_ЛЮДИ.ФАМИЛИЯ, Н\_ОБУЧЕНИЯ.ЧЛВК\_ИД, Н\_УЧЕНИКИ.НАЧАЛО)  │  Н\_ЛЮДИ ⨝ (Н\_ЛЮДИ.ИД = Н\_ОБУЧЕНИЯ.ЧЛВК\_ИД)  σ(НЗК = '001000') σ(НАЧАЛО < '1996-09-01')  ⨝ (Н\_ОБУЧЕНИЯ.ЧЛВК\_ИД = Н\_УЧЕНИКИ.ЧЛВК\_ИД)    Н\_ОБУЧЕНИЯ Н\_УЧЕНИКИ |

|  |
| --- |
| Выбранный план: **План №1**  П**отому что** план №1 использует стратегию **"ранней фильтрации" (selection pushdown)**, что позволяет значительно уменьшить количество обрабатываемых строк до выполнения соединения. Это снижает нагрузку на систему и ускоряет выполнение запроса.  **Индексы**  btree-индекс на ФАМИЛИЯ — позволяет быстро находить подходящие записи при сравнении < 'Иванов';  btree-индекс на НАЧАЛО — ускоряет фильтрацию по дате НАЧАЛО < '1996-09-01';  btree-индексы на ЧЛВК\_ИД в таблицах Н\_ОБУЧЕНИЯ и Н\_УЧЕНИКИ — ускоряют поиск строк при соединении таблиц;  btree-индекс на НЗК — оптимизирует точный поиск НЗК = '001000'.  -- 1) Для фильтрации по фамилии:  CREATE INDEX idx\_люди\_фамилия ON Н\_ЛЮДИ(ФАМИЛИЯ);  -- 2) Для фильтрации по дате начала учёбы:  CREATE INDEX idx\_ученики\_начало ON Н\_УЧЕНИКИ(НАЧАЛО);  -- 3) Для фильтрации по НЗК:  CREATE INDEX idx\_обучения\_нзк ON Н\_ОБУЧЕНИЯ(НЗК);  -- 4) Для ускорения JOIN по ЧЛВК\_ИД в обучении:  CREATE INDEX idx\_обучения\_члвк\_ид ON Н\_ОБУЧЕНИЯ(ЧЛВК\_ИД);  -- 5) Для ускорения JOIN по ЧЛВК\_ИД в учениках:  CREATE INDEX idx\_ученики\_члвк\_ид ON Н\_УЧЕНИКИ(ЧЛВК\_ИД);  Влияние на план выполнения:  Без индексов PostgreSQL был бы вынужден использовать Seq Scan для каждой таблицы и менее эффективные алгоритмы соединения, такие как Nested Loop без индекса, что значительно замедлило бы выполнение запроса, особенно при большом объеме данных.  **Влияние добавленных индексов:**  Ускорение фильтрации по ФАМИЛИЯ, НЗК, НАЧАЛО — благодаря Index Scan.  Быстрое соединение таблиц Н\_ЛЮДИ, Н\_ОБУЧЕНИЯ и Н\_УЧЕНИКИ по ЧЛВК\_ИД — благодаря Index Nested Loop Join.  Сокращение общего времени выполнения (в твоём случае — всего 0.212 ms).  Уменьшение количества строк, читаемых с диска.  Повышение масштабируемости при росте данных. |

Вывод

В ходе выполнения лабораторной работы я освоила работу с реляционной алгеброй и научился строить планы выполнения запросов, а также их диаграммы. Я изучил различные виды индексов и узнала, 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.