Trong MySQL Workbench 8.0 CE (Community Edition), các kiểu dữ liệu chính bao gồm:

1. **Kiểu dữ liệu số**:
2. **TINYINT, SMALLINT, MEDIUMINT, INT, BIGINT**: Các kiểu số nguyên
3. **FLOAT, DOUBLE**: Các kiểu số dấu phẩy động
4. **Kiểu dữ liệu chuỗi**:
5. **CHAR, VARCHAR**: Lưu trữ chuỗi ký tự
6. **BINARY, VARBINARY**: Lưu trữ dữ liệu nhị phân
7. **BLOB, TEXT**: Lưu trữ dữ liệu văn bản lớn
8. **Kiểu dữ liệu ngày tháng**:

* **DATE, DATETIME, TIMESTAMP, TIME, YEAR**

1. **Kiểu dữ liệu đặc biệt**:
2. **ENUM**: Lưu trữ một trong các giá trị đã định sẵn
3. **SET**: Lưu trữ một tập hợp các giá trị đã định sẵn
4. **JSON**: Lưu trữ dữ liệu JSON

Lưu ý rằng, trong **MySQL Workbench 8.0 CE**, không có kiểu dữ liệu **NVARCHAR** như trong **SQL Server**. Thay vào đó, bạn sẽ sử dụng **VARCHAR** để lưu trữ chuỗi ký tự **Unicode**

Ngoài ra, các kiểu dữ liệu có thể được tùy chỉnh kích thước hoặc độ chính xác tùy theo nhu cầu

Trong **MySQL Workbench 8.0 CE**, không có kiểu dữ liệu **Boolean** riêng biệt. Thay vào đó, có hai cách để lưu trữ dữ liệu **boolean**:

1. Sử dụng kiểu **TINYINT(1)**: Trong MySQL, kiểu **TINYINT(1)** thường được sử dụng để lưu trữ dữ liệu **boolean**. Trong đó, **0** thể hiện giá trị "**false**" và **1** thể hiện giá trị "**true**"

Ví dụ:

CREATE TABLE users (

id INT PRIMARY KEY,

is\_active TINYINT(1) DEFAULT 1

);

1. Sử dụng kiểu **BIT(1)**: Kiểu **BIT(1)** cũng có thể được sử dụng để lưu trữ dữ liệu **boolean**, với **0** là "**false**" và **1** là "**true**"

Ví dụ:

CREATE TABLE users (

id INT PRIMARY KEY,

is\_active BIT(1) DEFAULT b'1'

);

Cả hai cách trên đều có thể được sử dụng để lưu trữ dữ liệu boolean trong MySQL. Tuy nhiên, việc sử dụng TINYINT(1) phổ biến hơn vì nó tương thích với nhiều ngôn ngữ lập trình hơn

Ngoài ra, MySQL cũng có thể sử dụng các biểu thức logic như WHERE is\_active = 1 để kiểm tra giá trị boolean

*\*số trong cặp ngoặc tròn "****TINYINT(1)****" không nhất thiết phải luôn là* ***1****. Đây được gọi là "****kích thước hiển thị****"* ***(display size)*** *và không ảnh hưởng đến giá trị được lưu trữ*

Trong MySQL, **kích thước hiển thị** chỉ được sử dụng để định dạng đầu ra, chẳng hạn như khi sử dụng lệnh **SELECT**. Nó không ảnh hưởng đến kích thước bộ nhớ thực tế được sử dụng để lưu trữ giá trị

Bạn hoàn toàn có thể viết số khác ngoài 1 trong cặp ngoặc tròn, ví dụ:

CREATE TABLE users (

id INT PRIMARY KEY,

is\_active TINYINT(2) DEFAULT 1

);

Trong ví dụ trên, cột is\_active vẫn lưu trữ giá trị boolean (0 hoặc 1), nhưng khi hiển thị kết quả, giá trị sẽ được căn chỉnh trong một **trường** có **độ rộng** là 2 ký tự

Tóm lại, số trong cặp ngoặc tròn chỉ là **kích thước hiển thị**, không ảnh hưởng đến kiểu dữ liệu thực tế. Bạn có thể sử dụng bất kỳ giá trị nào trong đó tùy theo nhu cầu hiển thị

dưới đây là ví dụ về các **kiểu dữ liệu** **ngày tháng** trong MySQL và sự khác biệt giữa chúng, cùng với gợi ý về khi nào nên sử dụng:

**DATE:**

Định dạng: YYYY-MM-DD (ví dụ: 2023-04-14)

Miền giá trị: từ '1000-01-01' đến '9999-12-31'

Sử dụng khi: Chỉ cần lưu trữ ngày tháng mà không cần thông tin về thời gian

**DATETIME**:

Định dạng: YYYY-MM-DD HH:MM:SS (ví dụ: 2023-04-14 15:30:00)

Miền giá trị: từ '1000-01-01 00:00:00' đến '9999-12-31 23:59:59'

Sử dụng khi: Cần lưu trữ thông tin ngày tháng và thời gian

**TIMESTAMP**:

Định dạng: YYYY-MM-DD HH:MM:SS (ví dụ: 2023-04-14 15:30:00)

Miền giá trị: từ '1970-01-01 00:00:01' UTC đến '2038-01-19 03:14:07' UTC

Sử dụng khi: Cần lưu trữ thời gian theo chuẩn UTC, thường dùng để đánh dấu thời gian tạo/chỉnh sửa bản ghi

**TIME**:

Định dạng: HH:MM:SS (ví dụ: 15:30:00)

Miền giá trị: từ '-838:59:59' đến '838:59:59'

Sử dụng khi: Chỉ cần lưu trữ thông tin về thời gian mà không cần ngày tháng

**YEAR**:

Định dạng: YYYY (ví dụ: 2023)

Miền giá trị: từ 1901 đến 2155

Sử dụng khi: Chỉ cần lưu trữ thông tin về năm

Tóm lại, việc lựa chọn kiểu dữ liệu ngày tháng phụ thuộc vào nhu cầu của ứng dụng. DATE thích hợp khi chỉ cần lưu trữ ngày tháng, DATETIME và TIMESTAMP dùng khi cần lưu cả ngày tháng và thời gian, TIME khi chỉ cần lưu trữ thời gian, và YEAR khi chỉ cần lưu năm

**AUTO\_INCREMENT** là một tính năng trong SQL giúp tự động tăng giá trị của một **trường (cột)** khi thêm mới bản ghi vào bảng. Nó thường được sử dụng để tạo **khóa chính (primary key)** cho các bảng

**CURRENT\_DATE()** là một **hàm** trong SQL (cụ thể là trong **MySQL**) trả về ngày hiện tại của hệ thống. Nó sẽ trả về một giá trị kiểu **DATE** tương ứng với ngày tháng năm hiện tại