MySQL - Tipos de datos.

Es importante especificar el tipo de dato de cada campo o atributo lo más ajustado a la realidad de los contenidos que albergará. Esto repercutirá en eficacia tanto en espacio de almacenamiento como en velocidad de proceso.

Un tipo de dato concreto, además de indicar que valores puede tener, indica las operaciones que se puede realizar sobre él.

Básicamente los tipos de datos son: numéricos, caracteres, binarios, fecha-hora y booleanos. Pero dentro de estos tipos genéricos hay gran variedad de ellos para atender las distintas necesidades de uso. Ej: aún siendo numéricos, no es lo mismo almacenar una calificación que puede ser entre 0 y 10, que el presupuesto de una empresa que puede llegar a cientos de miles incluso millones.

1. Tipos numéricos.

1.1. Números enteros.

Tipo	Tamaño en bytes	Mínimo valor	Máximo valor.
TINYINT	1	-128	127
TINYINT UNSIGNED	1	0	255
SMALLINT	2	-32768	32767
SMALLINT UNSIGNED	2	0	65535
MEDIUMINT	3	-8388608	8388607
MEDIUMINT UNSIGNED	3	0	16777215
INT	4	-2147483648	2147483647
INT UNSIGNED	4	0	4294967295
INTEGER	4	-2147483648	2147483647
INTEGER UNSIGNED	4	0	4294967295
BIGINT	8	-0,922337E19	0,922337E19
BIGINT UNSIGNED	8	0	0,184467E20

ZEROFILL

Todos los tipos numéricos admiten el atributo ZEROFILL, esto incluye automáticamente el atributo UNSIGNED quedando: UNSIGNED ZEROFILL.

Se puede especificar detrás de INT y entre paréntesis un número: INT(N). Esto solo tiene sentido si se utiliza el atributo ZEROFILL pues para mostrar el contenido del campo mostrará como mínimo hasta N dígitos y si el valor tiene menos lo completa con ceros. Solo afecta a la forma de mostrar los datos.

Ej: supongamos un campo definido como INT(5) UNSIGNED ZEROFILL y que contiene el valor 23, por pantalla se mostrará: 00023. Si el valor fuese 120057, se mostrará 120057. Y siempre se almacenará en 4 bytes.

Otros tipos numéricos enteros:

BIT(M): M será un valor entre 1 y 64. Indica el número de bits que vamos a utilizara para ese campo. Si se omite el valor de M se utiliza un bit por defecto.

Observaciones. Podrá almacenar un entero positivo entre 0 y 2^M-1. Ejemplo: un atributo definido como BIT(4), puede contener números entre 0 y 15.

El contenido de un atributo BIT, no se muestra directamente en la pantalla con SELECT, pero si tiene almacenado los valores que se hayan asignado. Para mostrarlo habría que pasarlo a un tipo numérico, simplemente sumando 0 se mostraría.

BOOL, BOOLEAN: Es equivalente a TINYINT(1). Si el valor es 0 se considera FALSE, en caso contrario TRUE.

SERIAL: Es un alias para BIGINT UNSIGNED NOT NULL AUTO_INCREMENT UNIQUE.

1.2. Números en coma flotante.

Se utiliza para representar números con decimales en notación científica: mantisa y exponente.

Con notación científica tenemos los siguientes tipos:

Tipo	Tamaño en bytes	Mínimo valor	Máximo valor.
FLOAT	4		
FLOAT(m,d)	4	-3,402823466E+38	-1,175494351E-38
FLOAT(m,d) UNSIGNED	4	1,175494351E-38	3,402823466E+38
DOUBLE	8		
DOUBLE (m,d)	8	-1,797693..E+308	-2,225073..E-308
DOUBLE(m,d) UNSIGNED	8	2,225073..E-308	1,797693..E+308

m: Indica el número de dígitos en total que se va a mostrar.
d: Indica el número de decimales que se tendrá en cuenta.

Ejemplo: Un campo declarado FLOAT(7,3) podría representar entre: -9999,999 y 9999,999.

1.3. Números en coma fija.

Tipo	Tamaño en bytes.
DECIMAL	
DECIMAL(m,d)	M + 2 bytes si d > 0; M + 1 bytes si d = 0
DECIMAL(m,d) UNSIGNED	M + 1 bytes si d > 0
NUMERIC	
NUMERIC(m,d)	M + 2 bytes si d > 0; M + 1 bytes si d = 0
NUMERIC(m,d) UNSIGNED	M + 1 bytes si d > 0

m: Indica el número de dígitos en total, entre 1 y 65.
d: Indica el número de cifras decimales, entre 0 y 30.

NUMERIC y DECIMAL son equivalentes. Este tipo de dato es apropiado para trabajar con cantidades económicas en los que no se haga redondeo.

2. Fechas y Tiempo.

Tipo	Tamaño	Descripción	Mínimo	Máximo
DATE	3	YYYY-MM-DD	1000-01-01	9999-12-31
DATETIME	8	YYYY-MM-DD HH:MM:SS	1000-01-01 00:00:00	9999-12-31 23:59:59
TIMESTAMP	4	YYYY-MM-DD HH:MM:SS	1970-01-01 00:00:00	2038-01-09 03:14:07
TIME	3	HH:MM:SS	-838:59:59	838:59:59
YEAR	[(2|4)]	YY o YYYY	YY: 70 (1970) YYYY: 1901	YY: 69 (2069) YYYY: 2155

Al definir el tipo de dato YEAR lo almacenará en 4 dígitos. Cuando se le asigne un valor si tiene dos cifras priorizará a YY, es decir, entre 1970 y 2069. Si no considerará que el rango del YYYY. Un valor fuera del rango YYYY (1901-2155) provocará un error.

Ej: si queremos almacenar el año 1935, se ha de usar el: 1935, pues si se usa 35 se considerará 2035.

3. Cadenas de caracteres.

Tipo	Descripción
CHAR(m)	m: ha de estar entre 1 y 255. Siempre ocupa m bytes. Rellena a blancos por la derecha hasta el tamaño especificado. Si no se especifica el tamaño es 1.
VARCHAR(m)	m: ha de estar entre 1 y 255. Almacenara cadenas de longitud variable hasta un máximo indicado por m. Se ha de especificar m. Cada campo ocupa entre 2 y m+1 bytes, según la longitud del contenido.
TEXT	Campo carácter de longitud máxima 65535 bytes. Ocuparía la longitud del contenido más 2 bytes.
MEDIUMTEXT	Campo carácter de longitud máxima 16 MB. Ocuparía la longitud del contenido más 3 bytes.
LONGTEXT	Campo carácter de longitud máxima 4GB. Ocuparía la longitud del contenido más 4 bytes.

Los tipos TEXT se utilizan para almacenar gran cantidad de texto y no soportan la cláusula DEFAULT. Los valores de TEXT se tratan como cadenas no binarias (cadenas de caracteres). Tienen un conjunto de caracteres y los valores se ordenan y comparan según la clasificación del conjunto de caracteres.

4. Datos binarios.

Estos tipos permiten almacenar gran cantidad de información en un campo. Esa información puede ser un fichero pdf, una imágen, audio,..

No permiten la cláusula DEFAULT.

BLOB: Binary large Object.

Los valores BLOB se tratan como cadenas binarias (cadenas de bytes). No tienen juego de caracteres y la clasificación y la comparación se basan en los valores numéricos de los bytes que contengan cada campo.

Tipo	Descripción.
BLOB	Campo de longitud máxima 65535 bytes. Ocuparía la longitud del contenido más 2 bytes.
MEDIUMBLOB	Campo de longitud máxima 16 MB. Ocuparía la longitud del contenido más 3 bytes.
LONGBLOB	Campo de longitud máxima 4GB. Ocuparía la longitud del contenido más 4 bytes.

5. Enumeraciones.

Una enumeración es lista de valores, de tal manera que aquel campo que se defina como una enumeración solo podrá contener como posible valor alguno de los que aparece en la lista o NULL.

Tipo	Descripción.
ENUM('valor1', 'valor2', 'valor3',..)	Puede haber hasta 65535 valores, Solo se permitirá un valor de la lista.
SET('valor1', 'valor2', 'valor3',..)	Puede haber hasta 65 valores, Solo se permitirá un valor de la lista.

Ejemplo: al definir un campo como ENUM ('A', 'B', 'C'), estemos obligando a que ese campo solo puede tener los valores: "A" o "B" o "C" o NULL.

6. Valores fuera de rango.

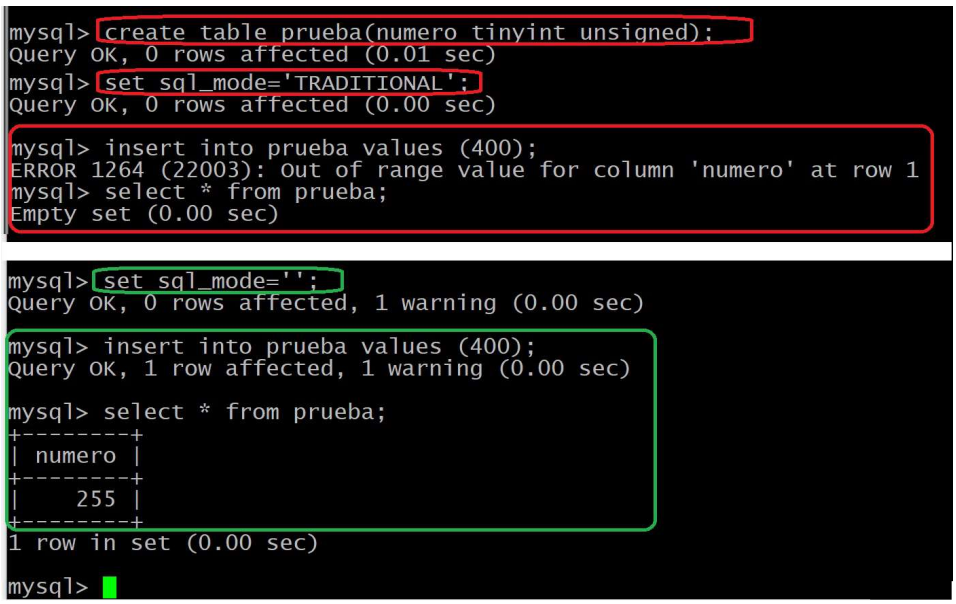
Al trabajar con una base de datos puede ocurrir que al realizar algún cálculo el resultado esté fuera del rango del tipo de dato definido en el campo donde se almacenará. Ante esta situación pueden ocurrir dos cosas: que nos de error y no se almacene; o que se almacene el valor del tipo de dato más cercano al resultado del cálculo. Pero nunca se almacenará el resultado pues está fuera del rango del tipo de dato.

MySQL reaccionará según el valor de la variable de entorno sql_mode:

SET sql_mode = 'TRADITIONAL'; # Nos dará error y no almacerá nada.

SET sql_mode = ''; # Avisará de un valor fuera de rango y almacena el valor más cercano.

Supongamos un campo definido como TINYINT UNSIGNED, cuyo rango es [0, 255] y se intenta almacenar en el valor 400.



La variable sql_mode se puede configurar a nivel de sesión o a nivel global. A nivel de sesión solo le afecta al usuario que la configura y durante su sesión. A nivel global se ha de cambiar con permiso de root y le afecta a todos los usuarios. El formato sería respectivamente:

- SET GLOBAL sql_mode = 'modo';
- SET SESSION sql_mode = 'modo';

Donde modo puede ser:

- ANSI: Comprueba la compatibilidad de los datos y si son mayores trunca o pone el valor más cercano. Genera un aviso.
- STRICT_TRANS_TABLES: Comprueba la compatibilidad y si no hay compatibilidad aborta la acción. Genera mensaje de error. Por defecto a partir de MySQL 5.7.5
- TRADITIONAL: Comprueba la compatibilidad y si no hay aborta la acción. Genera mensaje de error.

La variable sql_mode se puede consultar:

SELECT @@GLOBAL.sql_mode;
SELECT @@SESSION.sql_mode;