MySQL数据类型详解

MySQL数据类型，可以被分为3类：**数值类型**、**日期和时间类型**以及**字符串(字符)类型**

方括号("["和"]")指出可选的类型修饰符的部分

M

指出最大的显示尺寸。最大的合法的显示尺寸是 255 。

D

适用于浮点类型并且指出跟随在十进制小数点后的数码的数量。最大可能的值是30，但是应该不大于M-2。

UNSIGNED

为"无符号"的意思，即为非负数。是MYSQL自定义的类型，非标准SQL。unsigned 属性只针对整型,用途：

1.UNSIGNED 可用来约束数据的范围，例如有些年龄这种值一般能是负数，那么就可以设置一个 UNSIGNED ，这样可以不允许负数插入。

2.可以增加数值范围（相当于把负数那部分加到正数上）。不过少用，不方便移植。

ZEROFILL

填零，即在数字长度不够的数据前面填充0，以达到设定的长度，MySQL中字段设置为该属性时，将为该列自动地增加UNSIGNED属性。

一、**数值类型**

整型：（1个字节是8位）

在MySQL中创建整型字段INT（或其它像**tinyint**）时，可以设定该字段的位数。如**int**(11),**int**(5),如果不指定位数，INT型默认长度为11。字段插入长度与**int**设定的**M**无关，**当插入长度大于设定的M时，字段值不会被截断，还是按照类型的实际精度进行保存。**

整型字段有个**ZEROFILL**属性，在数字长度不够的数据前面填充0，以达到设定的长度。**M**值只当属性为**ZEROFILL**时，才有区别效果。

如表结构为**id1 int(10)**，**id2 int(5)**,对应数据为 第一行为（1,1），第二行为（1111111,1111111），当id1，id2字段设置为**ZEROFILL**属性时，显示数据为第一行为（0000000001,00001），第二行为（0001111111,1111111），不够的位数会以0补齐。

数据类型	大小	M (默认值)	范围（有符号）	范围（无符号）	用途
tinyint [(M)] [UNSIGNED] [ZEROFILL]	1字节	4	-128~127 (-27~27-1)	0 ~ 255 (0 ~ 28-1)	非常小整数值
smallint [(M)] [UNSIGNED] [ZEROFILL]	2字节	6	-32768~32767（-215 ~ 215-1)	0 ~ 65535 (0 ~ 216-1)	较小整数
mediumint [(M)] [UNSIGNED] [ZEROFILL]	3字节	9	-8388608 ~ 8388607 （-223 ~ 223-1）	0 ~ 16777215 (0 ~ 224-1)	中等大小整数
int [(M)] [UNSIGNED] [ZEROFILL]	4字节	11	-2147483648 ~ 2147483647（-231 ~ 231-1）	0 ~ 4294967295 (0 ~ 232-1)	标准整数
integer [(M)] [UNSIGNED] [ZEROFILL]	4字节	11	-2147483648 ~ 2147483647（-231 ~ 231-1）	0 ~ 4294967295 (0 ~ 232-1)	和int相同
bigint [(M)] [UNSIGNED] [ZEROFILL]	8字节	20	-9223372036854775808 ~92233720368547758087 （-263 ~ 263-1）	0 ~ 18446744073709551615 (0 ~ 264-1)	较大整数

浮点型：

(M,D) 表示总共M位，D个小数位，D包含于M中。浮点类型不能是unsigned的。

对每种浮点类型，可指定一个最大的显示尺寸M和小数位数D。M的值应该取1到255。D的值可为0到30，但是不应大于M-2。M和D对float和double都是可选的，但对于decimal是必须的，在选项M和D时，如果省略了它们，则使用缺省值，如果D被省略，它被设置为0。如果M被省掉，它被设置为10。

数据类型	大小	用途
float [(M,D)] [ZEROFILL]	4字节	单精度浮点型，8位精度；参数m只影响显示效果，不影响精度，d却不同，会影响到精度；m是十进制数字的总个数，d是小数点后面的数字个数
double[(M,D)] [ZEROFILL]	8字节	双精度浮点型，16位精度；参数m只影响显示效果，不影响精度，d却不同，会影响到精度
real[(M,D)] [ZEROFILL]	8字节	同double
decimal[(M[,D])] [ZEROFILL]	4字节	decimal(m,d) 定点类型浮点型在数据库中存放的是近似值，而定点类型在数据库中存放的是精确值。参数m是定点类型数字的最大个数（精 度），范围为0~65，d小数点右侧数字的个数，范围为0~30，但不得超过m。对定点数的计算能精确到65位数字。DECIMAL 数据类型用于精度要求非常高的计算中，这种类型允许指定数值的精度和计数方法作为选择参数。精度在这里指为这个值保存的有效数字的总个数，而计数方法表示小数点后数字的位数。比如语句 DECIMAL(7,3) 规定了存储的值不会超过 7 位数字，并且小数点后不超过 3 位
numeric[(M,D)] [ZEROFILL]	4字节	同decimal

DECIMAL 类型不同于FLOAT和DECIMAL，其中DECIMAL 实际是以串存放的。**DECIMAL 可能的最大取值范围与DOUBLE 一样，但**

是其有效的取值范围由M 和D 的值决定。如果改变M 而固定D，则其取值范围将随M 的变大而变大。表2 - 7的前三行说明了这一点。**如果固定M 而改变D，则其取值范围将随D 的变大而变小（但精度增加）**。表2 - 7的后三行说明了这一点。

给定的DECIMAL 类型的取值范围取决于MySQL数据类型的版本。对于MySQL3.23 以前的版本，DECIMAL(M, D) 列的每个值占用M 字节，而符号（如果需要）和小数点包括在M 字节中。因此，类型为DECIMAL(5, 2) 的列，其取值范围为-9.99 到9 9 . 9 9，因为它们覆盖了所有可能的5 个字符的值。正如MySQL3.23 一样，DECIMAL 值是根据ANSI 规范进行处理的， ANSI 规范规定DECIMAL(M, D) 必须能够表示M 位数字及D 位小数的任何值。

例如， DECIMAL(5, 2) 必须能够表示从-999.99 到999.99 的所有值。而且必须存储符号和小数点，因此自MySQL3.23以来DECIMAL值占**M + 2** 个字节。对于DECIMAL(5, 2)，"最长"的值（- 9 9 9 . 9 9）需要7个字节。在正取值范围的一端，不需要正号，因此MySQL数据类型利用它扩充了取值范围，使其超过了ANSI 所规范所要求的取值范围。如DECIMAL(5, 2) 的最大值为9 9 9 9 . 9 9，因为有7 个字节可用。

简而言之，在MySQL3.23 及以后的版本中，DECIMAL(M, D) 的取值范围等于更早版本中的DECIMAL(M + 2, D) 的取值范围。在MySQL数据类型的所有版本中，如果某个DECIMAL 列的D 为0，则不存储小数点。这样做的结果是扩充了列的取值范围，因为过去用来存储小数点的字节现在可用来存放其他数字了。

二、字符类型

对于可变长的字符类型，其值所占的存储量是不同的，这取决于实际存放在列中的值的长度，这个长度用**L**表示。

数据类型	大小（范围）	用途
char(M)[BINARY]	M个字节，0 <= M <= 255 (L为固定的=255，不够补空格)	定长字符串；CHAR 类型可以使用 BINARY 修饰符。当用于比较运算时，这个修饰符使 CHAR 以二进制方式参于运算，而不是以传统的区分大小写的方式。CHAR值根据缺省字符集以大小写不区分的方式排序和比较，除非给出BINARY关键词
varchar(M) [BINARY]	L+1个字节，其中L <= M 且0 <= M <= 65535（MySQL5.0之前都是最大255）	变长字符串；VARCHAR 类型在使用 BINARY 修饰符时与 CHAR 类型完全相同
tinyblob，tinytext	L+1个字节，其中L < 28-1 (255)	tinyblob：不超过 255 个字符的二进制字符串；tinytext：短文本字符串
blob,text	L+2个字节，其中L < 216-1 (65535)	blob：二进制形式的长文本数据，在分类和比较时BLOB 类型区分大小写;text：长文本数据，在分类和比较时TEXT 不区分大小写
mediumblob,mediumtext	L+3个字节，其中L < 224-1	mediumblob：二进制形式的中等长度文本数据；mediumtext：中等长度文本数据，
longblob,longtext	L+4个字节，其中L < 232-1	longblob：二进制形式的极大文本数据 ；longtext ：极大文本数据
enum('value1','value2',...)	1或2个字节，取决于枚举值的个数(最多65,535个值)	
set('value1','value2',...)	1、2、3、4或者8个字节，取决于set成员的数目(最多64个成员)	

三、日期类型

MySQL 带有 5 个不同的数据类型可供选择。它们可以被分成简单的日期、时间类型，和混合日期、时间类型。根据要求的精度，子类型在每个分类型中都可以使用，并且 MySQL 带有内置功能可以把多样化的输入格式变为一个标准格式。

类型	大小	范围	格式	用途
YEAR[(2|4)]	1字节	1901/2155	YYYY	年份值
DATE	3字节	'1000-01-01'--'9999-12-31'	YYYY-MM-DD	日期值
TIME	3字节	'-838:59:59'到'838:59:59'	HH:MM:SS	时间值或持续时间
DATETIME	8字节	'1000-01-01 00:00:00'--'9999-12-31 23:59:59'	YYYY-MM-DD HH:MM:SS	混合日期和时间值
TIMESTAMP[(M)]	8字节	1970-01-01 00:00:00/2037 年某时	YYYYMMDD HHMMSS	混合日期和时间值，时间戳

(1)**YEAR**

给YEAR类型赋值可以有三种方法。

第一种是直接插入4位字符串或者4位数字。

第二种是插入2位字符串，这种情况下如果插入'00'~'69'，则相当于插入2000~2069；如果插入'70'~'99'，则相当于插入1970~1999。第二种情况下插入的如果是'0'，则与插入'00'效果相同，都是表示2000年。

第三种是插入2位数字，它与第二种（插入两位字符串）不同之处仅在于：如果插入的是一位数字0，则表示的是0000，而不是2000年。所以在给YEAR类型赋值时，一定要分清0和'0'，虽然两者相差个引号，但实际效果确实相差了2000年。

(2)**DATE**

MySQL是以YYYY-MM-DD格式来显示DATE类型的值，插入数据时，数据可以保持这种格式。另外，MySQL还支持一些不严格的语法格式，分

隔符“-”可以用“@”、“.”等众多富豪来替代。在插入数据时，也可以使用“YY-MM-DD”格式，YY转化成对应的年份的规则与YEAR类型类似。如果我们想插入当前系统的时间，则可以插入CURRENT_DATE或者NOW()。

允许使用字符串或数字把值赋给DATE列。

(3)**TIME**

TIME类型表示为“时：分：秒”，尽管小时范围一般是0~23，但是为了表示某些特殊时间间隔，MySQL将TIME的小时范围扩发了，而且支持负值。对TIME类型赋值，标准格式是"HH：MM：SS"，但不一定非要是这种格式。如果插入的是"D HH：MM：SS"格式，则类似插入了"（D*24+HH）：MM：SS"。比如插入"2 23:50:50"，相当于插入了"71:50:50"。如果插入的是"HH：MM"或"SS"格式，则效果是其他未被表示位的值赋为零值。比如插入"30"，相当于插入了"00:00:30"；如果插入"11:25"，相当于插入了"11:25:00"。

另外也可以插入‘D HH’和‘D HH：MM’，效果按上面的例子可以推理出来了吧。 在MySQl中，对于'HHMMSS'格式，系统能够自动转化为标准格式。如果我们想插入当前系统的时间，则可以插入CURRENT_TIME或者NOW()。

TIME类型允许使用字符串或数字把值赋给TIME列,只占3个字节，如果只是存储时间数据，它最合适了。

需要注意的是，没有冒号分隔符的 TIME 类型值，将会被 MySQL 理解为持续的时间，而不是时间戳。

(4)**DATETIME**

通常用于自动存储包含当前日期和时间的时间戳，并可在需要执行大量数据库事务和需要建立一个调试和审查用途的审计跟踪的应用程序中发挥良好作用。

(5)**TIMESTAMP**

通常用于自动存储包含当前日期和时间的时间戳，并可在需要执行大量数据库事务和需要建立一个调试和审查用途的审计跟踪的应用程序中发挥良好作用。

TIMESTAMP的取值范围比较小，没有DATETIME的取值范围大，因此输入值时一定要保证在TIMESTAMP的范围之内。它的插入也与插入其他日期和时间数据类型类似。

那么TIMESTAMP类型如何插入当前时间？第一，可以使用CURRENT_TIMESTAMP；第二，输入NULL，系统自动输入当前的TIMESTAMP；第三，无任何输入，系统自动输入当前的TIMESTAMP。

另外有很特殊的一点：TIMESTAMP的数值是与时区相关.

MySQL以YYYYMMDDHHMMSS、YYMMDDHHMMSS、YYYYMMDD或YYMMDD格式来显示TIMESTAMP值，取决于是否M是14（或省略)、12、8或6，但是允许你使用字符串或数字把值赋给TIMESTAMP列。一个TIMESTAMP列对于记录一个INSERT或UPDATE操作的日期和时间是有用的，因为如果你不自己给它赋值，它自动地被设置为最近操作的日期和时间。你以可以通过赋给它一个NULL值设置它为当前的日期和时间