IT CAREER HUB

Урок 11.1. Оптимизация запросов. Индексация

Оптимизация запросов

 **Важно!**
База данных с доступом на запись:
hostname: ich-edit.edu.itcareerhub.de
MYSQL_USER: ich1
MYSQL_PASSWORD: ich1_password_ilovedbs

Оптимизация запросов в MySQL направлена на повышение скорости и эффективности работы базы данных.

Когда система управления базами данных (СУБД) выполняет запрос, она использует ресурсы, такие как процессорное время и память.

Цель оптимизации заключается в том, чтобы минимизировать количество используемых ресурсов и уменьшить время выполнения запроса. Одним из ключевых инструментов оптимизации является **индексация**.

Что такое индексация?

Индексация — это способ организации данных, который ускоряет выполнение запросов.

Индекс — это структура данных (чаще всего дерево), которая помогает быстрее находить нужные записи в таблице без необходимости полного сканирования всей таблицы.

IT CAREER HUB

Типы индексов в MySQL

- **PRIMARY KEY**: Индекс, который автоматически создается для поля, объявленного как первичный ключ. Этот индекс гарантирует уникальность значений и быстрый поиск по ключу.

Пример

```python
CREATE TABLE employees (

    id INT PRIMARY KEY,
    name VARCHAR(100),
    age INT
);
```

- **UNIQUE INDEX**: Индекс, который гарантирует уникальность значений в одном или нескольких столбцах.

Пример

```
CREATE UNIQUE INDEX idx_employee_email ON employees (email);
```

- **INDEX (или обычный индекс)**: Используется для ускорения поиска по одному или нескольким полям. Индекс не накладывает ограничений на уникальность значений.

Пример

```
CREATE INDEX idx_employee_name ON employees (name);
```

- **FULLTEXT INDEX**: Этот тип индекса используется для полнотекстового поиска по текстовым полям.

Пример

```
Unset
CREATE FULLTEXT INDEX idx_article_text ON articles (content);
```

- **COMPOSITE INDEX (составной индекс)**: Индекс, созданный на несколько столбцов одновременно. Он полезен для запросов, которые используют сразу несколько условий по разным полям.

Пример

```
Unset
CREATE INDEX idx_employee_name_age ON employees (name, age);
```

IT CAREER HUB

Как индексы ускоряют запросы

1. Индексы помогают MySQL быстрее находить строки, соответствующие условиям запроса, уменьшая количество данных, которые нужно просмотреть. Вместо полного сканирования таблицы СУБД может использовать индекс, чтобы напрямую перейти к записям, соответствующим критерию запроса.

Пример без индекса

```
Unset
SELECT * FROM employees WHERE age = 30;
```

MySQL будет просматривать каждую строку в таблице, чтобы найти все записи, где `age = 30`.

Пример с индексом

```
Unset
CREATE INDEX idx_employee_age ON employees (age);


SELECT * FROM employees WHERE age = 30;
```

С индексом MySQL будет использовать структуру индекса, чтобы быстро найти строки, соответствующие возрасту `30`.

Пример работы с индексами

1. Предположим, у нас есть таблица сотрудников:

```
Unset
CREATE TABLE employees (
    id INT PRIMARY KEY AUTO_INCREMENT,
    name VARCHAR(100),
    age INT,
    salary INT,
    department_id INT
);
```

2. Если в таблице много записей, и вы часто выполняете запросы на поиск сотрудников по полю age, имеет смысл создать индекс на это поле:

```
Unset
CREATE INDEX idx_age ON employees (age);
```

3. Теперь при выполнении следующего запроса MySQL не будет сканировать всю таблицу, а использовать индекс для быстрого доступа к нужным строкам:

```
Unset
SELECT * FROM employees WHERE age = 30;
```

Минусы индексации

Хотя индексация значительно улучшает производительность запросов, есть и некоторые **недостатки**:

- **Использование дискового пространства**: Индексы занимают дополнительное место на диске. Чем больше индексов, тем больше места потребуется.
- **Замедление операций вставки, обновления и удаления**: Каждый раз, когда данные в таблице изменяются (вставляются, удаляются или обновляются), MySQL должен также обновить соответствующие индексы. Это может замедлить выполнение операций изменения данных.
- **Неэффективность для очень маленьких таблиц**: Для таблиц с небольшим количеством строк индексация может не дать существенного прироста производительности, так как полный скан таблицы будет выполняться быстро и без индекса.

Составные индексы

Если запросы часто содержат несколько условий на разные поля, можно создать составной индекс для этих полей.

Пример с составным индексом

```
Unset
CREATE INDEX idx_department_age_salary ON employees (department_id, age, salary);
```

В этом примере запросы, использующие одновременно столбцы `department_id`, `age` и `salary`, будут работать быстрее.

Не индексируйте все подряд

Индексация всех столбцов подряд может замедлить операции вставки и обновления. Создавайте индексы только на те поля, которые действительно необходимы для ускорения поиска.

Использование индексов для сортировки и группировки

Индексы могут помочь ускорить запросы с `ORDER BY` и `GROUP BY`, если индекс включает столбцы, по которым производится сортировка или группировка.

Пример

```
Unset
CREATE INDEX idx_name_age ON employees (name, age);
```

Запрос:

```
Unset
SELECT * FROM employees ORDER BY name, age;
```

В этом случае MySQL может использовать составной индекс для сортировки записей.

IT CAREER HUB

Анализ запросов с помощью EXPLAIN

EXPLAIN— это команда, которая позволяет увидеть, как MySQL будет выполнять запрос.

Она помогает понять, использует ли MySQL индекс для данного запроса, и если нет — можно проанализировать, как улучшить запрос или добавить подходящий индекс.

Пример использования EXPLAIN:

```
Unset
EXPLAIN SELECT * FROM employees WHERE age = 30;
```

Вывод команды `EXPLAIN` покажет:

- Какой тип сканирования будет использоваться (например, полный скан таблицы или использование индекса).
- Оценку количества строк, которые должны быть прочитаны.
- Какой индекс будет использоваться (если применимо).

IT CAREER HUB

Практическая работа

1. Создайте таблицу students с такими столбцами: `id (INT)`, `name (VARCHAR)`, `age (INT)`, `grade (DECIMAL)`.

2. Заполните таблицу несколькими строками.

3. Создайте индекс на столбец age, чтобы ускорить поиск по возрасту.

4. Напишите запрос, который выбирает всех студентов определенного возраста.

5. Просмотрите план выполнения запроса с помощью команды `EXPLAIN`.

Решение

1. Создайте таблицу `students` с такими столбцами: `id (INT), name (VARCHAR), age (INT), grade (DECIMAL).`

```
Unset
CREATE TABLE students (
    id INT PRIMARY KEY AUTO_INCREMENT,
    name VARCHAR(100),
    age INT,
    grade DECIMAL(4, 2)
);
```

2. Заполните таблицу несколькими строками.

3. Создайте индекс на столбец age, чтобы ускорить поиск по возрасту.

```
Unset
CREATE INDEX idx_age ON students(age);
```

4. Напишите запрос, который выбирает всех студентов определенного возраста.

```
Unset
SELECT * FROM students WHERE age = 20;
```

5. Просмотрите план выполнения запроса с помощью команды `EXPLAIN`.

```
Unset
EXPLAIN SELECT * FROM students WHERE age = 20;
```