MySQL

MySQL là hệ quản trị cơ sở dữ liệu tự do nguồn mở phổ biến nhất thế giới và được các nhà phát triển rất ưa chuộng trong quá trình phát triển ứng dụng. Vì MySQL là cơ sở dữ liệu tốc độ cao, ổn định và dễ sử dụng, có tính khả chuyển, hoạt động trên nhiều hệ điều hành cung cấp một hệ thống lớn các hàm tiện ích rất mạnh.

- MySQL - Temporary Table
- MySQL - Clone Table

MySQL Tutorial

Kiến thức cơ bản về MySQL

MySQL overview

MySQL là gì?

MySQL là một hệ thống quản trị cơ sở dữ liệu mã nguồn mở (Relational Database Management System, viết tắt là RDBMS) hoạt động theo mô hình client-server. RDBMS là một phần mềm hay dịch vụ dùng để tạo và quản lý các cơ sở dữ liệu (Database) theo hình thức quản lý các mối liên hệ giữa chúng.

Công ty Thuy Điển MySQL AB phát triển MySQL vào năm 1994. Công ty công nghệ Mỹ Sun Microsystem sau đó giữ quyền sở hữu MySQL sau khi mua lại MySQL vào năm 2008. Năm 2010, gã khổng lồ Oracle mua Sun Microsystems và MySQL thuộc quyền sở hữu của Oracle từ đó.

Ưu điểm của MySQL

MySQL không phải là hệ quản lý cơ sở dữ liệu (RDBMS) duy nhất trên thị trường, nhưng nó đích thực phổ biến nhất và chỉ xếp sau Oracle Database khi xét đến những thông số chính như số lượng tìm kiếm, profile người dùng trên LinkedIn, và lượng thảo luận trên các diễn đàn internet. Lý do chính vì sao rất nhiều ông lớn công nghệ phụ thuộc vào MySQL là gì? Các lý do quan trọng như sau:

Linh hoạt và dễ dùng

Bạn có thể sửa source code để đáp ứng nhu cầu của bạn mà không phải thanh toán têm bất kỳ chi phí nào. Quá trình cài đặt cũng rất đơn giản và thường không quá 30 phút.

Hiệu năng cao

Nhiều server clusters sử dụng MySQL. Bất kể bạn lưu trữ dữ liệu lớn của các trang thương mại điện tử hoặc những hoạt động kinh doanh nặng nề liên quan đến công nghệ thông tin, MySQL cũng có thể đáp ứng được với tốc độ cao, mượt mà.

Tiêu chuẩn trong ngành

Ngành công nghệ và dữ liệu đã sử dụng MySQL nhiều năm, vì vậy nó là một kỹ năng căn bản một chuyên gia lập trình. Người dùng MySQL cũng có thể triển khai dự án nhanh và thuê các chuyên gia dữ liệu với mức phí nếu họ cần.

An toàn

An toàn dữ liệu luôn là vấn đề quan trọng nhất khi chọn phần mềm RDBMS. Với hệ thống phân quyền truy cập và quản lý tài khoản, MySQL đặt tiêu chuẩn bảo mật rất cao. Mã hóa thông tin đăng nhập và chứng thực từ host đều khả dụng.

MySQL Create database

Create Database sử dụng mysql client

- Để thực hiện create MySQL database, thực hiện command sau:

```
[root@host]# mysql -u username -p
CREATE DATABASE dbname;
```

- Cấp quyền truy cập database cho user:

```
[root@host]# mysql -u username -p
GRANT ALL PRIVILEGES ON dbname.* TO 'username'@'localhost'  IDENTIFIED BY 'password';
FLUSH PRIVILEGES;
```

Create Database sử dụng mysqladmin

Bạn cần quyền root để create hoặc delete một MySQL database.

Ví dụ:

```
[root@host]# mysqladmin -u root -p create LAPTRINHVN
Enter password: ******
```

MySQL Drop database

Để thực hiện drop database, thực hiện lệnh sau:

```
[root@host]# mysql -u root -p
mysql> DROP DATABASE database_name;
```

Việc thực hiện DROP sẽ gây ra việc xóa toàn bộ dữ liệu của database, do đó, cần thực hiện một cách cẩn thận và chính xác do không có khả năng khôi phục lại dữ liệu nếu không có bản backup của database.

Ví dụ:

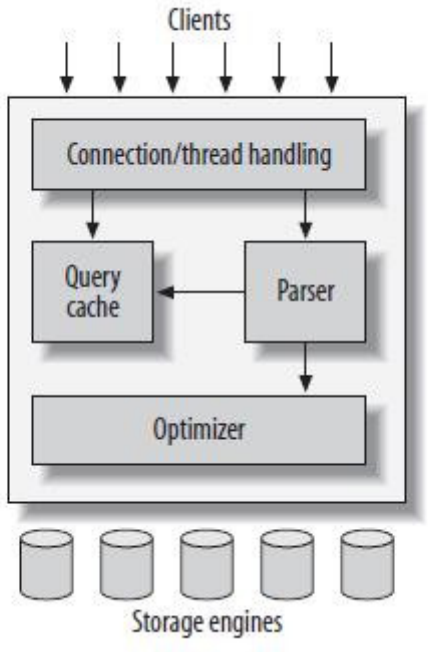
```
>mysql -u root -p
Enter password: ********
mysql> SHOW DATABASES;
+--------------------+
| Database           |
+--------------------+
| classicmodels      |
| information_schema |
| mysql              |
| performance_schema |
| sys                |
| testdb             |
| testdb2            |
+--------------------+
7 rows in set (0.00 sec)

mysql> DROP DATABASE testdb;
Query OK, 0 rows affected (0.03 sec)
```

MySQL Storage Engines

Giới thiệu

Kiến trúc logic của MySQL nhìn tổng quan có thể được mô tả như hình dưới đây:



Storage Engine thực chất là cách MySQL lưu trữ dữ liệu trên đĩa cứng. MySQL lưu mỗi database như là một thư mục con nằm dưới thư mục data. Khi một table được tạo ra, MySQL sẽ lưu định nghĩa bảng ở file đuôi .frm và tên trùng với tên của bảng được tạo. Việc quản lý định nghĩa bảng là nhiệm vụ của MySQL server, dù rằng mỗi storage engine sẽ lưu trữ và đánh chỉ mục (index) dữ liệu khác nhau.

Tiêu chí lựa chọn engine:

- Transactions: Nếu ứng dụng yêu cầu transactions, InnoDB là lựa chọn duy nhất. Nếu không yêu cầu transactions, MyISAM là lựa chọn tốt.
- Concurrency: Nếu yêu cầu chịu tải cao và không cần thiết transactions, MyISAM là lựa chọn số 1.
- Sao lưu: Các engine đều phần nào hỗ trợ sao lưu. Ngoài ra ta cần hỗ trợ sao lưu trên cả quan điểm thiết kế hệ thống. Ví dụ: bạn thiết kế database server gồm master và slave, master yêu cần transaction nên dùng innodb, slave cần sao lưu và đọc nên có thể dùng MyISAM. Cơ chế đồng bộ master-slave sẽ giúp bạn quản lý sự khác nhau giữa các engine nhưng đảm bảo tính sao lưu. Tiêu chí này có trọng số nhỏ.
- Phục hồi sau crash: MyISAM có khả năng phục hồi sau crash kém hơn InnoDB.

- Tính năng theo yêu cầu hệ thống: Nếu yêu cầu là logging, MyISAM hoặc Archive là lựa chọn hợp lý. Nếu cần lưu trực tiếp CSV, CSV engine là lựa chọn đáng cân nhắc. Nếu ứng dụng không thay đổi dữ liệu mấy (ví dụ cơ sở dữ liệu sách), MyISAM và tính năng nén là lựa chọn phù hợp.

1. MyISAM engine

Đặc điểm:

- full-text indexing
- compression.
- spatial functions (GIS)
- Không hỗ trợ transactions
- Không hỗ trợ row-level lock.

Lưu trữ:

MyISAM lưu mỗi bảng dữ liệu trên 2 file: .MYD cho dữ liệu và .MYI cho chỉ mục. Row có 2 loại: dynamic và static (tuỳ thuộc bạn có dữ liệu thay đổi độ dài hay không). Số lượng row tối đa có thể lưu trữ bị giới hạn bởi hệ điều hành, dung lượng đĩa cứng. MyISAM mặc định sử dụng con trỏ độ dài 6 bytes để trỏ tới bản ghi dữ liệu, do vậy giới hạn kích thước dữ liệu xuống 256TB.

Tính năng:

- MyISAM lock toàn bộ table. User (MySQL server) chiếm shared-lock khi đọc và chiếm exclusive-lock khi ghi. Tuy vậy, việc đọc ghi có thể diễn ra đồng thời!
- MyISAM có khả năng tự sửa chữa và phục hồi dữ liệu sau khi hệ thống crashed.
- Dùng command check table / repair table để kiểm tra lỗi và phục hồi sau khi bị lỗi.
- MyISAM có thể đánh chỉ mục full-text, hỗ trợ tìm kiếm full-text.
- MyISAM không ghi dữ liệu ngay vào ổ đĩa cứng, mà ghi vào 1 buffer trên memory (và chỉ ghi vào đĩa cứng sau 1 khoảng thời gian), do đó tăng tốc độ ghi. Tuy vậy, sau khi server crash, ta cần phải phục hồi dữ liệu bị hư hỏng bằng myisamchk.
- MyISAM hỗ trợ nén dữ liệu, hỗ trợ tăng tốc độ đọc dữ liệu. Mặc dù vậy dữ liệu sau khi nén không thể cập nhật được.

2. InnoDB engine

Đặc điểm:

- Là engine phức tạp nhất trong các engine của MySQL
- Hỗ trợ transactions
- Hỗ trợ phục hồi, sửa chữa tốt

Lưu trữ:

InnoDB lưu dữ liệu trên 1 file (thuật ngữ gọi là tablespace).

Tính năng:

- InnoDB hỗ trợ MVCC (Multiversion Concurrency Control) để cải thiện việc truy cập đồng thời và hỗ trợ chiến thuật next-key locking.
- InnoDB được xây dựng dựa trên clustered index, do đó việc tìm kiếm theo primary key có hiệu năng rất cao. InnoDB không hỗ trợ sắp xếp index do vậy việc thay đổi cấu trúc bảng sẽ dẫn tới toàn bộ dữ liệu phải được đánh chỉ mục từ đầu (CHẬM với những bảng lớn).

3. Memory engine

Đặc điểm:

- Còn được gọi là HEAP tables.

Lưu trữ:

Tất cả dữ liệu đều nằm trên memory.

Tính năng:

- Sau khi server restart, cấu trúc bảng được bảo toàn, dữ liệu bị mất hết.
- Memory engine sử dụng HASH index nên rất nhanh cho query lookup.
- Memory engine dùng table-level locking do vậy tính concurrency không cao.

4. Archive engine

Đặc điểm:

- Chỉ hỗ trợ Insert và Select.
- Không đánh chỉ mục
- Dữ liệu được buffer và nén bằng zlib nên tốn ít I/O, tốc độ ghi do đó cao.

Tính năng:

- Tốc độ ghi cao, phù hợp cho ứng dụng log.

5. CSV engine

Đặc điểm:

- Coi file CSV như là 1 table.
- Không hỗ trợ đánh chỉ mục

Tính năng:

- Nếu bài toán là trích xuất thông tin từ file CSV và ghi vào cơ sở dữ liệu, đồng thời cần kết quả CSV ngay từ DB, engine này có vẻ thích hợp.

6. Falcon engine

Đặc điểm:

- Được thiết kế cho phần cứng hiện đại: server 64 bit, bộ nhớ “thênh thang”
- Vẫn còn khá mới, chưa có nhiều usercase

7. Maria engine

Đặc điểm:

- Được thiết kế bởi với mục đích thay thế MyISAM
- Hỗ trợ transactions theo lựa chọn
- Khôi phục lỗi
- Row-level locking và MVCC
- Hỗ trợ BLOB tốt hơn.

8. Merge engine

Đặc điểm:

- Merge table là một bảng ảo kết hợp nhiều bảng MyISAM có cấu trúc tương tự như một bảng. Công cụ lưu trữ MERGE còn được gọi là công cụ MRG_MyISAM. Bảng MERGE không có index, thay vào đó, nó sử dụng các index của các bảng thành phần.
- Sử dụng Merge table, bạn có thể tăng tốc độ thực hiện các lệnh join giữa các bảng. MySQL chỉ cho phép bạn thực hiện các lệnh SELECT, DELETE, UPDATE và INSERT trên Merge table. Khi bạn thực hiện DROP TABLE trên Merge table, chỉ Merge table bị xóa, các bảng thành phần không bị ảnh hưởng

9. Federated engine

Công cụ lưu trữ Federated cho phép bạn quản lý dữ liệu từ máy chủ MySQL từ xa (remote) mà không cần sử dụng công nghệ cluster hoặc replication. Bảng liên kết cục bộ không lưu trữ dữ liệu. Khi bạn truy vấn dữ liệu từ một bảng được liên kết cục bộ (local), dữ liệu sẽ được kéo tự động từ các bảng được liên kết từ xa (remote).

MySQL - Connection

Để kết nối tới cơ sở dữ liệu MySQL bằng MySQL Binary, bạn cần làm theo các bước sau:

1. Tải xuống và cài đặt MySQL Binary từ trang web chính thức của MySQL.
2. Mở Command Prompt hoặc Terminal và nhập lệnh sau:

   ```
   mysql -h <hostname> -u <username> -p
   ```

   Trong đó:
   - `<hostname>` là địa chỉ hoặc tên miền của máy chủ MySQL.
   - `<username>` là tên người dùng của tài khoản MySQL.

   `mysql` là lệnh để kết nối tới cơ sở dữ liệu MySQL.
3. Nhập mật khẩu cho tài khoản MySQL.
4. Sau khi đăng nhập thành công, bạn có thể sử dụng các câu lệnh SQL để truy vấn cơ sở dữ liệu.

MySQL - Create Database

Để tạo một cơ sở dữ liệu trong MySQL, bạn có thể sử dụng câu lệnh SQL sau:

```
CREATE DATABASE tên_cơ_sở_dữ_liệu;
```

Thay thế `tên_cơ_sở_dữ_liệu` bằng tên mà bạn muốn đặt cho cơ sở dữ liệu của mình. Câu lệnh này sẽ tạo ra một cơ sở dữ liệu mới với tên được chỉ định.

Sau đó, bạn có thể sử dụng cơ sở dữ liệu này để tạo bảng và lưu trữ dữ liệu bằng các câu lệnh SQL như `CREATE TABLE`, `INSERT INTO` và `SELECT`.

Hãy nhớ sử dụng cú pháp SQL phù hợp và đảm bảo rằng tên cơ sở dữ liệu của bạn tuân theo bất kỳ quy ước hoặc hạn chế đặt tên nào được áp dụng.

MySQL - Drop Database

Để xóa một cơ sở dữ liệu trong MySQL, bạn có thể sử dụng câu lệnh SQL sau:

```
DROP DATABASE tên_cơ_sở_dữ_liệu;
```

Thay thế `tên_cơ_sở_dữ_liệu` bằng tên của cơ sở dữ liệu mà bạn muốn xóa. Câu lệnh này sẽ xóa cơ sở dữ liệu đó khỏi hệ thống.

Hãy nhớ rằng việc xóa cơ sở dữ liệu sẽ xóa tất cả các bảng và dữ liệu trong đó. Hãy đảm bảo rằng bạn đã sao lưu dữ liệu của mình trước khi xóa bất kỳ cơ sở dữ liệu nào.

MySQL - Select Database

Để chọn một cơ sở dữ liệu trong MySQL bằng tiếng Việt, bạn có thể sử dụng câu lệnh SQL sau:

```
USE tên_cơ_sở_dữ_liệu;
```

Thay thế `tên_cơ_sở_dữ_liệu` bằng tên của cơ sở dữ liệu mà bạn muốn chọn. Câu lệnh này sẽ chọn cơ sở dữ liệu đó để tất cả các truy vấn tiếp theo được thực hiện đối với cơ sở dữ liệu này.

Hãy nhớ rằng bạn cần phải kết nối đến máy chủ MySQL và đăng nhập vào tài khoản của mình trước khi thực hiện câu lệnh này. Bạn cũng cần đảm bảo rằng cơ sở dữ liệu đã được tạo trước đó và bạn có quyền truy cập vào nó.

MySQL - Data Types

Trong MySQL có các kiểu dữ liệu (Data Type) cơ bản sau:

Kiểu dữ liệu	Mô tả	Ví dụ
INT	Kiểu số nguyên, có thể lưu trữ các giá trị từ -2147483648 đến 2147483647.	`CREATE TABLE example_table (id INT, age INT);`
VARCHAR	Kiểu chuỗi ký tự có độ dài tối đa được xác định trước.	`CREATE TABLE example_table (name VARCHAR(50), email VARCHAR(100));`
TEXT	Kiểu chuỗi ký tự có độ dài không giới hạn.	`CREATE TABLE example_table (id INT, description TEXT);`
DATE	Kiểu ngày tháng, lưu trữ các giá trị ngày tháng theo định dạng YYYY-MM-DD.	`CREATE TABLE example_table (id INT, date_created DATE);`
TIME	Kiểu thời gian, lưu trữ các giá trị thời gian theo định dạng hh:mm:ss.	`CREATE TABLE example_table (id INT, time_created TIME);`
DATETIME	Kết hợp kiểu ngày tháng và thời gian, lưu trữ các giá trị theo định dạng YYYY-MM-DD hh:mm:ss.	`CREATE TABLE example_table (id INT, datetime_created DATETIME);`
FLOAT	Kiểu số thực có độ chính xác đơn giản.	`CREATE TABLE example_table (id INT, float_number FLOAT(5,2));`
DOUBLE	Kiểu số thực có độ chính xác gấp đôi so với kiểu FLOAT.	`CREATE TABLE example_table (id INT, double_number DOUBLE(5,2));`
BOOLEAN	Kiểu boolean, lưu trữ giá trị true hoặc false.	`CREATE TABLE example_table (id INT, is_active BOOLEAN);`

MySQL - Create Table

Để tạo bảng trong MySQL, bạn có thể sử dụng lệnh `CREATE TABLE` theo sau là tên bảng và danh sách các cột và kiểu dữ liệu của chúng.

Dưới đây là cú pháp ví dụ:

```
CREATE TABLE tên_b□ng (
    tên_cột_1 ki□u_dữ_liệu ràng_buộc,
    tên_cột_2 ki□u_dữ_liệu ràng_buộc,
    ...
);
```

Ví dụ, để tạo một bảng có tên là `users` với các cột `id`, `name`, `email` và `age`, bạn có thể sử dụng lệnh SQL sau:

```
CREATE TABLE users (
    id INT PRIMARY KEY,
    name VARCHAR(50) NOT NULL,
    email VARCHAR(100) UNIQUE,
    age INT
);
```

Trong ví dụ này, cột `id` được chỉ định là khóa chính bằng cách sử dụng ràng buộc `PRIMARY KEY`. Cột `name` được chỉ định là một trường bắt buộc bằng cách sử dụng ràng buộc `NOT NULL`, và cột `email`

được chỉ định là một trường duy nhất bằng cách sử dụng ràng buộc `UNIQUE`.

Bạn có thể thêm các cột khác vào bảng bằng cách bao gồm chúng trong danh sách các cột được phân tách bằng dấu phẩy. Bạn cũng có thể chỉ định các ràng buộc bổ sung cho mỗi cột nếu cần.

Sau khi đã tạo một bảng, bạn có thể sử dụng các lệnh SQL như `INSERT INTO`, `UPDATE` và `SELECT` để thêm, sửa đổi và truy xuất dữ liệu từ bảng.

Để biết thêm thông tin về cách tạo và quản lý bảng trong MySQL, bạn có thể tham khảo tài liệu chính thức của MySQL hoặc các tài nguyên trực tuyến khác.

MySQL - Drop Table

Để xóa một bảng trong MySQL, bạn có thể sử dụng câu lệnh SQL sau:

```
DROP TABLE tên_b□ng;
```

Thay thế `tên_b□ng` bằng tên của bảng mà bạn muốn xóa. Câu lệnh này sẽ xóa bảng đó khỏi cơ sở dữ liệu.

Ví dụ, để xóa bảng `users`, bạn có thể sử dụng câu lệnh SQL sau:

```
DROP TABLE users;
```

Hãy nhớ rằng việc xóa bảng sẽ xóa tất cả các dữ liệu trong bảng đó. Hãy đảm bảo rằng bạn đã sao lưu dữ liệu của mình trước khi xóa bất kỳ bảng nào.

MySQL - Insert Query

Để chèn dữ liệu vào một bảng trong MySQL, bạn có thể sử dụng câu lệnh SQL `INSERT INTO`. Dưới đây là cú pháp của câu lệnh này:

```
INSERT INTO tên_bảng (cột1, cột2, cột3, ...)
VALUES (giá_trị1, giá_trị2, giá_trị3, ...);
```

Thay thế `tên_bảng` bằng tên của bảng mà bạn muốn chèn dữ liệu vào. Danh sách các cột cũng phải được chỉ định. Nếu bạn muốn chèn dữ liệu vào tất cả các cột, bạn có thể sử dụng dấu * thay vì danh sách các cột. Sau đó, bạn cần chỉ định giá trị cho mỗi cột.

Ví dụ, để chèn thông tin của một người vào bảng `users`, bạn có thể sử dụng câu lệnh SQL sau:

```
INSERT INTO users (id, name, email, age)
VALUES (1, 'John Doe', 'johndoe@example.com', 30);
```

Trong ví dụ này, giá trị cho cột `id`, `name`, `email` và `age` được chỉ định. Bạn có thể chèn dữ liệu cho nhiều bản ghi hơn bằng cách sử dụng nhiều câu lệnh `INSERT INTO`.

Để truy vấn dữ liệu từ một bảng trong MySQL, bạn có thể sử dụng câu lệnh SQL `SELECT`. Dưới đây là cú pháp của câu lệnh này:

```
SELECT cột1, cột2, cột3, ...
FROM tên_bảng
WHERE điều_kiện;
```

Thay thế `cột1`, `cột2`, `cột3`, ... bằng danh sách các cột mà bạn muốn truy vấn. Nếu bạn muốn truy vấn tất cả các cột, bạn có thể sử dụng dấu * thay vì danh sách các cột. `Tên_bảng` là tên của bảng mà bạn muốn truy vấn. `Điều_kiện` là một biểu thức điều kiện để lọc các bản ghi. Nếu bạn muốn truy vấn tất cả các bản ghi, bạn có thể bỏ qua điều kiện.

Ví dụ, để truy vấn tất cả các người dùng trong bảng `users`, bạn có thể sử dụng câu lệnh SQL sau:

```
SELECT * FROM users;
```

Để tìm kiếm các bản ghi cụ thể trong bảng `users`, bạn có thể sử dụng câu lệnh SQL sau:

```
SELECT id, name, email FROM users WHERE age > 30;
```

Trong ví dụ này, chỉ có các bản ghi với giá trị `age` lớn hơn 30 được trả về, và chỉ các cột `id`, `name` và `email` được hiển thị.

MySQL - Select Query

Để truy vấn dữ liệu từ một bảng trong MySQL, bạn có thể sử dụng câu lệnh SQL `SELECT`. Dưới đây là cú pháp của câu lệnh này:

```
SELECT cột1, cột2, cột3, ...
FROM tên_b□ng
WHERE đi□u_kiện;
```

Thay thế `cột1`, `cột2`, `cột3`, ... bằng danh sách các cột mà bạn muốn truy vấn. Nếu bạn muốn truy vấn tất cả các cột, bạn có thể sử dụng dấu * thay vì danh sách các cột. `Tên_b□ng` là tên của bảng mà bạn muốn truy vấn. `Đi□u_kiện` là một biểu thức điều kiện để lọc các bản ghi. Nếu bạn muốn truy vấn tất cả các bản ghi, bạn có thể bỏ qua điều kiện.

Ví dụ, để truy vấn tất cả các người dùng trong bảng `users`, bạn có thể sử dụng câu lệnh SQL sau:

```
SELECT * FROM users;
```

Để tìm kiếm các bản ghi cụ thể trong bảng `users`, bạn có thể sử dụng câu lệnh SQL sau:

```
SELECT id, name, email FROM users WHERE age > 30;
```

Trong ví dụ này, chỉ có các bản ghi với giá trị `age` lớn hơn 30 được trả về, và chỉ các cột `id`, `name` và `email` được hiển thị.

```
SELECT * FROM users WHERE name LIKE 'John%';
```

Trong ví dụ này, chỉ có các bản ghi với tên bắt đầu bằng "John" được trả về.

MySQL - WHERE Clause

Trong MySQL, câu lệnh `WHERE` được sử dụng để lọc dữ liệu được trả về bởi câu lệnh `SELECT` dựa trên một điều kiện cụ thể. Cú pháp sử dụng `WHERE` như sau:

```
FROM tên_bảng
WHERE điều_kiện;
```

Thay thế `cột1, cột2, ...` bằng các cột mà bạn muốn chọn từ bảng và `tên_bảng` bằng tên của bảng bạn muốn truy vấn. `Điều_kiện` là bộ lọc bạn muốn áp dụng cho dữ liệu. Ví dụ, nếu bạn muốn chọn tất cả các hàng từ một bảng với giá trị cột `age` lớn hơn 30, bạn có thể sử dụng câu lệnh SQL sau:

```
SELECT * FROM tên_bảng WHERE age > 30;
```

Điều này sẽ trả về tất cả các hàng từ bảng với giá trị trong cột `age` lớn hơn 30.

Một ví dụ khác là nếu bạn muốn chọn tất cả các hàng từ một bảng với giá trị trong cột `name` bắt đầu bằng "John", bạn có thể sử dụng câu lệnh SQL sau:

```
SELECT * FROM tên_bảng WHERE name LIKE 'John%';
```

Điều này sẽ trả về tất cả các hàng từ bảng với giá trị trong cột `name` bắt đầu bằng "John".

Bạn có thể sử dụng các toán tử so sánh khác như `=`, `<`, `<=`, `>`, và `>=`, cũng như các toán tử logic như `AND`, `OR`, và `NOT`, để tạo điều kiện phức tạp hơn.

MySQL - Update Query

Để cập nhật dữ liệu trong MySQL, bạn có thể sử dụng câu lệnh SQL `UPDATE`. Dưới đây là cú pháp của câu lệnh này:

```
UPDATE tên_bảng
SET cột1 = giá_trị1, cột2 = giá_trị2, ...
WHERE điều_kiện;
```

Thay thế `tên_bảng` bằng tên của bảng mà bạn muốn cập nhật dữ liệu. Danh sách các cột cũng phải được chỉ định, cùng với giá trị mới mà bạn muốn cập nhật. `Điều_kiện` là một biểu thức điều kiện để xác định các bản ghi cần cập nhật. Nếu bạn muốn cập nhật tất cả các bản ghi, bạn có thể bỏ qua điều kiện.

Ví dụ, để cập nhật thông tin của một người trong bảng `users`, bạn có thể sử dụng câu lệnh SQL sau:

```
UPDATE users
SET name = 'Jane Doe', email = 'janedoe@example.com', age = 35
WHERE id = 1;
```

Trong ví dụ này, giá trị mới cho các cột `name`, `email`, và `age` được chỉ định. Chỉ có bản ghi với `id` là 1 sẽ được cập nhật.

Một ví dụ khác là nếu bạn muốn tăng giá trị của một cột bằng một số lượng cụ thể, bạn có thể sử dụng câu lệnh SQL sau:

```
UPDATE users
SET age = age + 1
WHERE id = 1;
```

Điều này sẽ tăng giá trị của cột `age` lên 1 cho bản ghi với `id` là 1.

MySQL - DELETE Query

Để xóa một hàng trong MySQL, bạn có thể sử dụng câu lệnh SQL `DELETE`. Dưới đây là cú pháp của câu lệnh này:

```
DELETE FROM tên_bảng
WHERE điều_kiện;
```

Thay thế `tên_bảng` bằng tên của bảng mà bạn muốn xóa hàng khỏi đó. `Điều_kiện` là một biểu thức điều kiện để xác định hàng cần xóa. Nếu bạn muốn xóa tất cả các hàng trong bảng, bạn có thể bỏ qua điều kiện.

Ví dụ, để xóa hàng có `id` là 1 trong bảng `users`, bạn có thể sử dụng câu lệnh SQL sau:

```
DELETE FROM users
WHERE id = 1;
```

Điều này sẽ xóa hàng đầu tiên có `id` là 1 trong bảng `users`.

Một ví dụ khác là nếu bạn muốn xóa tất cả các hàng trong bảng `orders` với `status` là "cancelled", bạn có thể sử dụng câu lệnh SQL sau:

```
DELETE FROM orders
WHERE status = 'cancelled';
```

Điều này sẽ xóa tất cả các hàng trong bảng `orders` với `status` là "cancelled".

Hãy đảm bảo rằng bạn đã sao lưu dữ liệu của mình trước khi xóa bất kỳ hàng nào.

MySQL - LIKE Clause

Trong MySQL, câu lệnh `SELECT` được sử dụng để truy vấn dữ liệu từ một bảng. Bạn có thể sử dụng từ khóa `LIKE` trong câu lệnh `SELECT` để lọc các bản ghi dựa trên một chuỗi ký tự cụ thể (pattern). Cú pháp sử dụng `LIKE` như sau:

```
SELECT cột1, cột2, ...
FROM tên_bảng
WHERE cột LIKE 'pattern';
```

Thay thế `cột1`, `cột2`, ... bằng danh sách các cột mà bạn muốn truy vấn. `Tên_bảng` là tên của bảng mà bạn muốn truy vấn. `Cột` là tên của cột mà bạn muốn áp dụng pattern. `Pattern` là chuỗi ký tự mà bạn muốn sử dụng để lọc các bản ghi. Bạn có thể sử dụng ký tự `%` để biểu thị bất kỳ chuỗi ký tự nào, hoặc ký tự `_` để biểu thị một ký tự bất kỳ.

Ví dụ, để tìm kiếm các bản ghi có tên bắt đầu bằng "John" trong bảng `users`, bạn có thể sử dụng câu lệnh SQL sau:

```
SELECT * FROM users
WHERE name LIKE 'John%';
```

Điều này sẽ trả về tất cả các bản ghi trong bảng `users` có tên bắt đầu bằng "John".

Một ví dụ khác là nếu bạn muốn tìm kiếm các bản ghi có email kết thúc bằng "@example.com", bạn có thể sử dụng câu lệnh SQL sau:

```
SELECT * FROM users
WHERE email LIKE '%@example.com';
```

Điều này sẽ trả về tất cả các bản ghi trong bảng `users` có email kết thúc bằng "@example.com".

MySQL - Sorting Results

Trong MySQL, câu lệnh `SELECT` có thể được sử dụng để sắp xếp dữ liệu trả về từ một bảng. Bạn có thể sử dụng từ khóa `ORDER BY` để sắp xếp các bản ghi dựa trên một hoặc nhiều cột cụ thể. Cú pháp sử dụng `ORDER BY` như sau:

```
SELECT cột1, cột2, ...
FROM tên_bảng
ORDER BY cột ASC|DESC;
```

Thay thế `cột1`, `cột2`, ... bằng danh sách các cột mà bạn muốn truy vấn. `Tên_bảng` là tên của bảng mà bạn muốn truy vấn. `ASC` và `DESC` là từ khóa để chỉ định thứ tự sắp xếp. `ASC` là sắp xếp tăng dần và `DESC` là sắp xếp giảm dần. Nếu bạn không chỉ định thứ tự sắp xếp, mặc định là `ASC`.

Ví dụ, để sắp xếp các bản ghi trong bảng `users` theo tên theo thứ tự bảng chữ cái, bạn có thể sử dụng câu lệnh SQL sau:

```
SELECT * FROM users
ORDER BY name ASC;
```

Điều này sẽ trả về tất cả các bản ghi trong bảng `users` được sắp xếp theo tên theo thứ tự bảng chữ cái.

Một ví dụ khác là nếu bạn muốn sắp xếp các bản ghi trong bảng `orders` theo thời gian tạo đơn hàng từ mới nhất đến cũ nhất, bạn có thể sử dụng câu lệnh SQL sau:

```
SELECT * FROM orders
ORDER BY created_at DESC;
```

Điều này sẽ trả về tất cả các bản ghi trong bảng `orders` được sắp xếp theo thời gian tạo đơn hàng từ mới nhất đến cũ nhất.

MySQL - Using Join

Trong MySQL, `JOIN` được sử dụng để kết hợp các bảng khác nhau với nhau bằng cách sử dụng một hoặc nhiều cột chung. Có nhiều loại `JOIN` khác nhau, bao gồm:

- `INNER JOIN`: Trả về các bản ghi chỉ khi có các giá trị khớp trong cả hai bảng.
- `LEFT JOIN`: Trả về tất cả các bản ghi từ bảng bên trái và các bản ghi khớp từ bảng bên phải.
- `RIGHT JOIN`: Trả về tất cả các bản ghi từ bảng bên phải và các bản ghi khớp từ bảng bên trái.
- `FULL OUTER JOIN`: Trả về tất cả các bản ghi từ cả hai bảng, bao gồm các bản ghi không khớp.

INNER JOIN

Giả sử bạn có hai bảng: `orders` và `customers`. Bảng `orders` chứa thông tin về các đơn hàng, bao gồm `order_id`, `customer_id` và `order_date`. Bảng `customers` chứa thông tin về khách hàng, bao gồm `customer_id`, `name` và `email`.

Để kết hợp hai bảng này để hiển thị thông tin về tên khách hàng và ngày đặt hàng cho mỗi đơn hàng, bạn có thể sử dụng câu lệnh SQL sau:

```
SELECT orders.order_id, customers.name, orders.order_date
FROM orders
INNER JOIN customers
ON orders.customer_id = customers.customer_id;
```

Trong ví dụ này, `orders` và `customers` là tên của các bảng cần kết hợp. `ON orders.customer_id = customers.customer_id` là điều kiện cần thiết để kết hợp các bảng này dựa trên cột `customer_id`. Các cột được chọn để hiển thị trong kết quả là `order_id`, `name` và `order_date`.

LEFT JOIN

Giả sử bạn có hai bảng: `orders` và `customers`. Bảng `orders` chứa thông tin về các đơn hàng, bao gồm `order_id`, `customer_id` và `order_date`. Bảng `customers` chứa thông tin về khách hàng, bao gồm `customer_id`, `name` và `email`.

Để kết hợp hai bảng này để hiển thị thông tin về tên khách hàng và ngày đặt hàng cho mỗi đơn hàng, kể cả các đơn hàng không có khách hàng liên quan, bạn có thể sử dụng câu lệnh SQL sau:

```
SELECT orders.order_id, customers.name, orders.order_date
FROM orders
LEFT JOIN customers
ON orders.customer_id = customers.customer_id;
```

Trong ví dụ này, `orders` và `customers` là tên của các bảng cần kết hợp. `ON orders.customer_id = customers.customer_id` là điều kiện cần thiết để kết hợp các bảng này dựa trên cột `customer_id`. `LEFT JOIN` đảm bảo rằng các bản ghi từ bảng `orders` sẽ được trả về ngay cả khi không có bản ghi khớp trong bảng `customers`. Các cột được chọn để hiển thị trong kết quả là `order_id`, `name` và `order_date`.

RIGHT JOIN

Giả sử bạn có hai bảng: `orders` và `customers`. Bảng `orders` chứa thông tin về các đơn hàng, bao gồm `order_id`, `customer_id` và `order_date`. Bảng `customers` chứa thông tin về khách hàng, bao gồm `customer_id`, `name` và `email`.

Để kết hợp hai bảng này để hiển thị thông tin về tên khách hàng và ngày đặt hàng cho mỗi đơn hàng, kể cả các đơn hàng không có khách hàng liên quan, bạn có thể sử dụng câu lệnh SQL sau:

```
SELECT orders.order_id, customers.name, orders.order_date
FROM orders
RIGHT JOIN customers
ON orders.customer_id = customers.customer_id;
```

Trong ví dụ này, `orders` và `customers` là tên của các bảng cần kết hợp. `ON orders.customer_id = customers.customer_id` là điều kiện cần thiết để kết hợp các bảng này dựa trên cột `customer_id`. `RIGHT JOIN` đảm bảo rằng tất cả các bản ghi từ bảng `customers` sẽ được trả về, kể cả khi không có bản ghi khớp trong bảng `orders`. Các cột được chọn để hiển thị trong kết quả là `order_id`, `name` và `order_date`.

FULL OUTER JOIN

Giả sử bạn có hai bảng: `orders` và `customers`. Bảng `orders` chứa thông tin về các đơn hàng, bao gồm `order_id`, `customer_id` và `order_date`. Bảng `customers` chứa thông tin về khách hàng, bao gồm `customer_id`, `name` và `email`.

Để kết hợp hai bảng này để hiển thị thông tin về tên khách hàng và ngày đặt hàng cho mỗi đơn hàng, kể cả các đơn hàng không có khách hàng liên quan, bạn có thể sử dụng câu lệnh SQL sau:

```
SELECT orders.order_id, customers.name, orders.order_date
FROM orders
FULL OUTER JOIN customers
ON orders.customer_id = customers.customer_id;
```

Trong ví dụ này, `orders` và `customers` là tên của các bảng cần kết hợp. `ON orders.customer_id = customers.customer_id` là điều kiện cần thiết để kết hợp các bảng này dựa trên cột `customer_id`. `FULL OUTER JOIN` đảm bảo rằng tất cả các bản ghi từ cả hai bảng sẽ được trả về, kể cả các bản ghi không khớp. Các cột được chọn để hiển thị trong kết quả là `order_id`, `name` và `order_date`.

MySQL - Regex

Trong MySQL, biểu thức chính quy (regular expressions) có thể được sử dụng trong các truy vấn để phù hợp với các mẫu trong giá trị chuỗi. Toán tử `REGEXP` được sử dụng để thực hiện phù hợp biểu thức chính quy.

Dưới đây là một ví dụ về truy vấn sử dụng biểu thức chính quy:

```
SELECT *
FROM my_table
WHERE my_column REGEXP '^A';
```

Truy vấn này chọn tất cả các hàng từ `my_table` trong đó giá trị trong `my_column` bắt đầu bằng chữ cái "A".

Trong ví dụ này, `^` là một ký tự đặc biệt (metacharacter) phù hợp với đầu của một chuỗi, và `A` là một ký tự đơn giản phù hợp với chữ cái "A". Các ký tự đặc biệt khác có thể được sử dụng trong biểu thức chính quy bao gồm `.` (phù hợp với bất kỳ ký tự nào), `*` (phù hợp với không hoặc nhiều lần xuất hiện của ký tự trước đó), `+` (phù hợp với một hoặc nhiều lần xuất hiện của ký tự trước đó), và `[]` (phù hợp với bất kỳ ký tự nào trong các dấu ngoặc vuông).

Biểu thức chính quy có thể rất mạnh mẽ cho việc phù hợp các mẫu phức tạp, nhưng cũng có thể khó đọc và viết. Quan trọng là sử dụng chúng một cách khôn ngoan và kiểm tra kỹ truy vấn của bạn để đảm bảo rằng chúng trả về kết quả mong muốn.

MySQL - Transaction

Trong MySQL, transaction là một chuỗi các thao tác cơ sở dữ liệu được thực hiện như một đơn vị đơn lẻ. Các thao tác này được thực hiện hoặc hoàn tác toàn bộ, và không bị phân mảnh hay bị gián đoạn bởi các thao tác khác trong cùng một thời điểm.

Các transaction được sử dụng để đảm bảo tính toàn vẹn dữ liệu trong các ứng dụng cơ sở dữ liệu. Nếu một transaction không thành công, tất cả các thao tác trong transaction đó sẽ bị hoàn tác, và cơ sở dữ liệu sẽ được phục hồi về trạng thái trước khi transaction được thực hiện.

Để bắt đầu một transaction trong MySQL, bạn có thể sử dụng câu lệnh `START TRANSACTION`. Sau khi transaction đã bắt đầu, bạn có thể thực hiện các thao tác cơ sở dữ liệu bình thường bằng các câu lệnh SQL thông thường.

Khi bạn đã hoàn tất các thao tác trong transaction, bạn có thể chấp nhận các thay đổi bằng cách sử dụng câu lệnh `COMMIT`. Nếu bạn muốn hủy bỏ các thay đổi trong transaction, bạn có thể sử dụng câu lệnh `ROLLBACK`.

Dưới đây là một ví dụ về cách sử dụng transaction trong MySQL:

```
START TRANSACTION;
UPDATE accounts SET balance = balance - 100 WHERE account_id = 1;
UPDATE accounts SET balance = balance + 100 WHERE account_id = 2;
COMMIT;
```

Trong ví dụ này, hai thao tác được thực hiện trên hai tài khoản khác nhau. Nếu cả hai thao tác đều thành công, thì sẽ được chấp nhận bằng cách sử dụng câu lệnh `COMMIT`. Nếu một trong hai thao tác

không thành công, tất cả các thao tác trong transaction sẽ bị hoàn tác bằng cách sử dụng câu lệnh `ROLLBACK`.

MySQL - ALTER Command

Trong MySQL, câu lệnh `ALTER` được sử dụng để thay đổi cấu trúc của một bảng hoặc cơ sở dữ liệu. Các thay đổi có thể bao gồm thêm hoặc xóa cột, thay đổi kiểu dữ liệu của cột, thêm hoặc xóa ràng buộc, hoặc thay đổi tên của bảng hoặc cơ sở dữ liệu.

Dưới đây là một số ví dụ về cách sử dụng câu lệnh `ALTER` trong MySQL:

Thêm cột

Để thêm một cột mới vào một bảng, bạn có thể sử dụng câu lệnh `ALTER TABLE` như sau:

```
ALTER TABLE table_name
ADD COLUMN column_name data_type;
```

Ví dụ:

```
ALTER TABLE customers
ADD COLUMN email VARCHAR(255);
```

Xóa cột

Để xóa một cột khỏi một bảng, bạn có thể sử dụng câu lệnh `ALTER TABLE` như sau:

```
ALTER TABLE table_name
DROP COLUMN column_name;
```

Ví dụ:

```
ALTER TABLE customers
DROP COLUMN email;
```

Thay đổi kiểu dữ liệu của cột

Để thay đổi kiểu dữ liệu của một cột trong một bảng, bạn có thể sử dụng câu lệnh `ALTER TABLE` như sau:

```
ALTER TABLE table_name
MODIFY COLUMN column_name new_data_type;
```

Ví dụ:

```
ALTER TABLE customers
MODIFY COLUMN phone_number INT;
```

Đổi tên bảng

Để đổi tên của một bảng, bạn có thể sử dụng câu lệnh `ALTER TABLE` như sau:

```
ALTER TABLE old_table_name
RENAME TO new_table_name;
```

Ví dụ:

```
ALTER TABLE customers
RENAME TO clients;
```

MySQL - INDEX

Trong MySQL, index là một cơ chế tìm kiếm nhanh để tìm kiếm các bản ghi trong bảng. Index được tạo ra bằng cách lưu trữ một bản sao của một phần hoặc toàn bộ bảng trong một cấu trúc dữ liệu khác nhau, cho phép truy cập nhanh hơn vào các bản ghi.

Để tạo một index trong MySQL, bạn có thể sử dụng câu lệnh `CREATE INDEX`. Đây là một ví dụ về cách tạo một index trên một cột trong bảng `users`:

```
CREATE INDEX idx_users_email ON users (email);
```

Trong ví dụ này, `idx_users_email` là tên của index được tạo ra, và `email` là tên của cột được sử dụng để tạo index.

Khi tìm kiếm các bản ghi trong một bảng có index, MySQL sử dụng index để tìm kiếm các bản ghi nhanh hơn. Nếu không có index, MySQL sẽ phải quét toàn bộ bảng để tìm kiếm các bản ghi, điều này có thể mất nhiều thời gian hơn.

Tuy nhiên, việc sử dụng index không phải lúc nào cũng là tốt nhất. Việc tạo index cũng có thể làm chậm các thao tác cập nhật và thêm mới dữ liệu vào bảng. Do đó, bạn nên tạo index cho các cột được sử dụng thường xuyên để tìm kiếm hoặc sắp xếp dữ liệu, và tránh tạo index cho các cột có ít hoặc không được sử dụng trong các truy vấn.

MySQL - Temporary Table

Trong MySQL, Temporary Table là bảng tạm thời được tạo ra trong bộ nhớ và sẽ tự động bị xóa khi kết thúc session hoặc khi kết thúc transaction. Temporary Table thường được sử dụng để lưu trữ tạm thời dữ liệu trong quá trình thực hiện các thao tác phức tạp trên các bảng khác.

Để tạo một Temporary Table trong MySQL, bạn có thể sử dụng câu lệnh `CREATE TEMPORARY TABLE`. Dưới đây là một ví dụ về cách tạo một Temporary Table:

```
CREATE TEMPORARY TABLE temp_table (
  id INT PRIMARY KEY,
  name VARCHAR(255)
);
```

Trong ví dụ này, `temp_table` là tên của Temporary Table được tạo ra, và bảng này có hai cột là `id` và `name`.

Sau khi Temporary Table đã được tạo ra, bạn có thể thực hiện các thao tác trên bảng này bằng các câu lệnh SQL thông thường. Ví dụ:

```
INSERT INTO temp_table (id, name) VALUES (1, 'John');
INSERT INTO temp_table (id, name) VALUES (2, 'Jane');

SELECT * FROM temp_table;
```

Trong ví dụ này, chúng ta đã thêm hai bản ghi vào Temporary Table và sau đó truy vấn tất cả các bản ghi trong bảng.

Temporary Table là một công cụ mạnh mẽ trong MySQL để lưu trữ tạm thời dữ liệu khi thực hiện các thao tác phức tạp trên các bảng khác. Tuy nhiên, bạn cần cẩn thận khi sử dụng Temporary Table, vì chúng có thể làm chậm hiệu suất của các thao tác cơ sở dữ liệu và sử dụng quá nhiều bộ nhớ nếu không được sử dụng đúng cách.

MySQL - Clone Table

Trong MySQL, câu lệnh `CREATE TABLE` được sử dụng để tạo một bảng mới trong cơ sở dữ liệu. Tuy nhiên, đôi khi bạn muốn tạo một bảng mới bằng cách sao chép một bảng đã có, thay vì viết lại câu lệnh `CREATE TABLE` từ đầu. Để sao chép một bảng trong MySQL, bạn có thể sử dụng câu lệnh `CREATE TABLE ... LIKE`.

Cú pháp

Cú pháp của câu lệnh `CREATE TABLE ... LIKE` như sau:

```
CREATE TABLE new_table_name LIKE existing_table_name;
```

Trong đó:

- `new_table_name`: Tên của bảng mới sẽ được tạo ra.
- `existing_table_name`: Tên của bảng đã có sẽ được sao chép.

Ví dụ

Dưới đây là một ví dụ về cách sao chép một bảng trong MySQL:

```
CREATE TABLE products_copy LIKE products;
```

Trong ví dụ này, chúng ta tạo một bảng mới có tên là `products_copy`, bằng cách sao chép cấu trúc của bảng `products`.

Sau khi đã sao chép bảng, bạn cũng có thể sao chép dữ liệu từ bảng gốc vào bảng mới bằng cách sử dụng câu lệnh INSERT INTO SELECT. Ví dụ:

```
INSERT INTO products_copy SELECT * FROM products;
```

Trong ví dụ này, chúng ta sao chép toàn bộ dữ liệu từ bảng `products` sang bảng `products_copy`.