Tipos de Datos en MySql

¿Por qué necesitamos conocer los tipos de datos que soporta MySql?

Porque queremos crear tablas:

```
CREATE TABLE Nombre_Tabla ( Nombre_Campo1 TipoDeDato, Nombre_Campo2 TipoDeDato, Nombre_Campo3 TipoDeDato)
```

Necesitamos saber qué tipos de datos soporta MySql. ¿Soporta números decimales? ¿Y binarios? ¿Caracteres? ¿Qué tipo de caracteres?

Criterios para elegir un tipo de dato

Los criterios básicos son:

- Tipos apropiados: que se ajusten lo máximo posible a lo especificado por el diccionario de datos
- Tipos ajustados: que utilicen la cantidad menor de espacio de almacenaje posible.
- Tipos completos: capaces de albergar todos los valores posibles del campo.

Categorías

Cada SGBDR tendrá sus propios tipos de datos

MySql consta de 4 categorías:

- Números
- Fechas y horas
- Caracteres
- Binarios

¿Dónde encontrar TODA la información? Aquí

Tipos Numéricos

De cada uno hay que considerar:

- Rango: conjunto de valores que abarca cada tipo
- Cantidad de espacio en memoria
- Precisión y escala usada

Hay cuatro clases de números según las consideraciones anteriores:

- Integers: Enteros
- Floating-point: números racionales en punto flotante
- Fixed-point: números racionales con número exacto de decimales
- BIT: conjunto de bits (entre 1 y 64)

ENTEROS

Tipo	Almacenamiento
TINYINT	1byte
SMALLINT	2bytes
MEDIUMINT	3bytes
INT	4bytes
BIGINT	8bytes

ENTEROS

NOTA: los corchetes indican opcionalidad

- SINTAXIS: INT [(M)] [<Atributo>]
  - M es opcional. Indica ancho mostrado. No tiene ninguna relación con el rango del entero. Es una opción solo requerida con el atributo ZEROFILL (esta opción obliga a MySql a definir el entero como UNSIGNED)
  - Opcionalmente podemos añadir uno o los dos de los siguientes atributos:
    - UNSIGNED: sin signo. Valores positivos.
    - ZEROFILL: rellena con ceros a la izquierda
  - Valor por defecto será cero.
- Ejemplo:
  - Nombre_Campo1 INT → Este campo contendrá valores enteros. Cada valor ocupará 4bytes
  - Nombre_Campo2 INT UNSIGNED → Este campo contendrá valores enteros positivos. Cada valor ocupará 4 bytes

ENTEROS

- +Ejemplos:
  - Siguiente instrucción crea una tabla con dos campos numéricos. El primero podrá albergar cualquier valor entero entre 0 y 255 y el segundo entre -128 y 127

```
CREATE TABLE Alumno ( Edad_Alumno tinyint unsigned, Valoración tinyint);
```

Problema Tipos.1

- Crea una tabla con los siguientes campos. A continuación inserta algunos valores y realiza una consulta para observar como nos muestra el sistema dichos valores.
  - Campo1: entero que oscila entre 0 y 30000
  - Campo2: entero que oscila entre 0 y 50000 y, si el número ocupa menos de 5 posiciones, el sistema rellena las posiciones necesarias con ceros para mostrar 5 dígitos del número.
  - Campo3: entero que oscila entre -300000 y 3000000

En coma flotante

- Usado para valores aproximados
- Dos tipos:
  - FLOAT: muy pequeño número decimal
  - DOUBLE: pequeño número decimal
- **SINTAXIS: FLOAT [(M,D)] [UNSIGNED] [ZEROFILL]**
  - M indica número máximo de dígitos. Si es menor de 25 (dígitos) utilizará 4 bytes (a grosso modo) para su almacenaje y si es mayor utilizará 8 bytes.
  - D indica número de decimales
  - Si M y D se omiten se utilizará el límite de cifras marcada por el hardware
  - Valor por defecto es NULL y, si el campo se define como NOT NULL, será 0.

Con Punto Fijo

- Usado para valores exactos
- DECIMAL | NUMERIC |DEC |FIXED → todos lo mismo. Completamente equivalentes.
- **SINTAXIS: DECIMAL (M,D) [UNSIGNED] [ZEROFILL]**
  - M indica número de dígitos
  - D número de dígitos detrás del punto
  - Valor por defecto NULL o 0

BITS

- **SINTAXIS: BIT(M)**
  - M indica número de bits (entre 1 y 64)
  - Se pueden utilizar valores numéricos para asignar valores a los campos de tipo BIT
  - Se puede utilizar la forma b'1001' para asignar valores concretos
- También se pueden utilizar los tipos **BOOL** y **BOOLEAN** para definir booleanos. Aunque no son más que sinónimos de TINYINT.

NOTA: en mysql
- la palabra clave *true* y 1 son equivalentes
- la palabra clave *false* y 0 son equivalentes

Problema Tipos.2

Define una tabla con los siguientes campos. Rellénala con datos y realiza una consulta para observar como nos muestra dichos datos el sistema.

- Campo1: número en coma flotante con 10 dígitos de los que 5 son decimales
- Campo2: número con punto fijo con 10 dígitos de los que 5 son decimales
- Campo3: booleano
- Campo4: número binario de 20 dígitos.

Fechas y horas

TIME

- Pensado para expresar tiempos o duraciones de un día.
- Formato en que es mostrado (configurable): `HH:MM:SS` Es decir: horas: minutos:segundos
- Utiliza 3 bytes
- Rango: (-838:59:59 a 838:59:59)
- Se le puede asignar valores utilizando números o caracteres

YEAR

- Dos opciones: dos dígitos o cuatro
- SINTAXIS: YEAR [(2|4)]
- Por defecto, son 4 dígitos que se muestran con el formato 'YYYY'
- Rango para 4 dígitos: 1901 a 2155
- Rango para 2 dígitos: 1970 a 2069 y se mostrará 'YY' (70 a 69)

DATE

- Rango: Desde '1000-01-01' hasta '9999-12-31'
- Utiliza 3 bytes
- Formato 'YYYY-MM-DD'
- Se asigna valor con numéricos o literales

DATETIME

- Combinación de DATE y TIME
- Utiliza 8 bytes
- Rango: Desde 1000-01-01 00:00:00 hasta 9999-12-31 23:59:59
- Valores asignables mediante números o caracteres

TIMESTAMP

- Lo mismo que DATETIME pero los valores se guardan en UTC y cada vez que se muestran se traducen a la hora local. Es decir, tendrá valores distintos según la zona horario configurada en el servidor.

Problemas Tipos.3

1. Crear tabla con cada uno de los tipos temporales vistos, realizar inserciones y comprobar como muestra los resultados el servidor
2. (Ampliación) Crear una tabla con un TIMESTAMP y un DATETIME. A continuación, cambia la variable time_zone (siguiendo las indicaciones del manual (10.6)) Después realiza alguna consulta para observar cómo cambian los valores.
3. (Ampliación) Crear una tabla con un TIMESTAMP y un número entero. A continuación insertar alguna fila en la tabla. Después, cambiar el valor de algún campo de la fila. Observar los resultados. ¿Cómo se puede evitar este comportamiento?

Caracteres y conjunto de caracteres

- Todo carácter pertenece a un conjunto de caracteres. Cada conjunto se llama CHARACTER SET en MySql. Por ejemplo, el conjunto de caracteres que forman el alfabeto latino pertenecen al CHARACTER SET llamado *latin1*.
- Un conjunto de caracteres es una ristra de caracteres en la que todos ellos pertenecen al mismo CHARSET (abreviación de CHARACTER SET)
- Hay dos clases de tipos en este apartado:
  - Texto: alberga caracteres sin formato
  - Entero: alberga caracteres estructurados

Texto

CHAR:

- Almacena los caracteres utilizando un número fijo de bytes. (Se rellena con espacios si es necesario)
- Sintaxis: char (M)
  - Donde M es la longitud fija del tipo. La longitud máxima posible es 255.

VARCHAR:

- Almacena conjunto de caracteres de longitud variable. Es decir, en este caso, a diferencia del anterior, si que habrá un final de ristra (EOF).
- Sintaxis: varchar (M) Donde M es la longitud máxima que puede alcanzar el dato almacenado. El valor máximo posible que puede tener la longitud depende del CHARSET y de la longitud máxima de la fila en la tabla en la que está definido el campo.
- El espacio ocupado por un varchar depende por tanto de la longitud del *string* que contiene y no de la longitud máxima que pueda alcanzar.

TEXT

- Aquí realmente tenemos un conjunto de tipos que son:
  - TINYTEXT (máxima longitud 255 bytes)
  - TEXT (65535 bytes)
  - MEDIUMTEXT (16777215 bytes)
  - LONGTEXT ($4*10^9$ bytes).

CARACTERES ESTRUCTURADOS

ENUM

- Es una lista de literales asociada a un campo. El valor del campo siempre será uno de los elementos de la lista.
- El número máximo de elementos que puede contener la lista es 65535
- Los elementos son almacenados internamente como enteros.
- Sintaxis: enum('Elemento1'[,...])

SET

- Conjunto de literales asociado a un campo. El valor del campo será un subconjunto de uno o más elementos del conjunto.
- El número máximo de miembros distintos que puede tener un SET es de 64.
- Sintaxis: SET ('Elemento1'[,'Elemento2'...])

Problemas Tipos.4

- Mira en la bbdd world diferentes campos definidos con los tipos de carácter
- Crea una tabla con diferentes tipos (por ejemplo: char, varchar, text, enum, set) y realiza varias inserciones. ¿Cómo las muestra el sistema?

CHARACTER SET y COLLATION

- Conviene detenerse en esto un poco puesto que es uno de los puntos fuertes de MySql: su capacidad para adaptarse a todas las latitudes.
- Como hemos dicho un CHARACTER SET o CHARSET es un conjunto de caracteres. Técnicamente es la codificación de un conjunto de caracteres.
- Por ejemplo, podríamos definir un alfabeto de cuatro elementos: {A,a,B,b}. El CHARSET de este alfabeto sería la asociación de un conjunto de bits con la grafía correspondiente:

A	01
a	10
B	11
b	00

- Un COLLATION es un conjunto de reglas que definen:
  - El orden de un CHARSET. Es decir, qué letras van primero y cuáles después.
  - Como se deben comparar los caracteres de un CHARSET
- Siguiendo con el ejemplo anterior, el COLLATION de nuestro particular CHARSET podría dictaminar si la grafía 'A' es o no igual a 'a' y viceversa.
- En las definiciones de un campo de tipo literal podemos incluir el CHARSET y el COLLATION que utilizará dicho CHARSET.

```
CREATE TABLE tabla (

    campo1 VARCHAR(20) CHARACTER SET utf8,

    campo2 TEXT CHARACTER SET latin1 COLLATE latin1_general_cs)
```

Problema Tipos.5

Crea una tabla con distintos CHARSET y COLLATIONS (mira el manual para averiguar los diferentes tipos de CHARSET soportados). Realiza inserciones de diferentes caracteres. A continuación, realiza alguna consulta para ver como muestra el sistema los datos introducidos.

TIPOS BINARIOS

- Almacenan secuencias de bytes. Muy parecidos a los conjuntos de caracteres.
- Pueden almacenar binarios que representan datos (video, ejecutables, etc.)
  - BINARY: de longitud fija
  - VARBINARY: de longitud variable
  - TINYBLOB, BLOB, MEDIUMBLOB, LONGBLOB

NULL

- Tipo especial de valor no perteneciente a ningún tipo
- Permite valores indefinidos para los campos de una tabla
- Es aconsejable su uso en los inicios del diseño de una base de datos ya que es probable que encontremos valores para los campos de las tablas no esperados.