Отчет по лабораторной работе №4.

Вариант 0900909889899. Раевский Григорий, Р3121.

1. Исходные запросы:
2. SELECT

Н\_ЛЮДИ.ОТЧЕСТВО,

Н\_ВЕДОМОСТИ.ИД

FROM

Н\_ЛЮДИ

LEFT JOIN

Н\_ВЕДОМОСТИ ON Н\_ЛЮДИ.ИД = Н\_ВЕДОМОСТИ.ЧЛВК\_ИД

WHERE

Н\_ЛЮДИ.ИД < 163249 AND

Н\_ВЕДОМОСТИ.ЧЛВК\_ИД > 100012;

1. SELECT

Н\_ЛЮДИ.ИД,

Н\_ОБУЧЕНИЯ.ЧЛВК\_ИД,

Н\_УЧЕНИКИ.НАЧАЛО

FROM

Н\_ЛЮДИ

INNER JOIN

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

INNER JOIN

Н\_УЧЕНИКИ ON Н\_ЛЮДИ.ИД = Н\_УЧЕНИКИ.ЧЛВК\_ИД

WHERE

Н\_ЛЮДИ.ФАМИЛИЯ < 'Афанасьев' AND

Н\_ОБУЧЕНИЯ.ЧЛВК\_ИД > 113409 AND

Н\_УЧЕНИКИ.НАЧАЛО = '1996-09-01';

1. Индексы:
2. B-tree индекс на Н\_ЛЮДИ.ИД и Н\_ВЕДОМОСТИ.ЧЛВК\_ИД. Улучшит скорость выполнения операций сравнения.
3. B-tree индекс на Н\_ЛЮДИ.ИД, Н\_ОБУЧЕНИЯ.ЧЛВК\_ИД, и Н\_УЧЕНИКИ.ЧЛВК\_ИД ускорит операцию соединения.

B-tree на Н\_ЛЮДИ.ФАМИЛИЯ, Н\_ОБУЧЕНИЯ.ЧЛВК\_ИД, и Н\_УЧЕНИКИ.НАЧАЛО.Это ускорит выполнение условий WHERE.

1. Планы:
2. Seq scan таблицы Н\_ЛЮДИ, с учетом Н\_ЛЮДИ.ИД < 163249.

Seq scan таблицы Н\_ВЕДОМОСТИ, с учетом Н\_ВЕДОМОСТИ.ЧЛВК\_ИД > 100012.

LEFT JOIN результатов на основе Н\_ЛЮДИ.ИД = Н\_ВЕДОМОСТИ.ЧЛВК\_ИД.

1. Sequential scan таблицы Н\_ЛЮДИ, с фильтром Н\_ЛЮДИ.ФАМИЛИЯ < 'Афанасьев'.

Sequential scan таблицы Н\_ОБУЧЕНИЯ, с учетом Н\_ОБУЧЕНИЯ.ЧЛВК\_ИД > 113409.

Seq scan таблицы Н\_УЧЕНИКИ, с учетом Н\_УЧЕНИКИ.НАЧАЛО = '1996-09-01'.

INNER JOIN трех результатов на основе Н\_ЛЮДИ.ИД = Н\_ОБУЧЕНИЯ.ЧЛВК\_ИД и Н\_ЛЮДИ.ИД = Н\_УЧЕНИКИ.ЧЛВК\_ИД.

1. Эти планы оптимальны, тк просматривают строки последовательно. Однако их можно улучшить с помощью индексов.
2. Планы с индексами:
3. Index scan таблицы Н\_ЛЮДИ по индексу Н\_ЛЮДИ.ИД с учетом Н\_ЛЮДИ.ИД < 100012.

Index scan таблицы Н\_ВЕДОМОСТИ по индексу Н\_ВЕДОМОСТИ.ЧЛВК\_ИД с учетом Н\_ВЕДОМОСТИ.ЧЛВК\_ИД > 163249.

LEFT JOIN результатов на основе Н\_ЛЮДИ.ИД = Н\_ВЕДОМОСТИ.ЧЛВК\_ИД.

1. Index scan таблицы Н\_ЛЮДИ по индексу Н\_ЛЮДИ.ФАМИЛИЯ и Н\_ЛЮДИ.ИД с учетом Н\_ЛЮДИ.ФАМИЛИЯ < 'Афанасьев'.

Index scan таблицы Н\_ОБУЧЕНИЯ по индексу Н\_ОБУЧЕНИЯ.ЧЛВК\_ИД с учетом Н\_ОБУЧЕНИЯ.ЧЛВК\_ИД > 113409.

Index scan таблицы Н\_УЧЕНИКИ по индексу Н\_УЧЕНИКИ.НАЧАЛО и Н\_УЧЕНИКИ.ЧЛВК\_ИД с учетом Н\_УЧЕНИКИ.НАЧАЛО = '1996-09-01'.

INNER JOIN трех результатов на основе условия Н\_ЛЮДИ.ИД = Н\_ОБУЧЕНИЯ.ЧЛВК\_ИД и Н\_ЛЮДИ.ИД = Н\_УЧЕНИКИ.ЧЛВК\_ИД.

1. В первом запросе без индексов основная операция - это полное сканирование таблицы(медленно и затратно) для применения условий в WHERE и в последующем LEFT JOIN.

Однако, когда мы добавляем индексы на поля Н\_ЛЮДИ.ИД и Н\_ВЕДОМОСТИ.ЧЛВК\_ИД, план выполнения меняется. Становится возможным использовать сканирование индексов вместо сканирования всей таблицы. Это намного эффективнее, особенно если индекс совпадает с условием запроса.

Во втором запросе ситуация аналогична. Без индексов основными операциями будут полное сканирование таблиц и соединение результатов. Но когда мы добавляем индексы на Н\_ЛЮДИ.ФАМИЛИЯ, Н\_ОБУЧЕНИЯ.ЧЛВК\_ИД, и Н\_УЧЕНИКИ.НАЧАЛО, план выполнения меняется. Операции seq scan заменяются на более эффективные операции index scan.

Таким образом, добавление индексов позволяет системе более эффективно использовать ресурсы и быстрее выполнять запросы.

1. Explain analyze:
2. Hash Join (cost=227.85..7658.73 rows=222092 width=24) (actual time=2.256..100.283 rows=222421 loops=1)

Hash Cond: ("Н\_ВЕДОМОСТИ"."ЧЛВК\_ИД" = "Н\_ЛЮДИ"."ИД")

-> Seq Scan on "Н\_ВЕДОМОСТИ" (cost=0.00..6846.50 rows=222440 width=8) (actual time=0.024..45.607 rows=222440 loops=1)

Filter: ("ЧЛВК\_ИД" > 100012)

-> Hash (cost=163.97..163.97 rows=5110 width=24) (actual time=2.170..2.172 rows=5095 loops=1)

Buckets: 8192 Batches: 1 Memory Usage: 356kB

-> Seq Scan on "Н\_ЛЮДИ" (cost=0.00..163.97 rows=5110 width=24) (actual time=0.006..1.205 rows=5095 loops=1)

Filter: ("ИД" < 163249)

Rows Removed by Filter: 23

Planning Time: 0.715 ms

Execution Time: 110.693 ms

1. Nested Loop (cost=6.53..416.33 rows=15 width=16) (actual time=0.416..1.659 rows=8 loops=1)

Join Filter: ("Н\_ЛЮДИ"."ИД" = "Н\_УЧЕНИКИ"."ЧЛВК\_ИД")

-> Nested Loop (cost=6.25..240.49 rows=195 width=8) (actual time=0.119..0.653 rows=195 loops=1)

-> Bitmap Heap Scan on "Н\_ЛЮДИ" (cost=5.96..108.68 rows=217 width=4) (actual time=0.105..0.244 rows=214 loops=1)

Recheck Cond: (("ФАМИЛИЯ")::text < 'Афанасьев'::text)

Heap Blocks: exact=78

-> Bitmap Index Scan on "ФАМ\_ЛЮД" (cost=0.00..5.91 rows=217 width=0) (actual time=0.089..0.090 rows=214 loops=1)

Index Cond: (("ФАМИЛИЯ")::text < 'Афанасьев'::text)

-> Index Only Scan using "ОБУЧ\_ЧЛВК\_FK\_I" on "Н\_ОБУЧЕНИЯ" (cost=0.28..0.60 rows=1 width=4) (actual time=0.001..0.001 rows=1 loops=214)

Index Cond: (("ЧЛВК\_ИД" = "Н\_ЛЮДИ"."ИД") AND ("ЧЛВК\_ИД" > 113409))

Heap Fetches: 0

-> Index Scan using "УЧЕН\_ОБУЧ\_FK\_I" on "Н\_УЧЕНИКИ" (cost=0.29..0.89 rows=1 width=12) (actual time=0.005..0.005 rows=0 loops=195)

Index Cond: ("ЧЛВК\_ИД" = "Н\_ОБУЧЕНИЯ"."ЧЛВК\_ИД")

Filter: ("НАЧАЛО" = '1996-09-01 00:00:00'::timestamp without time zone)

Rows Removed by Filter: 4

Planning Time: 0.766 ms

Execution Time: 1.712 ms

1. Дополнительная теория
2. DQL (Data Query Language): DQL является частью языка SQL (Structured Query Language) и используется для извлечения данных из базы данных. Он предоставляет команды SELECT для выборки данных с использованием различных условий и фильтров.
3. TCL (Transaction Control Language): TCL используется для управления транзакциями в базе данных. Он содержит команды COMMIT для фиксации изменений, ROLLBACK для отката транзакции и SAVEPOINT для создания точек сохранения, позволяющих выполнять частичный откат.
4. DCL (Data Control Language): DCL используется для управления правами доступа и безопасностью в базе данных. Этот язык включает команды, такие как GRANT для предоставления прав доступа к объектам базы данных, REVOKE для отзыва прав доступа и DENY для запрета определенных операций.
5. DML (Data Manipulation Language) расширений: В дополнение к стандартным DML командам INSERT, UPDATE и DELETE, некоторые базы данных предлагают расширенные команды DML для более сложных операций манипуляции данными. Например, в PostgreSQL есть команда UPSERT, которая выполняет обновление, если запись уже существует, и вставку, если записи нет.
6. План запроса для задания 2.4 из ЛР2:
7. Запрос:

EXPLAIN ANALYZE SELECT Н\_ЛЮДИ.ФАМИЛИЯ, COUNT(\*) AS КОЛИЧЕСТВО

FROM Н\_ЛЮДИ

INNER JOIN Н\_УЧЕНИКИ ON Н\_ЛЮДИ.ИД = Н\_УЧЕНИКИ.ЧЛВК\_ИД

INNER JOIN Н\_ФОРМЫ\_ОБУЧЕНИЯ ON Н\_УЧЕНИКИ.ВИД\_ОБУЧ\_ИД = Н\_ФОРМЫ\_ОБУЧЕНИЯ.ИД

WHERE Н\_ФОРМЫ\_ОБУЧЕНИЯ.НАИМЕНОВАНИЕ = 'Очная'

AND Н\_ЛЮДИ.ФАМИЛИЯ IN

(SELECT Н\_ЛЮДИ.ФАМИЛИЯ

FROM Н\_ЛЮДИ

INNER JOIN Н\_УЧЕНИКИ ON Н\_ЛЮДИ.ИД = Н\_УЧЕНИКИ.ЧЛВК\_ИД

INNER JOIN Н\_ФОРМЫ\_ОБУЧЕНИЯ ON Н\_УЧЕНИКИ.ВИД\_ОБУЧ\_ИД = Н\_ФОРМЫ\_ОБУЧЕНИЯ.ИД

WHERE Н\_ФОРМЫ\_ОБУЧЕНИЯ.НАИМЕНОВАНИЕ = 'Очная'

GROUP BY Н\_ЛЮДИ.ФАМИЛИЯ

HAVING COUNT(\*) = 10)

GROUP BY Н\_ЛЮДИ.ФАМИЛИЯ;

1. Без with:
2. Выполнение подзапроса. Выбор таблиц Н\_ЛЮДИ,Н\_УЧЕНИКИ,Н\_ФОРМЫ\_ОБУЧЕНИЯ; объединяются таблицы Н\_УЧЕНИКИ и Н\_ЛЮДИ, Н\_УЧЕНИКИ и Н\_ФОРМЫ ОБУЧЕНИЯ. После чего проверяется условие WHERE. После этого происходит группировка по полю ФАМИЛИЯ с условием, что количество таких людей равно 10.
3. Выбор таблиц Н\_ЛЮДИ, Н\_УЧЕНИКИ, Н\_ФОРМЫ\_ОБУЧЕНИЯ
4. Объединение таблиц Н\_УЧЕНИКИ и Н\_ЛЮДИ по ИД конкретного человека
5. Объединение таблиц Н\_УЧЕНИКИ и Н\_ЛЮДИ по ИД конкретного человека
6. Объединение таблиц Н\_ФОРМЫ\_ОБУЧЕНИЯ и Н\_УЧЕНИКИ по ИД формы обучения
7. Выполнение условия WHERE. Выбираются только те поля, которые одновременно удовлетворяют условию, в котором форма обучения ‘Очная’ и подзапрос верен
8. После выполнения подзапрос вернет фамилии людей, которые подходят по условию.
9. Затем основной запрос производит группировку по полю ФАМИЛИЯ
10. Запрос с with:

EXPLAIN ANALYZE WITH ОФ AS (

SELECT Н\_ЛЮДИ.ФАМИЛИЯ, COUNT(\*) AS КОЛИЧЕСТВО

FROM Н\_ЛЮДИ

INNER JOIN Н\_УЧЕНИКИ ON Н\_ЛЮДИ.ИД = Н\_УЧЕНИКИ.ЧЛВК\_ИД

INNER JOIN Н\_ФОРМЫ\_ОБУЧЕНИЯ ON Н\_УЧЕНИКИ.ВИД\_ОБУЧ\_ИД = Н\_ФОРМЫ\_ОБУЧЕНИЯ.ИД

WHERE Н\_ФОРМЫ\_ОБУЧЕНИЯ.НАИМЕНОВАНИЕ = 'Очная'

GROUP BY Н\_ЛЮДИ.ФАМИЛИЯ

)

SELECT ФАМИЛИЯ, КОЛИЧЕСТВО

FROM ОФ

WHERE КОЛИЧЕСТВО = 10;

1. План запроса с with:
2. Выполнение подзапроса with для создания представления. В этот подзапросе выбираются таблицы Н\_ЛЮДИ, Н\_УЧЕНИКИ, Н\_ФОРМЫ\_ОБУЧЕНИЯ. Эти таблицы объединяются по ИД человека и ИД формы обучения. После этого накладывается фильтр WHERE. После этого произойдет группировка по фамилии и подсчет количества в каждой группе. После этого подзапрос сохранится (представление ФО).
3. В основном запросе выбирается представление ФО, на него накладывается фильтр WHERE. После этого выводится имя и количество для строк, удовлетворяющих условию where.
4. EXPLAIN ANALYZE на первый запрос:

GroupAggregate (cost=1008.01..1008.54 rows=30 width=24) (actual time=21.234..21.467 rows=108 loops=1)

Group Key: "Н\_ЛЮДИ"."ФАМИЛИЯ"

-> Sort (cost=1008.01..1008.09 rows=30 width=16) (actual time=21.228..21.296 rows=1080 loops=1)

Sort Key: "Н\_ЛЮДИ"."ФАМИЛИЯ"

Sort Method: quicksort Memory: 113kB

-> Hash Join (cost=814.53..1007.27 rows=30 width=16) (actual time=17.462..19.576 rows=1080 loops=1)

Hash Cond: ("Н\_УЧЕНИКИ"."ВИД\_ОБУЧ\_ИД" = "Н\_ФОРМЫ\_ОБУЧЕНИЯ"."ИД")

-> Nested Loop (cost=813.46..1005.57 rows=118 width=20) (actual time=17.437..19.297 rows=1080 loops=1)

-> Nested Loop (cost=813.18..993.79 rows=26 width=20) (actual time=17.425..18.788 rows=148 loops=1)

-> HashAggregate (cost=812.89..859.71 rows=19 width=16) (actual time=17.386..17.724 rows=108 loops=1)

Group Key: "Н\_ЛЮДИ\_1"."ФАМИЛИЯ"

Filter: (count(\*) = 10)

Batches: 1 Memory Usage: 721kB

Rows Removed by Filter: 3542

-> Hash Join (cost=215.44..783.75 rows=5828 width=16) (actual time=1.884..12.197 rows=23294 loops=1)

Hash Cond: ("Н\_УЧЕНИКИ\_1"."ЧЛВК\_ИД" = "Н\_ЛЮДИ\_1"."ИД")

-> Nested Loop (cost=0.29..553.29 rows=5828 width=4) (actual time=0.018..5.514 rows=23294 loops=1)

-> Seq Scan on "Н\_ФОРМЫ\_ОБУЧЕНИЯ" "Н\_ФОРМЫ\_ОБУЧЕНИЯ\_1" (cost=0.00..1.05 rows=1 width=4) (actual time=0.002..0.004 rows=1 loops=1)

f) Explain analyze на запрос с with:

1. Индексы для запросов:
2. CREATE INDEX idx\_Н\_ЛЮДИ\_ИД ON Н\_ЛЮДИ (ИД);

CREATE INDEX idx\_Н\_ВЕДОМОСТИ\_ЧЛВК\_ИД ON Н\_ВЕДОМОСТИ (ЧЛВК\_ИД);

1. CREATE INDEX idx\_Н\_ЛЮДИ\_ИД ON Н\_ЛЮДИ (ИД);

CREATE INDEX idx\_Н\_ОБУЧЕНИЯ\_ЧЛВК\_ИД ON Н\_ОБУЧЕНИЯ (ЧЛВК\_ИД);

CREATE INDEX idx\_Н\_УЧЕНИКИ\_ЧЛВК\_ИД ON Н\_УЧЕНИКИ (ЧЛВК\_ИД);

CREATE INDEX idx\_Н\_УЧЕНИКИ\_НАЧАЛО ON Н\_УЧЕНИКИ (НАЧАЛО);