

В этом разделе мы изучим концепции более продвинутых вещей в реляционных базах данных

Оптимизация SQL запросов

Оптимизация SQL запросов в реляционных базах данных, включая PostgreSQL, это стандартная практика, особенно в компаниях с большим количеством данных.

Но прежде чем показать вам стандартный алгоритм оптимизации, давайте напомним что СУБД - это довольно сложные системы, обеспечивающие эффективное управление данными. **За что отвечают СУБД:**

1. Хранение данных

- Физическое хранение: СУБД определяет, как данные физически хранятся на диске. Это может включать структурирование данных в таблицы, индексы, и другие структуры хранения.
- Буферизация и кэширование: СУБД используют буферы и кэши для оптимизации доступа к данным, снижая количество обращений к физическим носителям.

2. Транзакции и согласованность данных

- Механизмы транзакций: СУБД управляют группировкой операций в транзакции для обеспечения атомарности, согласованности, изолированности и долговечности (ACID).
- Уровни изоляции: Различные уровни изоляции транзакций позволяют сбалансировать между строгостью контроля за согласованностью данных и производительностью.

3. Обработка запросов и оптимизация

- Парсинг и оптимизация запросов: СУБД анализируют и оптимизируют SQL-запросы, составляя эффективные планы их выполнения.
- Выборка и обработка данных: Выполнение запросов включает выборку, сортировку, соединение и агрегирование данных согласно плану запроса.

4. Индексация

- Управление индексами: СУБД предоставляют механизмы индексации для ускорения доступа к данным. Индексы могут быть созданы на одном или нескольких полях таблицы.
- Поддержание индексов: СУБД поддерживает актуальность индексов при операциях вставки, удаления и обновления.

5. Безопасность и доступ

- Контроль доступа: СУБД управляют доступом к данным, используя системы учетных записей, роли и права доступа.
- Аудит и мониторинг: Функции аудита позволяют отслеживать и записывать операции с данными для обеспечения безопасности и соответствия нормативным требованиям.

6. Резервное копирование и восстановление

- Стратегии бэкапа: СУБД предоставляют инструменты для резервного копирования данных, позволяя восстанавливать данные после сбоев или потерь.
- Точки восстановления: СУБД позволяют создавать точки восстановления, чтобы можно было откатить систему к определенному состоянию.

7. Масштабируемость и производительность

- Горизонтальное и вертикальное масштабирование: СУБД поддерживают масштабирование для обработки увеличивающегося объема данных и запросов.
- Оптимизация производительности: СУБД предоставляют инструменты и методы для мониторинга и улучшения производительности системы.

Как же СУБД занимается выполнением SQL запроса?

Выполнение SQL запросов в PostgreSQL, как и в большинстве реляционных СУБД, является сложным процессом, включающим несколько этапов. Эти этапы обеспечивают эффективную обработку и оптимизацию запросов. Вот как обычно происходит этот процесс:

1. Парсинг и синтаксический анализ

Когда запрос поступает в PostgreSQL, он сначала проходит этап парсинга. На этом этапе происходит:

- Синтаксический анализ: Проверяется, соответствует ли запрос синтаксису SQL. Если в запросе есть синтаксические ошибки, они будут обнаружены здесь.
- Разбор запроса: Запрос преобразуется во внутреннее представление, известное как дерево разбора.

2. Анализ и перезапись

- Анализ: На этом шаге PostgreSQL проверяет, существуют ли все указанные в запросе таблицы, столбцы и функции, и есть ли у пользователя соответствующие права доступа.
- Перезапись запросов: Некоторые запросы могут быть переписаны для оптимизации, например, при использовании представлений (views) или правил (rules).

3. Планирование и оптимизация

- Генерация плана выполнения: Оптимизатор запросов (query planner) рассматривает различные способы выполнения запроса и выбирает наиболее эффективный план. План включает методы доступа к данным (например, сканирование таблицы или использование индекса) и методы соединения таблиц.
- Оценка стоимости: Каждый потенциальный план оценивается по стоимости, основываясь на различных факторах, таких как количество операций ввода-вывода, CPU и размер предполагаемых данных.

4. Выполнение

- Выполнение плана: Выбранный план выполнения передается исполнителю (executor), который выполняет запрос, обращаясь к данным в базе.
- Обработка данных: Данные извлекаются, объединяются, фильтруются, агрегируются и т.д., в соответствии с запросом.
- Возвращение результата: После обработки данных результаты возвращаются пользователю.

5. Кэширование

- Кэш планов выполнения: PostgreSQL может кэшировать планы выполнения запросов, чтобы ускорить выполнение аналогичных запросов в будущем.
- Буферный кэш: PostgreSQL также использует буферный кэш для хранения данных в памяти, что позволяет ускорить доступ к часто используемым данным.

Отлично, теперь вы понимаете, что выполнение запроса для СУБД - это долгий и сложный процесс. Но даже в этом случае именно вы ответственны за оптимизацию запроса, пока что СУБД не умеет сделать всё без вас.

Познакомимся с понятием "стоимость запроса".

Понятие "стоимости запроса" в контексте реляционных баз данных, включая PostgreSQL, относится к оценке ресурсов, необходимых для выполнения данного запроса. Стоимость запроса - это метрика, используемая СУБД для оценки количества "работы", необходимой для выполнения запроса. Эта стоимость выражается в условных единицах и обычно учитывает такие факторы, как время CPU и дисковые операции.

- Составляющие стоимости:
 - Время CPU: Сколько времени процессора потребуется на обработку запроса.
 - Операции ввода-вывода: Сколько дисковых операций нужно для чтения данных из памяти или с диска.

Примеры:

1. Простой запрос:

- Запрос: `SELECT * FROM employees WHERE department = 'Sales';`
- Если в таблице `employees` нет индекса по столбцу `department`, СУБД должна будет выполнить полное сканирование таблицы, чтобы найти все строки, соответствующие условию.
- Стоимость: Высокая, особенно если таблица большая.

2. Запрос с соединением (JOIN):

- Запрос: `SELECT * FROM orders JOIN customers ON orders.customer_id = customers.id;`
- Без индексов на `customer_id`, СУБД выполнит полное сканирование обеих таблиц и попытается сопоставить строки на основе условия соединения.
- Стоимость: Очень высокая, особенно если обе таблицы большие.

3. Запрос с агрегацией:

- Запрос: `SELECT COUNT(*), department FROM employees GROUP BY department;`
- СУБД должна просмотреть все строки в таблице `employees` для вычисления количества сотрудников в каждом отделе.
- Стоимость: Высокая, зависит от общего числа строк в таблице.

Как можно уменьшить стоимость запроса?

1. Ограничение выборки:

- Использование `LIMIT` для уменьшения количества обрабатываемых строк.
- Пример: `SELECT * FROM employees LIMIT 100;` будет иметь меньшую стоимость, чем запрос без `LIMIT`.

2. Уменьшение обрабатываемых данных:

- Выбор только необходимых столбцов вместо `*`.
- Пример: `SELECT id, name FROM employees;` эффективнее, чем `SELECT * FROM employees;`.

3. Добавление индексов