DOI 10.37882/2223-2966.2024.7.22

ОПТИМИЗАЦИЯ ЗАПРОСОВ В РЕЛЯЦИОННЫХ БАЗАХ ДАННЫХ

QUERY OPTIMIZATION IN RELATIONAL DATABASES

A. Conde

Summary. This article discusses query optimization technologies in relational databases. The purpose of this article is to optimize the query structure in relational databases when processing large amounts of information. As a result of the work, the main provisions were developed in the form of the results of the analysis of technologies for improving database performance using indexing technologies, methods for optimizing the database structure to improve performance, using the identification of bottlenecks and problematic queries, as well as monitoring and testing database performance.

Keywords: query optimization, relational database, indexing, stored procedures, database performance monitoring.

Конде Абдул Карим
Аспирант, Московский государственный технологический университет «СТАНКИН», г. Москва
karimconde112@gmail.com

Аннотация. В данной статье рассматриваются технологии оптимизации запросов в реляционных базах данных. Целью данной статьи является оптимизация структуры запросов в реляционных базах данных при обработке больших объемов информации. В результате выполнения работы разработаны основные положения в виде результатов анализа технологий повышения производительности баз данных с использованием технологий индексации, методики оптимизации структуры базы данных для повышения производительности, использующая идентификацию узких мест и проблемных запросов, а также проведение мониторинга и тестирования производительности базы данных.

Ключевые слова: оптимизация запросов, реляционная БД, индексация, хранимые процедуры, мониторинг производительности БД.

Поддержка большинства бизнес-процессов по всех сферах деятельности ведется сегодня с использованием как минимум учетных систем, а при необходимости проведения анализа с применением разного рода систем принятия решений. Структуры данных, которые используются для ведения учета и анализа данных, формируются на результате сбора разного рода структурированных и неструктурированных данных. При этом ведение учета возможно только на основе работы со структурированными особым образом данными. Построенная стройная теория организации обработки данных с использованием принципов реляционной алгебры до сих пор является оптимальным решением для получения оперативной учетной информации и извлечения необходимых для ведения деятельности данных.

Однако расширение спектра систем и компаний, вышедших на интернет-площадки, привело к резкому росту объемов данных, которые используются для организации работы даже в средней компании. В результате уже хорошо зарекомендовавшие себя средства обработки и хранения данных в виде популярных СУБД не полной мере справляются с поставленными задачами. Вариантом решения данной проблемы может стать применение новых видов нереляционных баз данных, которые работают с частично неструктурированной информацией. Однако такие базы не совсем удачно использовать для решения учетных задач и получения оперативных данных. Второй вариант решения связан с оптимизацией структуры запросов в существующей базе данных. Исследованию вопросов оптимизации запросов в реляционной базе данных при обработке больших объемов информации и посвящена данных работа.

Структура базы данных предполагает разделение данных для хранения в таблицах, однако используется обычно уже сводная информация, полученная в ходе выбора по некоторым свойствам и проведения агрегирования. Механизм запросов как основной инструмент работы с базой данных также предполагает несколько путей оптимизации, начиная от перестроения структуры запросов и заканчивая использованием встроенных функций.

Оптимизации структуры запросов включает организацию работы по следующим направлениям: корректное применение индексации в структуре запросов; выбор оптимальной структуры запросов для сокращения объема обрабатываемых данных; применение технологий подзапросов для отбора данных со сложными условиями; приведение агрегирования данных в рамках структуры запроса, а не на прикладном уровне в ходе работы приложения.

Корректное применение индексации в структуре запросов предполагает предварительный анализ использования индексов для оператора SELECT, так как при большом числе операций по вставке и обновлению данных операторами UPDATE, DELETE и INSERT индексация серьезным образом снижает скорость обработки данных. Отбор записей согласно определенным условиям должен включать одновременное использование разделов WHERE и LIMIT для фильтрации и оценка количества возвращаемых записей. Дополнительно перед отбором данных должны использоваться оператор JOIN для отражения соединения записей таблиц. Применение подзапросов в ходе построения сложных условий и выпол-

нения агрегирования может как повысить, так и снизить скорость обработки данных если запрос имеет очень сложную структуру.

Снижение времени на обработку данных может быть реализовано также путем передачи уже отобранных и агрегированных данных вместо формирования списка всех записей для расчета необходимых показателей на сервере приложений. Учитывая приведенные особенности оптимизации структуры запросов, перед применением любой технологии оптимизации структуры запросов необходимо провести всесторонний анализ проводимых изменений с оценкой предполагаемого уровня повышения производительности корректируемой базы данных.

Хранимые процедуры, реализующие наиболее частые операции по оперированию данными, также могут позволить повысить производительность. Применение хранимых процедур позволяет передавать на сервер приложений не полную структуру запроса и результат в виде отобранных записей, а уже проведенные изменения или параметры хранимой процедуры. Таким образом, хранимая процедура выполняет часть операций по работе сданных на севере СУБД вместо передачи управления серверу приложений [12].

Несмотря на то, что хранимые процедуры реализуются на стороне сервера, они позволяют существенно оптимизировать его работу и повысить ее скорость. Это становится возможным благодаря тому, что код процедуры после первого ее выполнения сохраняется в кэше оперативной памяти. Таким образом, для дальнейших вызовов, процедура будет извлекается из кэша, в место того, чтобы снова выполняться. Также хранимые процедуры увеличивают скорость выполнения запросов, поскольку они не требуют компиляции — они хранятся уже в скомпилированном виде. Применение хранимых процедур переносит нагрузку с клиентской стороны, на сторону сервера, из этого можно сделать вывод, что применение хранимых процедур оправдано в том, случае, когда один сервер, обслуживает большой поток клиентов.

Как и в случае с оптимизацией структуры запросов базы данных использование хранимых процедур без предварительного анализа может привести не к повышению производительности базы данных, а к ее снижению.

Современные базы данных, в том числе объектно-ориентированные типа NoSQL предоставляют также инструменты для создания разного типа индексов. Что также способствует повышению производительности базы данных. Могут быть реализованы B-деревья, а также растровые индексы, хеш-индексы [13].

Частью оптимальной стратегии повышения производительности конкретной базы данных становится постоянный мониторинг используемости введенных дополнительных индексов и их реорганизация по необходимости.

Таким образом сама технология введения индексации напрямую не способствует повышению производительности, требуется разработка особой политики работы с индексами в конкретной базе данных, привязанной с возможным изменениям, которые будут происходить в структуре более высоких уровней приложений, использующих базы данных в качестве средства управления данными и обеспечения их хранения [14].

Для демонстрации технологий оценки производительности используется тестовая база данных AdventureWorks, созданная компанией Microsoft для тестирования различных инструментов выполнение анализа данных и оценки производительности. База описывает технологии производства и продаж велосипедов, которые реализуются на коммерческих рынках Северной Америки, Европы и Азии через региональные представительства. Велосипеды производятся из металла и композитных материалов.

Проблемы возникающие в процессе работы с базой данных могут быть вызваны несоответствием используемого оборудования требованиям производителя соответствующей СУБД. В данном случае рассматривается не условно определенная загрузка дискового пространства и памяти сервера базы данных, но и необходимость по резервированию памяти и самого дискового пространства.

При соблюдении этих условий возможно проведение анализа с учетом проблемных запросов, которые могут вызывать задержки в обработки и становится в ожидающие задачи, а также выполняться достаточно долго с учетом потребностей пользователей и их числа (рисунок 1).

В данном случае рассматривается процесс разработки, что определяет малую загрузку сервера в процессе работы.

Для оценки времени исполнения запросов к базе данных и числа обращений используется мониторинг активности и статистика [15]. Например, данные по ресурсоёмким запросам. Для анализа структуры запроса используется предполагаемый план выполнения. По характеристикам каждой операции может быть выбран вариант изменения. В качестве другого примера можно рассмотреть технологию выполнения запроса (рисунок 2).

Последние ресурсоемкие запросы

Запрос	Число выполнени в минуту	ЦП (мс/сек)	Число физических операций чтения в секунду	Число логических операций записи в секунду	Число логических операций чтения в секунду	Средняя продолжите (мсек)	Число планов	Database Name
select xp.name, cast(xp.[value] as nvarchar(40...	0	0	0	0	0	0	34	AdventureWorks2008R
select xp.name, cast(xp.[value] as nvarchar(40...	0	0	0	0	0	1	17	AdventureWorks2008R
select xp.name, cast(xp.[value] as nvarchar(40...	0	0	0	0	0	3	15	AdventureWorks2008R
SELECTCAST(fti.is_enabled AS bit) AS [IsEnab...	12	0	0	0	1	0	1	AdventureWorks2008R
SELECTclmns.name AS [Name]FROMsys.all_vi...	15	0	0	0	2	4	1	AdventureWorks2008R
SELECTclmns.name AS [Name],clmns.column_...	0	0	0	0	0	2	1	AdventureWorks2008R
SELECTSCHEMA_NAME(sp.schema_id) AS [S...	0	0	0	0	0	517	1	AdventureWorks2008R

Рис. 1. Данные по ресурсоемким запросам базы данных

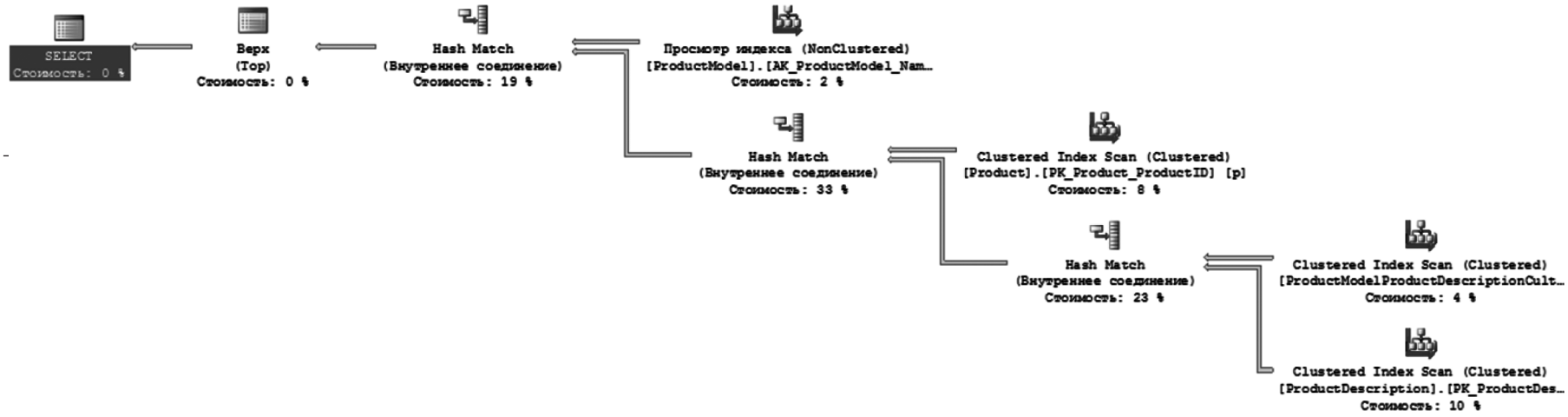

Рис. 2. Структура данных запроса с технологий поиска по описанию

Возможная перестройка запроса проводится на результатах анализа предварительного плана выполнения с учетом всех особенностей. В данном запросе используется таблица с составным индексом и связанные объекты. Для каждого используемого индекса формируется соответствующая статистика для оценки частоты использования и возможностей его перестроения или реорганизации.

Анализ возможных проблем базы данных можно провести также с учетом физического анализа индексации (рисунок 3).

Перестроение сформирует новые индексы без сохранения старых. Для кластеризованных индексов обычно предлагается реорганизация, которая состоит в дефрагментации уже построенных страниц.

Оценка производительности в этом аспекте производится с учетом данных запросов выполняется для самых затратных запросов. Выделяются пакеты и запросы с: наибольшим общим временем загрузки ЦП; наибольшим средним временем загрузки ЦП; наибольшим средним числом операций ввод и вывода и наибольшим общим и средним числом операций ввод и вывода.

В результате анализа предлагается перестроить или реорганизовать, например, для таблицы Production. Product (рисунок 4), что позволит увеличить плотность страниц.

По результатам такой оценки производительности можно ограничится анализом конкретных запросов и технологиями работы с ними.

Таким образом, введение индексов наращивает производительность базы данных, но вместе с тем загружает часть памяти и дискового пространства, что может снизить общую скорость обработки данных. При этом занятое дисковое пространство в базе данных не обязательно напрямую таковым является и при необходимости это можно откорректировать, например, проанализировав результаты отчета с указанием зарезервированного и используемой дисковой памяти.

⊟ Production.Location

Имя индекса	Тип индекса	Число секций	Глубина	Рекомендованная операция
AK_Location_Name	NONCLUSTERED INDEX	⊞ 1	1	-
PK_Location_LocationID	CLUSTERED INDEX	⊟ 1	1	-

Номер секции	Средняя фрагментация (%)	Число фрагментов	Среднее число страниц на фрагмент	Число страниц
1	0	1	1	1

⊟ Production.Product

Имя индекса	Тип индекса	Число секций	Глубина	Рекомендованная операция
AK_Product_Name	NONCLUSTERED INDEX	⊟ 1	2	Перестроить

Номер секции	Средняя фрагментация (%)	Число фрагментов	Среднее число страниц на фрагмент	Число страниц
1	67	3	1	3

Имя индекса	Тип индекса	Число секций	Глубина	Рекомендованная операция
AK_Product_ProductNumber	NONCLUSTERED INDEX	⊟ 1	2	Перестроить

Номер секции	Средняя фрагментация (%)	Число фрагментов	Среднее число страниц на фрагмент	Число страниц
1	50	2	1	2

Имя индекса	Тип индекса	Число секций	Глубина	Рекомендованная операция
AK_Product_rowguid	NONCLUSTERED INDEX	⊟ 1	2	Перестроить

Номер секции	Средняя фрагментация (%)	Число фрагментов	Среднее число страниц на фрагмент	Число страниц
1	50	2	1	2

Имя индекса	Тип индекса	Число секций	Глубина	Рекомендованная операция
PK_Product_ProductID	CLUSTERED INDEX	⊟ 1	2	Реорганизовать

Номер секции	Средняя фрагментация (%)	Число фрагментов	Среднее число страниц на фрагмент	Число страниц
1	23	4	3	13

Рис. 3. Фрагмент физической статистики по индексации

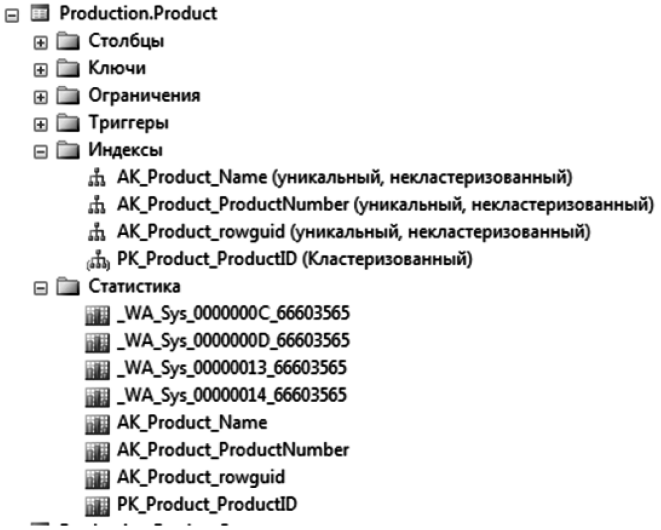

Рис. 4. Описание структуры таблицы Production.Product

По результатам общей оценки определяются пути для оптимизации, в том числе перестройка индексов, реорганизация или работы непосредственно с запросами путем формирования дополнительных блоков для отбора в виде созданных архивных таблиц на основе запросов и исключения слабо используемых данных в виде их архивирования

Теоретическая значимость состоит в развитии технологий повышения производительности базы данных, а также развитии уже существующих методов повышения производительности баз данных. Практическая значимость заключается в выработке реальных предложений по оптимизации производительности тестовой базы данных с использованием идентификации узких мест и проблемных запросов, а также представлены варианты инструментов проведения мониторинга и тестирования производительности базы данных.

Заключение

В результате проведенного исследования проблем оптимизации запросов в реляционной базе данных выявлены разнообразные пути решения. Основой задачей в этом случае является определение проблемных запросов и проведение постоянного мониторинга производительности базы данных. Определено, что все известные технологии оптимизации дают разнообразные результаты, связанные со структурой построения проблемных запросов, так и с частотой обращения к ним пользователей или верхнего уровня приложений.

Поэтому для повышения производительности базы данных необходимо обеспечить не только оптимальную структур запросов при помощи индексации или хранимых процедур, а оценить полученные изменения для конкретной информационной системы, основной которой является оптимизируемая база данных.

ЛИТЕРАТУРА

1. Вачкова С.Н., Каган Э.М., Козин С.В. Большие данные для педагогических исследований: возможности, проблемы, ограничения // Вестник Православного Свято-Тихоновского гуманитарного университета. Серия 4: Педагогика. Психология. 2021. № 63. С. 28–39.
2. Зайнуллин С.Б., Шишова Ю.А., Чжан Яци, Пин Фэн, Розмари Нвачукву Чидимма Проблематика интернет-торговли на российском и евразийском рынках // Евразийский Союз: вопросы международных отношений. 2024. Т. 13. № 1 (54). С. 69–82.
3. Лавриненко Д.О., Сорочан В.В. Методика повышения уровня производительности запросов к базам данных // Вестник Калужского университета. 2021. № 1 (50). С. 94–95.
4. Лемешев К.А. Техники оптимизации запросов в базах данных / // В сборнике: Цифровые, компьютерные и информационные технологии в науке и образовании. Сборник статей Межрегиональной научно-практической конференции с международным участием. Брянск, 2023. С. 60–64.
5. Лягушева М.А., Гринченко Н.Н. Анализ производительности SELECT-запросов SQL // В сборнике: Методы и средства обработки и хранения информации. Межвузовский сборник научных трудов. Рязань, 2021. С. 157–159.
6. Боровской И.Г., Харченко И.К. Модификация иерархических битовых индексов для повышения производительности систем управления базами данных // Доклады Томского государственного университета систем управления и радиоэлектроники. 2020. Т. 23. № 2. С. 65–72.
7. Mukkamala, R., Purna Chandra Rao, V. (2020). Approaches for Efficient Query Optimization Using Semantic Web Technologies. In: Saini, H., Sayal, R., Buyya, R., Aliser, G. (Eds) Innovations in Computer Science and Engineering. Lecture Notes in Networks and Systems, vol 103. Springer, Singapore. https://doi.org/10.1007/978-981-15-2043-3_47.
8. Абсатаров Р.Н. Хранимые процедуры как один из способов повышения производительности информационной системы // Молодой ученый. 2020. № 44 (334). С. 4–5.
9. Azgomi, H., Sohrabi, M.K. A novel coral reefs optimization algorithm for materialized view selection in data warehouse environments. Appl Intell 49, 3965–3989 (2019). https://doi.org/10.1007/s10489-019-01481-w.
10. Малыгина Э.Э. Эффективность хранимых процедур // В сборнике: XXVI Туполевские чтения (школа молодых ученых). Материалы Международной молодёжной научной конференции. Сборник докладов. Казань, 2023. С. 2447–2453.

Журнал «Современная наука: актуальные проблемы теории и практики»