УДК 004.6

Нурматова Елена Вячеславовна,
доцент кафедры аппаратного, программного и математического обеспечения вычислительных систем, кандидат технических наук, Института кибербезопасности и цифровых технологий РТУ МИРЭА
nurmatova@mirea.ru

Эм Ирина Дмитриевна,
студентка 4 курса специальности «Информационные системы и технологии» Института кибербезопасности и цифровых технологий РТУ МИРЭА

РЕАЛИЗАЦИЯ МОДУЛЯ ОПТИМИЗАЦИИ ЗАПРОСОВ ЧТЕНИЯ ДАННЫХ

Аннотация. *В данной работе были исследованы причины, влияющие на ресурсоемкость запросов чтения данных, объясняются основы физических операций, обеспечиваемых подсистемой выполнения запросов и описаны результаты разработки модуля оптимизации запросов чтения данных.*

Ключевые слова: *реляционные системы, оптимизация запросов, select, эффективность индексов, статистика, оценка стоимости.*

На время выполнения запроса влияют следующие показатели, объединенные в одну функцию стоимости: циклы процессора, количество операций ввода-вывода (чтение и запись дисковых блоков), а также количество доступной памяти (влияет на соотношение циклов процессора и количество операций ввода-вывода). Вычислить запрос в реляционной БД помогают оптимизатор запросов и подсистема выполнения запросов, предоставляющая процедурный план выполнения запроса[1]. Стоимостная модель физической операции оценивает ресурсы, необходимые для выполнения операции.

Выделим основные причины, которое могут влиять на скорость запроса[2]. Их характеристики приведены в табл. 1.

[1] Домбровская Г., Новиков Б., Бейликова А. Оптимизация запросов в PostgreSQL / пер. с англ. Д.А. Беликова. М. ДМК Пресс, 2022. 278 с.: ил.

[2] Болье, А. Изучаем SQL. Генерация, выборка и обработка данных, 3-е изд./ Алан Болье; пер. с англ. И.В. Красикова. Киев: «Диалектика», 2021. 402 с.: ил.

Таблица 1. Причины медленных запросов

Основание	Вариант решения
Наличие и качество собранной статистики таблиц и индексов, используемым в запросах	● Подбор и следование стратегии индексирования с соблюдением критериев по производительности ● Сбор и обновление статистики по индексам
Наличие и правильность применения индексов	● Создание/ изменение индекса по данным, входящим в запрос в части where/ having/ group by/order by ● Модификикация запроса с целью освобождения от блокирующих операций или создать индекс по функции, блокирующей индекс ● Сбор статистики по индексу ● Проверка наличия хинтов, блокирующих работу индекса, например NO_INDEX ● Проверка настройки эффективных параметров инициализации БД ● Проверка наличия более «сильных» индексов ● Проверка процента разрежённости индекса
Наличие и эффективность хинтов	● Проверка наличия и правильности следования хинтов ● Проверка отсутствия или изменения индекса, указанного в хинте
Качество построения структуры запроса	● Следование рекомендациям по написанию эффективного sql-кода ● Перестройка логических условий части where/ having в соответствии с правилами минимизации логических функций ● Использование приёмов улучшения запросов с соединениями данных из нескольких таблиц/ БД. ● Использование хинтов
Правильность настройки параметров инициализации БД	● Сократить задержки и уменьшить время необходимое для увеличения файлов данных ● Сократить время старта СУБД ● Сократить время при восстановлении БД

Для тестирования реализованного модуля был использован запрос на выборку к БД, который был выполнен без применения индексации и с ним. Выборка производилась на основе схемы из трёх таблиц: на 59 280 записей, на3 724 654 записей и на 180 записей, соответственно. Запрос включал операцию

проекции на пять колонок из этих таблиц, с двойным соединением таблиц и дополнительной горизонтальной селекцией.

Результирующие планы изображены на рис. 1. Как видно по рис. 1, наивысшая стоимость для одной операции – 91% - для выполненного запроса до применения индексации. После применения индексации, стоимость имеет чуть более распределенный характер. Так, наибольшая стоимость после индексации – 43%.

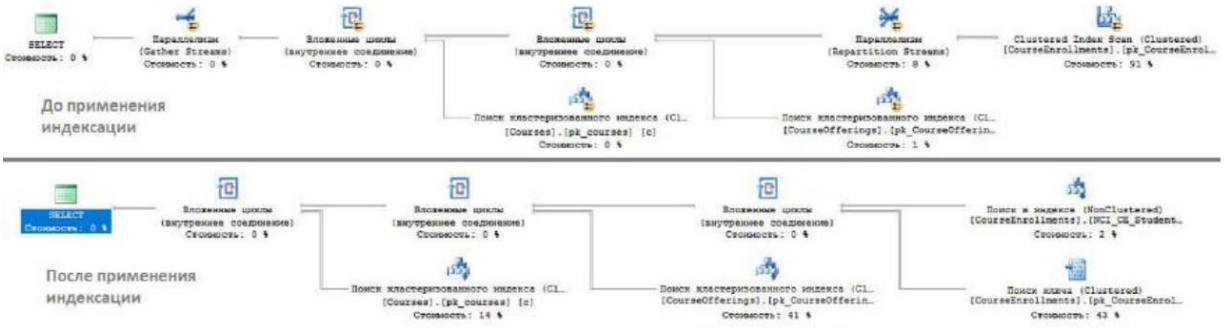

Рисунок 1. Вид планов выполнения до и после индексации

В табл. 2 сравнивается скорость выполнения запроса до и после применения индексации.

Таблица 2. Скорости выполнения запроса

Метрика	Без индексации	С индексацией
Операция доступа к данным	Сканирование кластерного индекса	Поиск по ключу
Стоимость выражения	12,013	0,089
Стоимость операции ввода/вывода	8,874	0,003
Время CPU (мс)	219	16
Затраченное время (мс)	195	63

Из данных таблицы видно, что индексация дает ощутимый прирост в производительности выполнения запроса. Нагрузка на процессор уменьшилась в 13.7 раз, а затраченное время в 3,1 раз.

Другой вариант тестирования модуля осуществлялся по запросу к секционированной таблице, состоящей из одной секции, секционированной по столбцу a. При этом, был использован индекс по столбцу b, в запросе использована сортировка по данному полю. В результате, было рекомендовано применение

хинта “ENABLE_QUERY_OPTIMIZER_HOTFIX”, который включает исправления оптимизатора в определенной БД.

На рис. 2 показан фрагмент кода хранимой функции, формирующей подсказку для оптимизации поступившего для анализа select-запроса. Показано, каким образом модулем находятся ключевые слова на проверку запроса и подбора подсказки.

```
-- Проверяем количество соединений таблиц и их именование
DECLARE @count_join varchar(15);
DECLARE @count_as varchar(15);

-- Считаем количество 'JOIN' и 'AS'
SELECT @count_join = count(*) from @columns group by RowName having RowName LIKE 'JOIN';
SELECT @count_as = count(*) from @columns group by RowName having RowName LIKE 'AS';

IF @count_as is NULL
    SET @count_as = 0;

IF @count_join is NULL
    SET @count_join = 0;

IF @count_as - 1 != @count_join and @count_join != 0
    INSERT INTO @res_table (text) VALUES('Используйте "AS" для упрощенного именования таблиц.')

IF @count_join > 3
    INSERT INTO @res_table (text) VALUES('Сделать из запроса представление, что ускорит получение данных.')

-- Оптимизация группировок

-- Разделение запроса
DECLARE @split_request_agr varchar(100);

-- Ищем группировку
IF (SELECT RowName FROM @columns WHERE RowName LIKE 'GROUP') IS NOT NULL and (SELECT RowName FROM @columns WHERE RowName LIKE 'BY') IS NOT NULL
BEGIN
    -- Ищем любое вхождение агрегирующей функции
    SELECT @split_request_agr = RowName FROM @columns WHERE RowName LIKE 'COUNT(%'
        OR RowName LIKE 'AVG(%' OR RowName LIKE 'SUM(%' OR RowName LIKE 'MIN(%' OR RowName LIKE 'MAX(%';

    IF @split_request_agr IS NULL
        INSERT INTO @res_table (text) VALUES('Использовать DISTINCT вместо GROUP BY. ')
END
```

Рисунок 2. Создание функции, принимающей select-запрос на оптимизацию

На рис. 3 приведён пример вывода рекомендаций по оптимизации запроса, соединяющего данные из двух таблиц без дополнительной селекции.

```
SELECT * from Check_Select(N'SELECT top 7 customers.name_customer, customers.id_clothes, clothes.id, clothes.title
FROM customers join clothes on customers.id_clothes = clothes.id');
GO
```

90 %

Результаты | Сообщения

	text
1	Рекомендации по оптимизации:
2	Используйте "AS" для упрощенного именования таблиц.
3	Рекомендуется создать индексы для: customers.name_customer
4	Воспользуйтесь "OPTION (FORCE ORDER)".

Рисунок 3. Результат работы модуля

Представление запроса путем его синтаксических и семантических преобразований позволяет определить место, которое можно оптимизировать при обращении к базе данных (БД)[3]. Но, важно отметить, что оптимизация запроса предполагает намного более широкую работу, чем преобразования и оценка эквивалентности запросов, применение эффективных индексов, хинтов, обновление статистики и т.п. Несмотря на многие годы работы и исследований в этой области, существенные проблемы остаются открытыми.

Список источников и литературы

1. Домбровская Г., Новиков Б., Бейликова А. Оптимизация запросов в PostgreSQL / пер. с англ. Д.А. Беликова. М. ДМК Пресс, 2022. 278 с.: ил.

2. Болье А. Изучаем SQL. Генерация, выборка и обработка данных, 3-е изд./ Алан Болье; пер. с англ. И.В. Красикова. Киев: «Диалектика», 2021. 402 с.: ил.

3. Льюис, Джонатан. Oracle. Основы стоимостной оптимизации / Дж. Льюис; [пер. с англ.: В. Голубев, В. Щербинин]. Санкт-Петербург [и др.]: Питер, 2007 (СПб.: Печатный двор им. А.М. Горького). 527 с.: ил., табл.; 23 см.

[3] Льюис, Джонатан. Oracle. Основы стоимостной оптимизации / Дж. Льюис; [пер. с англ.: В. Голубев, В. Щербинин]. Санкт-Петербург [и др.]: Питер, 2007 (СПб.: Печатный двор им. А.М. Горького). - 527 с.: ил., табл.; 23 см.