МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ

Федеральное государственное автономное
образовательное учреждение высшего образования
«Санкт-Петербургский национальный исследовательский университет
ИТМО»

ФАКУЛЬТЕТ ПРОГРАММНОЙ ИНЖЕНЕРИИ И КОМПЬЮТЕРНОЙ ТЕХНИКИ

ЛАБОРАТОРНАЯ РАБОТА № 4

по дисциплине

‘Базы данных’

Вариант №313121

Выполнил:
Студент группы P3131
Дворкин Борис
Александрович

Преподаватель:
Наумова Надежда
Александровна

УНИВЕРСИТЕТ ИТМО

Санкт-Петербург, 2023

1. Текст задания

Для выполнения лабораторной работы №4 необходимо:

- 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.

2. Реализация запросов на SQL

```sql
-- 1.
-- Сделать запрос для получения атрибутов из указанных таблиц, применив фильтры по –
-- указанным условиям:
-- Таблицы: Н_ТИПЫ_ВЕДОМОСТЕЙ, Н_ВЕДОМОСТИ.
-- Вывести атрибуты: Н_ТИПЫ_ВЕДОМОСТЕЙ.НАИМЕНОВАНИЕ, Н_ВЕДОМОСТИ.ИД.
-- Фильтры (AND):
-- a) Н_ТИПЫ_ВЕДОМОСТЕЙ.НАИМЕНОВАНИЕ > Ведомость.
-- b) Н_ВЕДОМОСТИ.ДАТА < 1998-01-05.                <- год 2003, там больше народу
-- Вид соединения: INNER JOIN.

SELECT
   Н_ТИПЫ_ВЕДОМОСТЕЙ.НАИМЕНОВАНИЕ,
   Н_ВЕДОМОСТИ.ИД
FROM
   Н_ТИПЫ_ВЕДОМОСТЕЙ
INNER JOIN
   Н_ВЕДОМОСТИ ON Н_ТИПЫ_ВЕДОМОСТЕЙ.ИД = Н_ВЕДОМОСТИ.ТВ_ИД
WHERE
   Н_ТИПЫ_ВЕДОМОСТЕЙ.НАИМЕНОВАНИЕ > 'Ведомость' AND
   Н_ВЕДОМОСТИ.ДАТА < '2003-01-05';

------------------------------------------------------------
-- 2.
-- Сделать запрос для получения атрибутов из указанных таблиц, применив фильтры по -
-- указанным условиям:
-- Таблицы: Н_ЛЮДИ, Н_ВЕДОМОСТИ, Н_СЕССИЯ.
-- Вывести атрибуты: Н_ЛЮДИ.ОТЧЕСТВО, Н_ВЕДОМОСТИ.ЧЛВК_ИД, Н_СЕССИЯ.ИД.
-- Фильтры (AND):
-- a) Н_ЛЮДИ.ИМЯ = Роман.
-- b) Н_ВЕДОМОСТИ.ДАТА = 1998-01-05.       <- аналогично год 2003, там больше народу
-- c) Н_СЕССИЯ.ЧЛВК_ИД = 106059.           <- убрать, иначе ничего не выведется
-- Вид соединения: INNER JOIN.

SELECT Н_ЛЮДИ.ОТЧЕСТВО, Н_ВЕДОМОСТИ.ЧЛВК_ИД, Н_СЕССИЯ.ИД
FROM Н_ЛЮДИ
INNER JOIN Н_ВЕДОМОСТИ ON Н_ЛЮДИ.ИД = Н_ВЕДОМОСТИ.ЧЛВК_ИД
INNER JOIN Н_СЕССИЯ ON Н_ВЕДОМОСТИ.СЭС_ИД = Н_СЕССИЯ.СЭС_ИД
WHERE Н_ЛЮДИ.ИМЯ = 'Роман' AND Н_ВЕДОМОСТИ.ДАТА = '2003-01-05' AND
Н_СЕССИЯ.ЧЛВК_ИД = 106059;
```

3. **Уменьшение выполнения времени 1 запроса**

Индексы, добавление которых уменьшит время выполнения запроса:

a) На таблице Н_ТИПЫ_ВЕДОМОСТЕЙ:

- Индекс на атрибуте НАИМЕНОВАНИЕ (B-tree) Это ускорит фильтрацию строк с условием "Н_ТИПЫ_ВЕДОМОСТЕЙ.НАИМЕНОВАНИЕ > 'Ведомость'".

b) На таблице Н_ВЕДОМОСТИ:

- Индекс на атрибуте ДАТА (B-tree) Это ускорит фильтрацию строк с условием "Н_ВЕДОМОСТИ.ДАТА < '2003-01-05'".
- Индекс на атрибуте ТВ_ИД (B-tree) Это ускорит соединение таблиц по атрибуту ТВ_ИД.

Добавление индексов на указанные столбцы позволит ускорить поиск нужных записей по фильтрам в запросе.

Возможные планы выполнения запросов без индексов:

План 1:

- Полный скан таблицы Н_ТИПЫ_ВЕДОМОСТЕЙ.
- Полный скан таблицы Н_ВЕДОМОСТИ с применением фильтра по условию "Н_ВЕДОМОСТИ.ДАТА < '2003-01-05'".
- Соединение таблиц с использованием Nested Loops Join по атрибуту ТВ_ИД.
- Фильтрация результата соединения по условию "Н_ТИПЫ_ВЕДОМОСТЕЙ.НАИМЕНОВАНИЕ > 'Ведомость'".

План 2:

- Полный скан таблицы Н_ТИПЫ_ВЕДОМОСТЕЙ с применением фильтра по условию "Н_ТИПЫ_ВЕДОМОСТЕЙ.НАИМЕНОВАНИЕ > 'Ведомость'".
- Полный скан таблицы Н_ВЕДОМОСТИ с применением фильтра по условию "Н_ВЕДОМОСТИ.ДАТА < '2003-01-05'".
- Соединение таблиц с использованием Nested Loops Join по атрибуту ТВ_ИД.

Оптимальный план:

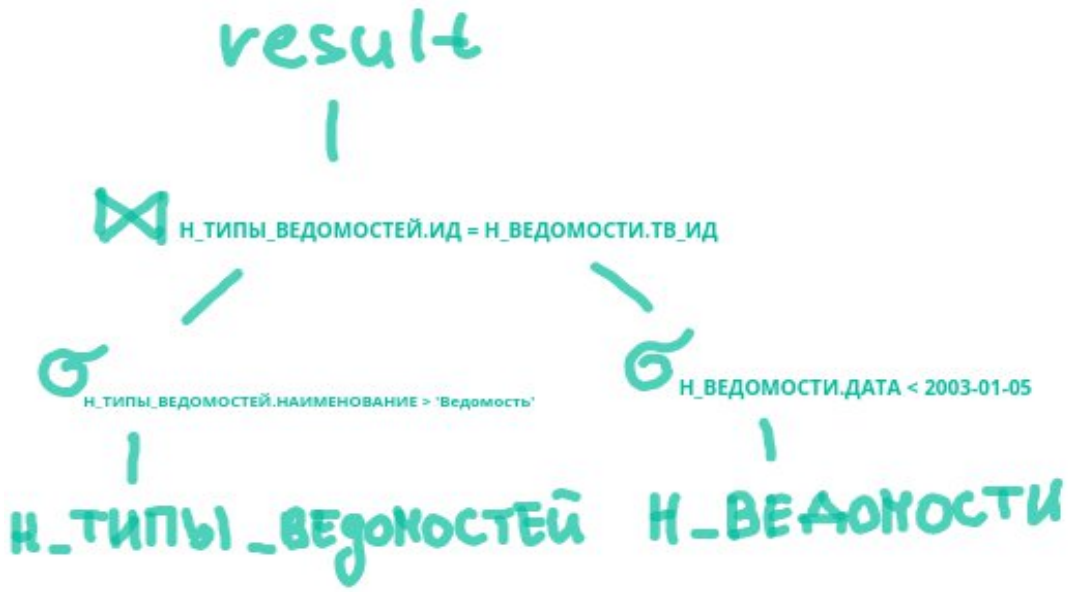
- План 2, потому что фильтрация данных выполняется до соединения таблиц, что уменьшает количество строк для обработки.

При добавлении индексов планы выполнения запросов изменятся:

- Вместо полного скана таблиц будет использоваться индексный скан.
- Nested Loops Join станет быстрее благодаря индексу на атрибуте ТВ_ИД.

4. **План выполнения 1 запроса**

```
 1st QUERY PLAN
------------------------------------------------------------------------------------------------------------------------
 Hash Join  (cost=727.38..5993.93 rows=21249 width=422) (actual time=8.755..24.468 rows=1633 loops=1)
   Hash Cond: ("Н_ВЕДОМОСТИ"."ТВ_ИД" = "Н_ТИПЫ_ВЕДОМОСТЕЙ"."ИД")
   ->  Bitmap Heap Scan on "Н_ВЕДОМОСТИ"  (cost=726.33..5589.15 rows=63746 width=8) (actual time=2.614..15.400 rows=64243 loops=1)
         Recheck Cond: ("ДАТА" < '2003-01-05 00:00:00'::timestamp without time zone)
         Heap Blocks: exact=1476
         ->  Bitmap Index Scan on "ВЕД_ДАТА_I"  (cost=0.00..710.39 rows=63746 width=0) (actual time=2.388..2.389 rows=64243 loops=1)
               Index Cond: ("ДАТА" < '2003-01-05 00:00:00'::timestamp without time zone)
   ->  Hash  (cost=1.04..1.04 rows=1 width=422) (actual time=0.062..0.063 rows=2 loops=1)
         Buckets: 1024  Batches: 1  Memory Usage: 9kB
         ->  Seq Scan on "Н_ТИПЫ_ВЕДОМОСТЕЙ"  (cost=0.00..1.04 rows=1 width=422) (actual time=0.047..0.050 rows=2 loops=1)
               Filter: (("НАИМЕНОВАНИЕ")::text > 'Ведомость'::text)
               Rows Removed by Filter: 1
 Planning Time: 1.332 ms
 Execution Time: 24.661 ms
```



5. **Уменьшение выполнения времени 2 запроса**

Индексы, добавление которых уменьшит время выполнения запроса:

a) На таблице Н_ЛЮДИ:

- Индекс на атрибуте ИМЯ (B-tree). Это ускорит фильтрацию строк с условием "Н_ЛЮДИ.ИМЯ = 'Роман'".

b) На таблице Н_ВЕДОМОСТИ:

- Индекс на атрибуте ДАТА (B-tree). Это ускорит фильтрацию строк с условием "Н_ВЕДОМОСТИ.ДАТА = '2003-01-05'".

c) На таблице Н_СЕССИЯ:

- Индекс на атрибуте ЧЛВК_ИД (B-tree). Это ускорит фильтрацию строк с условием "Н_СЕССИЯ.ЧЛВК_ИД = 106059".

Добавление индексов на указанные столбцы позволит ускорить поиск нужных записей по фильтрам в запросе.

Возможные планы выполнения запросов без индексов:

План 1:
- Полный скан таблицы Н_ЛЮДИ с применением фильтра по условию "Н_ЛЮДИ.ИМЯ = 'Роман'".
- Полный скан таблицы Н_ВЕДОМОСТИ с применением фильтра по условию "Н_ВЕДОМОСТИ.ДАТА = '2003-01-05'".
- Соединение таблиц Н_ЛЮДИ и Н_ВЕДОМОСТИ с использованием Nested Loops Join по атрибуту ЧЛВК_ИД.
- Полный скан таблицы Н_СЕССИЯ с применением фильтра по условию "Н_СЕССИЯ.ЧЛВК_ИД = 106059".
- Соединение результатов предыдущего этапа с таблицей Н_СЕССИЯ с использованием Nested Loops Join по атрибуту СЭС_ИД.

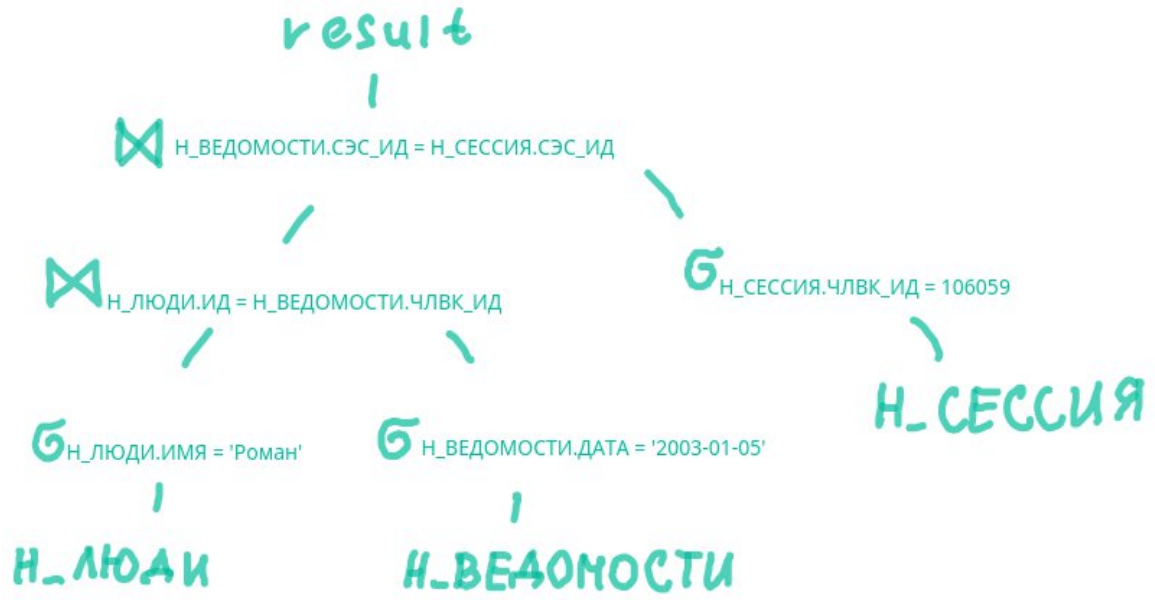
План 2:
- Полный скан таблицы Н_ЛЮДИ с применением фильтра по условию "Н_ЛЮДИ.ИМЯ = 'Роман'".
- Полный скан таблицы Н_ВЕДОМОСТИ с применением фильтра по условию "Н_ВЕДОМОСТИ.ДАТА = '2003-01-05'".
- Соединение таблиц Н_ЛЮДИ и Н_ВЕДОМОСТИ с использованием Hash Join по атрибуту ЧЛВК_ИД.
- Полный скан таблицы Н_СЕССИЯ с применением фильтра по условию "Н_СЕССИЯ.ЧЛВК_ИД = 106059".
- Соединение результатов предыдущего этапа с таблицей Н_СЕССИЯ с использованием Hash Join по атрибуту СЭС_ИД.

Оптимальный план:
- План 1, так как Nested Loops Join предпочтительнее Hash Join в случаях, когда обрабатываемые наборы данных небольшие, и заранее известно, что результаты фильтрации будут содержать малое количество строк..

При добавлении индексов планы выполнения запросов изменятся:
- Вместо полного скана таблиц будет использоваться индексный скан.
- Nested Loops Join станет быстрее благодаря индексам на атрибутах ЧЛВК_ИД и СЭС_ИД.

6. **План выполнения 2 запроса**

```
2nd QUERY PLAN
----------------------------------------------------------------------------------------------------------------------------------
 Nested Loop  (cost=14.02..1736.27 rows=7 width=28) (actual time=0.863..2.760 rows=8 loops=1)
   ->  Nested Loop  (cost=13.74..1727.12 rows=11 width=28) (actual time=0.085..2.724 rows=5 loops=1)
         ->  Seq Scan on "Н_ЛЮДИ"  (cost=0.00..163.97 rows=88 width=24) (actual time=0.021..0.878 rows=88 loops=1)
               Filter: (("ИМЯ")::text = 'Роман'::text)
               Rows Removed by Filter: 5030
         ->  Bitmap Heap Scan on "Н_ВЕДОМОСТИ"  (cost=13.74..17.75 rows=1 width=8) (actual time=0.019..0.019 rows=0 loops=88)
               Recheck Cond: (("ЧЛВК_ИД" = "Н_ЛЮДИ"."ИД") AND ("ДАТА" = '2003-01-05 00:00:00'::timestamp without time zone))
               Heap Blocks: exact=5
               ->  BitmapAnd  (cost=13.74..13.74 rows=1 width=0) (actual time=0.018..0.018 rows=0 loops=88)
                     ->  Bitmap Index Scan on "ВЕД_ЧЛВК_FK_IFK"  (cost=0.00..4.08 rows=68 width=0) (actual time=0.006..0.006 rows=37 loops=88)
                           Index Cond: ("ЧЛВК_ИД" = "Н_ЛЮДИ"."ИД")
                     ->  Bitmap Index Scan on "ВЕД_ДАТА_I"  (cost=0.00..9.24 rows=660 width=0) (actual time=0.016..0.016 rows=635 loops=57)
                           Index Cond: ("ДАТА" = '2003-01-05 00:00:00'::timestamp without time zone)
   ->  Index Scan using "СЕС_СЭС_FK" on "Н_СЕССИЯ"  (cost=0.28..0.81 rows=2 width=8) (actual time=0.005..0.006 rows=2 loops=5)
         Index Cond: ("СЭС_ИД" = "Н_ВЕДОМОСТИ"."СЭС_ИД")
 Planning Time: 1.082 ms
 Execution Time: 2.808 ms
```



7. Вывод

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.

Доп.задание:

```
-- Доп: написать триггер, который будет выводить информацию об объекте,
-- который был добавлен/удалён в табличку ship_type.
--

-- Удаляем существующие триггеры, если они существуют
DROP TRIGGER IF EXISTS ship_type_after_insert ON ship_type;
DROP TRIGGER IF EXISTS ship_type_before_delete ON ship_type;
DROP TRIGGER IF EXISTS ship_type_delete_cascade_trigger ON ship_type;

-- Создаем функцию, которая вызывается при вставке записи в таблицу
ship_type
CREATE OR REPLACE FUNCTION ship_type_insert_trigger() RETURNS TRIGGER
AS $$
BEGIN
    RAISE NOTICE 'INSERT: Ship Type with ID %, Type %, Capacity %, Max
Speed %, and Range % has been added.', NEW.id, NEW.ship_type,
NEW.ship_capacity, NEW.max_speed, NEW.range;
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

-- Создаем функцию, которая вызывается при удалении записи из таблицы
ship_type
CREATE OR REPLACE FUNCTION ship_type_delete_trigger() RETURNS
TRIGGER AS $$
BEGIN
    RAISE NOTICE 'DELETE: Ship Type with ID %, Type %, Capacity %, Max
Speed %, and Range % has been deleted.', OLD.id, OLD.ship_type,
OLD.ship_capacity, OLD.max_speed, OLD.range;
    RETURN OLD;
END;
$$ LANGUAGE plpgsql;

-- Создаем триггер, который вызывает функцию ship_type_insert_trigger при
вставке записи в таблицу ship_type
CREATE TRIGGER ship_type_after_insert
    AFTER INSERT ON ship_type
    FOR EACH ROW
    EXECUTE FUNCTION ship_type_insert_trigger();

-- Создаем триггер, который вызывает функцию ship_type_delete_trigger при
удалении записи из таблицы ship_type
CREATE TRIGGER ship_type_before_delete
    BEFORE DELETE ON ship_type
    FOR EACH ROW
    EXECUTE FUNCTION ship_type_delete_trigger();

-- Создаем функцию, которая вызывается перед удалением записи из таблицы
ship_type
CREATE OR REPLACE FUNCTION ship_type_delete_cascade() RETURNS
TRIGGER AS $$
```

```
BEGIN
    DELETE FROM ship_troubles WHERE ship_id IN (SELECT id FROM ship
WHERE ship_type = OLD.id);
    DELETE FROM ship WHERE ship_type = OLD.id;
    RETURN OLD;
END;
$$ LANGUAGE plpgsql;

-- Создаем триггер, который вызывает функцию ship_type_delete_cascade при
удалении записи из таблицы ship_type
CREATE TRIGGER ship_type_delete_cascade_trigger
    BEFORE DELETE ON ship_type
    FOR EACH ROW
    EXECUTE FUNCTION ship_type_delete_cascade();

-----------------------------------------------------------------------------------------

-- Давайте покажем функциональность написанных триггеров и процедурных
функций

INSERT INTO ship_type (ship_type, ship_capacity, max_speed, range)
VALUES ('Cargo', 100, 400, 3000),
       ('Passenger', 200, 600, 5000);

DELETE FROM ship_type WHERE id = 1;
```