

Dữ liệu và cấu trúc trong MySQL

1- Kiểu dữ liệu

1.1- Kiểu dữ liệu số

1.1.1- Các kiểu số nguyên

Các kiểu số nguyên tiêu chuẩn của SQL như **INTEGER** (or **INT**) và **SMALLINT** đều được hỗ trợ bởi **MySQL**. Và các mở rộng tiêu chuẩn, **MySQL** cũng hỗ trợ các kiểu số nguyên khác như **TINYINT**, **MEDIUMINT**, và **BIGINT**. Bảng dưới đây sẽ liệt kê các kiểu và không gian lưu trữ đòi hỏi và phạm vi của chúng (Giá trị nhỏ nhất, lớn nhất cho kiểu số nguyên có dấu, và không dấu).

Kiểu dữ liệu	Độ dài (số byte)	Giá trị nhỏ nhất (Có dấu)	Giá trị lớn nhất (Có dấu)	Giá trị nhỏ nhất (Không dấu)	Giá trị lớn nhất (Không dấu)
TINYINT	1	-128	127	0	255
SMALLINT	2	-32768	32767	0	65535
MEDIUMINT	3	-8388608	8388607 to	0	16777215
INT	4	-2147483648	2147483647	0	4294967295
BIGINT	8	-9223372036854775808	9223372036854775807	0	18446744073709551615

1.1.2- Kiểu dấu chấm động (Floating-Point Types)

Kiểu dữ liệu **FLOAT** và **DOUBLE** mô tả gần đúng các giá trị số thực. **MySQL** sử dụng **4 byte** để lưu trữ dữ liệu **FLOAT** và **8 byte** dành cho kiểu dữ liệu **DOUBLE**.

Types	Description
FLOAT(M,D)	Một số chấm động (floating-point number) không thể không có dấu (unsigned). Bạn có thể định nghĩa độ dài phần nguyên (M) và độ dài phần thập phân (D). Điều này không bắt buộc và mặc định là 10,2 , ở đây 10 là độ dài phần nguyên còn 2 là số số thập phân. Phần thập phân có thể sử dụng 24 vị trí cho một số FLOAT .
DOUBLE(M,D)	Một số chấm động DOUBLE (Độ chính xác gấp 2) cũng không thể không có dấu (unsigned). Bạn có thể định nghĩa độ dài phần nguyên (M) và độ dài phần thập phân (D). Điều này không bắt buộc và mặc định là 16,4 , ở đó 16 là độ dài phần nguyên còn 4 là độ dài phần thập phân. Phần thập phân có thể sử dụng tới 53 vị trí cho một số DOUBLE . REAL là một từ đồng nghĩa với DOUBLE .

Following table shows the required storage and range (maximum and minimum value for signed and unsigned integer) for each floating-point type.

Kiểu dữ liệu	Độ dài (Số Bytes)	Giá trị nhỏ nhất (Có dấu)	Giá trị lớn nhất (Có dấu)	Giá trị nhỏ nhất (Không dấu)	Giá trị lớn nhất (Không dấu)
FLOAT	4	-3.402823466E+38	-1.175494351E-38	1.175494351E-38	3.402823466E+38
DOUBLE	8	-1.7976931348623157E+ 308	-2.2250738585072014E- 308	0, and 2.2250738585072014E- 308	1.7976931348623157E+ 308

1.1.3- Kiểu dấu chấm cố định (Fixed-Point Types)

Kiểu dấu chấm cố định (Fixed-Point data type) được sử dụng để bảo vệ độ chính xác (precision), ví dụ như với dữ liệu tiền tệ. Trong MySQL kiểu DECIMAL và NUMERIC lưu trữ chính xác các dữ liệu số. MySQL 5.6 lưu trữ giá trị DECIMAL theo định dạng nhị phân.

Trong SQL chuẩn, cú pháp **DECIMAL(5,2)** nghĩa là độ chính xác (precision) là 5, và 2 là phần thập phân (scale), nghĩa là nó có thể lưu trữ một giá trị có 5 chữ số trong đó có 2 số thập phân. Vì vậy giá trị lưu trữ sẽ là -999.99 tới 999.99. Cú pháp **DECIMAL(M)** tương đương với **DECIMAL(M,0)**. Tương tự **DECIMAL** tương đương với **DECIMAL(M,0)** ở đây M mặc định là 10.

Độ dài tối đa các con số cho **DECIMAL** là 65.

1.1.4- Kiểu dữ liệu Bit (Bit Value Types)

Kiểu dữ liệu **BIT** được sử dụng để lưu trữ trường giá trị **bit**. Kiểu **BIT(N)** có thể lưu trữ N giá trị **bit**. **N** có phạm vi từ 1 tới 64. Để chỉ định giá trị các **bit**, có thể sử dụng **b'value'**. **value** là dãy các số nhị phân 0 hoặc 1. Ví dụ **b'111'** mô tả số 7, và **b'10000000'** mô tả số 128.

1.1.5- Kiểu số và thuộc tính

MySQL hỗ trợ một mở rộng cho việc tùy chọn chỉ định độ dài hiển thị là một số nguyên trong dấu ngoặc ngay sau từ khóa kiểu dữ liệu.

Kiểu	Mô tả
TYPE(N)	Tại đây N là một số nguyên hiển thị chiều rộng cho kiểu độ dài lên đến N chữ số.
ZEROFILL	Các khoảng đệm (padding) được thay thế bởi số 0. Ví dụ với cột kiểu INT(3) ZEROFILL , 7 sẽ hiển thị là 007.

1.2- Các kiểu Date and Time

Các kiểu dữ liệu ngày tháng và thời gian đại diện bao gồm DATE, TIME, DATETIME, TIMESTAMP, and YEAR. Mỗi kiểu có một phạm vi hợp lệ.

1.2.1- Kiểu dữ liệu DATETIME, DATE, và TIMESTAMP

Kiểu dữ liệu	Mô tả	Định dạng hiển thị	Phạm vi
DATETIME	Sử dụng khi bạn cần giá trị lưu trữ cả hai thông tin ngày tháng và thời gian.	YYYY-MM-DD HH:MM:SS	'1000-01-01 00:00:00' to '9999-12-31 23:59:59'.
DATE	Sử dụng khi bạn muốn lưu trữ chỉ thông tin ngày tháng.	YYYY-MM-DD	'1000-01-01' to '9999-12-31'.
TIMESTAMP	Lưu trữ cả hai thông tin ngày tháng và thời gian. Giá trị này sẽ được chuyển đổi từ múi giờ hiện tại sang UTC trong khi lưu trữ, và sẽ chuyển trở lại múi giờ hiện tại khi lấy dữ liệu ra.	YYYY-MM-DD HH:MM:SS	'1970-01-01 00:00:01' UTC to '2038-01-19 03:14:07' UTC

1.2.2- Kiểu dữ liệu TIME

MySQL lấy và hiển thị thời gian theo định dạng ''HH:MM:SS' (hoặc định dạng 'HHH:MM:SS' đối với các giá trị giờ lớn). Giá trị của TIME có thể trong khoảng '-838:59:59' tới '838:59:59'. Phần thời gian có thể lớn bởi vì kiểu TIME có thể không chỉ mô tả thời gian của một ngày (Vốn chỉ có tối đa 24 giờ), mà nó có thể là thời gian trôi qua hoặc khoảng thời gian giữa hai sự kiện (Cái mà có thể lớn hơn 24h thậm chí có giá trị âm).

```
1 -- !!
2 SELECT
3   TIME_FORMAT(foo_hour, '%H:%i')
4   FROM bar
```

1.2.3- Kiểu dữ liệu YEAR

Kiểu dữ liệu YEAR được sử dụng 1-byte để mô tả giá trị. Nó có thể khai báo YEAR(2) hoặc YEAR(4) chỉ định rõ chiều rộng hiển thị là 2 hay 4 ký tự. Nếu không chỉ rõ chiều rộng mặc định là 4 ký tự.

YEAR(4) và **YEAR(2)** khác nhau định danh hiển thị nhưng có cùng phạm vi giá trị.
 Với định dạng 4 số, MySQL hiển thị giá trị **YEAR** theo định dạng **YYYY**, với phạm vi 1901 tới 2155, hoặc 0000.
 Với định dạng 2 số, MySQL chỉ hiển thị 2 số cuối; ví dụ 70 (1970 hoặc 2070) hoặc 69 (2069).

Bạn có thể chỉ định giá trị **YEAR** theo một vài định dạng khác nhau:

Độ dài chuỗi	Phạm vi
Chuỗi 4 con số	'1901' tới '2155'.
Một số có 4 con số	1901 tới 2155.
Chuỗi 1 hoặc 2 chữ số	Giá trị từ '0' tới '99'. MySQL chuyển đổi '0' tới '69' tương đương với giá trị YEAR từ 2000-2069. Và '70' tới '99' tương đương với YEAR từ 1970 tới 1999.
Một số có 1 hoặc 2 chữ số	Giá trị 1 từ 99. MySQL chuyển đổi giá trị từ 1 tới 69 tương đương với YEAR từ 2001 tới 2069. Và 70 tới 99 tương đương với YEAR từ 1970 tới 1999.

1.2.4- Khác nhau giữa kiểu dữ liệu Datetime và Timestamp trong MySQL

Kiểu dữ liệu **DATETIME** được sử dụng khi bạn cần lưu trữ cả hai thông tin ngày tháng và thời gian. MySQL lấy và hiển thị **DATETIME** theo định dạng '**YYYY-MM-DD HH:MM:SS**'. Và hỗ trợ phạm vi từ '1000-01-01 00:00:00' tới '9999-12-31 23:59:59'.

Kiểu dữ liệu **TIMESTAMP** cũng được sử dụng khi bạn muốn lưu trữ cả hai thông tin ngày tháng và thời gian. **TIMESTAMP** có phạm vi '1970-01-01 00:00:01' UTC tới '2038-01-19 03:14:07' UTC

Sự khác biệt chính của **DATETIME** và **TIMESTAMP** là giá trị của **TIMESTAMP** được chuyển đổi từ múi giờ hiện tại sang UTC trong khi lưu trữ, và chuyển ngược trở lại từ UTC sang múi giờ hiện tại trong lúc lấy ra. Còn kiểu dữ liệu **DATETIME** thì không có gì thay đổi.

1.3- Các kiểu chuỗi (String Types)

Các kiểu dữ liệu String bao gồm:

- CHAR
- VARCHAR
- BINARY
- VARBINARY
- BLOB
- TEXT
- ENUM
- SET,

1.3.1- Kiểu dữ liệu CHAR và VARCHAR

Kiểu dữ liệu CHAR và VARCHAR là giống nhau, nhưng khác nhau ở cách chúng được lưu trữ và truy xuất. Chúng cũng khác nhau về chiều dài tối đa và giữ lại hay không khoảng trắng phía trước (trailing spaces).

The → without a trailing space.
The → with a trailing space.

Kiểu dữ liệu	Mô tả	Định dạng hiển thị	Phạm vi các ký tự
CHAR	Chứa chuỗi không phải nhị phân (non-binary strings). Độ dài là cố định như khi bạn khai báo cột của bảng. Khi lưu trữ chúng được độn thêm bên phải (right-padded) để có độ dài chỉ được chỉ định.	Khoảng trắng phía trước (Trailing spaces) được loại bỏ	Giá trị từ 0 tới 255
VARCHAR	Chứa các chuỗi không phải nhị phân (non-binary strings). Cột là chuỗi có chiều dài thay đổi.	Giống như lưu trữ.	Giá trị từ 0 tới 255 với MySQL trước phiên bản 5.0.3. Và 0 tới 65,535 với các phiên bản MySQL 5.0.3 hoặc mới hơn.

1.3.2- Kiểu dữ liệu BINARY và VARBINARY

Các kiểu dữ liệu **BINARY** và **VARBINARY** tương tự như **CHAR** và **VARCHAR**, ngoại trừ việc chúng có chứa các chuỗi nhị phân chứ không phải là chuỗi non-binary.

Kiểu dữ liệu	Mô tả	Phạm vi các bytes
BINARY	Chứa các chuỗi nhị phân (Binary Strings)	Giá trị từ 0 tới 255
VARBINARY	Chứa các chuỗi nhị phân (Binary Strings)	Giá trị từ 0 tới 255 đối với MySQL trước 5.0.3, và 0 tới 65,535 với MySQL 5.0.3 và mới hơn.

1.3.3- Kiểu dữ liệu BLOB và TEXT

BLOB là một đối tượng nhị phân lớn (Binary Large OBject) có thể chứa một lượng lớn dữ liệu. Có bốn loại BLOB, TINYBLOB, BLOB, MEDIUMBLOB, và LONGBLOB. Những chỉ khác nhau về độ dài tối đa của các giá trị mà họ có thể giữ.

Bốn loại TEXT là TINYTEXT, TEXT, MEDIUMTEXT, và LONGTEXT. Chúng tương ứng với bốn loại BLOB và có độ dài tối đa và các yêu cầu lưu trữ tương tự.

Kiểu dữ liệu	Mô tả	Loại	Độ dài
BLOB	Đối tượng nhị phân lớn (Large binary object) chứa khối lượng dữ liệu lớn. Giá trị được xem như một chuỗi nhị phân. Bạn không cần thiết phải chỉ định độ dài khi tạo cột.	TINYBLOB	Chiều dài tối đa là 255 ký tự.
		MEDIUMBLOB	Chiều dài tối đa là 16777215 ký tự.
		LONGBLOB	Chiều dài tối đa là 4294967295 ký tự
TEXT	Lưu trữ giá trị được coi như một chuỗi các ký tự có mã hóa (character set).	TINYBLOB	Chiều dài tối đa là 255 ký tự.
		MEDIUMBLOB	Chiều dài tối đa là 16777215 ký tự.
		LONGBLOB	Chiều dài tối đa là 4294967295 ký tự.

1.3.4- Kiểu dữ liệu ENUM

Một đối tượng chuỗi có giá trị được chọn từ một danh sách các giá trị được đưa ra ở thời điểm tạo ra bảng. Ví dụ:

```
1 | CREATE TABLE My_Table (
2 |   length ENUM('small', 'medium', 'large')
3 | );
```

1.3.5- Kiểu dữ liệu SET

Một đối tượng chuỗi có không hoặc nhiều dấu phẩy tách giá trị (tối đa 64). Các giá trị được lựa chọn từ một danh sách các giá trị được đưa ra ở thời điểm tạo ra bảng

- TODO Example:

2- Phân biệt các câu lệnh DDL, DML và DCL

Các câu lệnh trong MySQL có thể được chia làm 3 loại:

- DDL - Data Definition Language

- DML - Data Manipulation Language
- DCL - Data Control Language

Trong tài liệu hướng dẫn này tôi sẽ đề cập tới 2 nhóm con: **DDL & DCL**.

2.1- DML *

Data Manipulation Language (DML) các câu lệnh được sử dụng để quản lý dữ liệu bên trong SCHEME. Ví dụ:

- SELECT - Lấy dữ liệu từ một database
- INSERT - Trèn dữ liệu vào một bảng
- UPDATE - Cập nhập dữ liệu đang tồn tại trong bảng
- DELETE - Xóa các bản ghi trên bảng.
- MERGE - UPSERT Toán tử insert hoặc update
- CALL - Gọi thủ tục trong DB
- EXPLAIN PLAN - Giải thích đường dẫn truy cập dữ liệu
- LOCK TABLE - Điều khiển sự đồng thời (control concurrency).

2.2- DDL

Data Definition Language (DDL) là các câu lệnh được sử dụng để định nghĩa cấu trúc database hoặc schema. Ví dụ:

- CREATE - Tạo các đối tượng trong database
- ALTER - Sửa đổi cấu trúc của database
- DROP - Xóa các đối tượng trong database
- TRUNCATE - Xóa hết các bản ghi trong bảng, bao gồm cả các không gian được phân bổ cho các bản ghi đã xóa
- COMMENT - Thêm các chú thích vào từ điển dữ liệu (data dictionary).
- RENAME - Thay đổi tên của đối tượng.

2.2.1- CREATE TABLE

Cú pháp:

```

1 -- Chú ý một bảng không nhất thiết phải có khóa chính.
2 -- Phải có dấu chấm phẩy cuối dòng lệnh.
3
4 CREATE TABLE <Table_Name> (
5   Column_Name_Id  DataType [NOT NULL] [AUTO_INCREMENT] ,
6   Column_name2    DataType [NOT NULL] ,
7   .....
8   Column_nameN    DataType [NOT NULL] ,
9   Primary Key (Column_Name_Id)
10 );

```

?

Tại đây có một vài giải thích:

- Thuộc tính **NOT NULL** được sử dụng nếu bạn không muốn trường này null. Như vậy nếu người dùng cố gắng tràn vào một bản ghi với dữ liệu NULL, MySQL sẽ ném ra một lỗi.
- Thuộc tính **AUTO_INCREMENT** nói với MySQL tự gán giá trị tăng dần cho trường ID.
- Từ khóa **PRIMARY KEY** được sử dụng để định nghĩa cột này là một khóa chính. Bạn có thể sử dụng nhiều cột ngăn cách nhau bởi dấu phẩy để định nghĩa một khóa chính.

Ví dụ:

```
1 Create Table Members (
2     Member_Id INT NOT NULL AUTO_INCREMENT,
3     Full_Name VARCHAR(64) NOT NULL,
4     Address VARCHAR(256),
5     Birth_Day DATE NOT NULL,
6     PRIMARY KEY (Member_Id)
7 );
```

MEMBER

Column Name	Datatype	PK	NN	UQ	BIN	UN	ZF	AI	Default
Member_Id	INT(11)	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	
Full_Name	VARCHAR(64)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
Address	VARCHAR(256)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL
Birth_Day	DATE	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	

Column Name:	Member_Id	Data Type:	INT(11)				
Collation:	Table Default	Default:					
Comments:		Pri	<input checked="" type="checkbox"/>	No	<input checked="" type="checkbox"/>	Uni	<input type="checkbox"/>
		Bi	<input type="checkbox"/>	Un	<input type="checkbox"/>	Ze	<input type="checkbox"/>
		<input checked="" type="checkbox"/> Auto					

2.2.2- ALTER

MySQL **ALTER** là một lệnh rất tiện dụng khi bạn muốn thay đổi tên của bảng, cột hoặc xóa các cột có sẵn trong bảng.

2.2.2.1- ALTER - Thêm cột vào bảng

Cú pháp:

```
1 | ALTER TABLE <Table_Name>
2 | Add <Column_Name> Data_Type [NOT NULL DEFAULT value];
```

?

Ví dụ:

```
1 | -- Thêm cột Address2 có kiểu dữ liệu Varchar(256)
2 | ALTER TABLE Members ADD Address2 Varchar(256);
3 |
4 | -- Thêm cột Active có kiểu dữ liệu Varchar(1), NOT NULL
5 | -- Chú ý nếu bảng đã có dữ liệu sử dụng với NOT NULL cần thêm giá trị mặc định.
6 | ALTER TABLE Members ADD Active Varchar(1) NOT NULL DEFAULT 'Y';
```

?

2.2.2.2- ALTER - Thay đổi tên cột

```
1 | -- Cú pháp chỉ đổi tên cột.
2 | ALTER TABLE <Table_Name> RENAME COLUMN <Column_Name> <New_Column_Name>;
3 |
4 | -- Cú pháp vừa đổi tên cột vừa đổi kiểu dữ liệu:
5 | ALTER TABLE <Table_Name> RENAME COLUMN <Column_Name> <New_Column_Name> Data_Type;
```

?

2.2.2.3- ALTER - Xóa cột trong bảng

Cú pháp:

```
1 | ALTER TABLE <Table_Name> DROP <Column_Name>;
```

?

Ví dụ:

```
1 | ALTER TABLE Member DROP Address2;
```

?

2.2.2.4- ALTER - Thay đổi kiểu dữ liệu của cột

Cú pháp:

```
1 | ALTER TABLE <Table_Name> MODIFY <Column_Name> NewDataType;
2 |
```

?

Ví dụ:

```
1 | -- Kiểu dữ liệu cũ của Cột Address là Varchar(225)
2 | -- Sửa thành Varchar(512).
3 | ALTER TABLE Member MODIFY Address Varchar(512);
```

?

2.2.2.5- ALTER - Thay đổi tên bảng

Cú pháp:

```
1 | ALTER TABLE <Table_Name> RENAME TO <New_Table_Name>;
2 |
```

?

Ví dụ:

```
1 | ALTER TABLE Member RENAME TO User_Member;
```

?

2.2.3- DROP TABLE

Rất dễ dàng xóa (drop) một bảng MySQL hiện có, nhưng bạn cần phải rất cẩn thận khi xóa bất kỳ bảng hiện tại, vì dữ liệu bị mất sẽ không được phục hồi sau khi xóa một bảng.

Cú pháp:

```
1 | DROP TABLE <table_name>;
```

?

2.3- DCL

Data Control Language (DCL) là các lệnh điều khiển truy cập dữ liệu. Ví dụ:

- GRANT - Gán quyền truy cập cơ sở dữ liệu
- REVOKE - Rút đặc quyền được cho bởi lệnh GRANT.