数值类型

MySQL 的数值数据类型可以大致划分为两个类别，一个是整数，另一个是浮点数或小数。许多不同的子类型对这些类别中的每一个都是可用的，每个子类型支持不同大小的数据，并且 MySQL 允许我们指定数值字段中的值是否有正负之分或者用零填补。

表列出了各种数值类型以及它们的允许范围和占用的内存空间。

类型	大小	范围（有符号）	范围（无符号）	用途
TINYINT	1 字节	(-128，127)	(0，255)	小整数值
SMALLINT	2 字节	(-32 768，32 767)	(0，65 535)	大整数值
MEDIUMINT	3 字节	(-8 388 608，8 388 607)	(0，16 777 215)	大整数值
INT 或 INTEGER	4 字节	(-2 147 483 648，2 147 483 647)	(0，4 294 967 295)	大整数值
BIGINT	8 字节	(-9 233 372 036 854 775 808，9 223 372 036 854 775 807)	(0，18 446 744 073 709 551 615)	极大整数值
FLOAT	4 字节	(-3.402 823 466 E+38，1.175 494 351 E-38)，0，(1.175 494 351 E-38，3.402 823 466 351 E+38)	0，(1.175 494 351 E-38，3.402 823 466 E+38)	单精度浮点数值
DOUBLE	8 字节	(1.797 693 134 862 315 7 E+308，2.225 073 858 507 201 4 E-308)，0，(2.225 073 858 507 201 4 E-308，1.797 693 134 862 315 7 E+308)	，(2.225 073 858 507 201 4 E-308，1.797 693 134 862 315 7 E+308)	双精度浮点数值
DECIMAL	对 DECIMAL(M,D) ，如果 M>D，为 M+2 否则为 D+2	依赖于 M 和 D 的值	依赖于 M 和 D 的值	小数值

INT 类型

在 MySQL 中支持的 5 个主要整数类型是 TINYINT，SMALLINT，MEDIUMINT，INT 和 BIGINT。这些类型在很大程度上是相同的，只有它们存储的值的大小是不相同的。

MySQL 以一个可选的显示宽度指示器的形式对 SQL 标准进行扩展，这样当从数据库检索一个值时，可以把这个值加长到指定的长度。例如，指定一个字段的类型为 INT(6)，就可以保证所包含数字少于 6 个的值从数据库中检索出来时能够自动地用空格填充。需要注意的是，使用一个宽度指示器不会影响字段的大小和它可以存储的值的范围。

万一我们需要对一个字段存储一个超出许可范围的数字，MySQL 会根据允许范围最接近它的一端截短后再进行存储。还有一个比较特别的地方是，MySQL 会在不合规定的值插入表前自动修改为 0。

UNSIGNED 修饰符规定字段只保存正值。因为不需要保存数字的正、负符号，可以在储时节约一个“位”的空间。从而增大这个字段可以存储的值的范围。

ZEROFILL 修饰符规定 0（不是空格）可以用来真补输出的值。使用这个修饰符可以阻止 MySQL 数据库存储负值。

FLOAT、DOUBLE 和 DECIMAL 类型

MySQL 支持的三个浮点类型是 FLOAT、DOUBLE 和 DECIMAL 类型。FLOAT 数值类型用于表示单精度浮点数值，而 DOUBLE 数值类型用于表示双精度浮点数值。

与整数一样，这些类型也带有附加参数：一个显示宽度指示器和一个小数点指示器。比如语句 FLOAT(7,3) 规定显示的值不会超过 7 位数字，小数点后面带有 3 位数字。

对于小数点后面的位数超过允许范围的值，MySQL 会自动将它四舍五入为最接近它的值，再插入它。

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。

忽略 DECIMAL 数据类型的精度和计数方法修饰符将会使 MySQL 数据库把所有标识为这个数据类型的字段精度设置为 10，计算方法设置为 0。

UNSIGNED 和 ZEROFILL 修饰符也可以被 FLOAT、DOUBLE 和 DECIMAL 数据类型使用。并且效果与 INT 数据类型相同。

字符串类型

MySQL 提供了 8 个基本的字符串类型，可以存储的范围从简单的一个字符到巨大的文本块或二进制字符串数据。

类型	大小	用途
CHAR	0-255 字节	定长字符串
VARCHAR	0-255 字节	变长字符串
TINYBLOB	0-255 字节	不超过 255 个字符的二进制字符串
TINYTEXT	0-255 字节	短文本字符串
BLOB 0-65	535 字节	二进制形式的长文本数据
TEXT	0-65 535 字节	长文本数据
MEDIUMBLOB	0-16 777 215 字节	二进制形式的中等长度文本数据
MEDIUMTEXT	0-16 777 215 字节	中等长度文本数据
LOGNGBLOB	0-4 294 967 295 字节	二进制形式的极大文本数据
LONGTEXT	0-4 294 967 295 字节	极大文本数据

CHAR 和 VARCHAR 类型

CHAR 类型用于定长字符串，并且必须在圆括号内用一个大小修饰符来定义。这个大小修饰符的范围从 0-255。比指定长度大的值将被截短，而比指定长度小的值将会用空格作填补。

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 类型的一个变体是 VARCHAR 类型。它是一种可变长度的字符串类型，并且也必须带有一个范围在 0-255 之间的指示器。CHAR 和 VARCHGAR 不同之处在于 MuSQL 数据库处理这个指示器的方式：CHAR 把这个大小视为值的大小，不长度不足的情况下就用空格补足。而 VARCHAR 类型把它视为最大值并且只使用存储字符串实际需要的长度（增加一个额外字节来存储字符串本身的长度）来存储值。所以短于指示器长度的 VARCHAR 类型不会被空

格填补，但长于指示器的值仍然会被截短。

因为 VARCHAR 类型可以根据实际内容动态改变存储值的长度，所以在不能确定字段需要多少字符时使用 VARCHAR 类型可以大大地节约磁盘空间、提高存储效率。

VARCHAR 类型在使用 BINARY 修饰符时与 CHAR 类型完全相同。

TEXT 和 BLOB 类型

对于字段长度要求超过 255 个的情况下，MySQL 提供了 TEXT 和 BLOB 两种类型。根据存储数据的大小，它们都有不同的子类型。这些大型的数据用于存储文本块或图像、声音文件等二进制数据类型。

TEXT 和 BLOB 类型在分类和比较上存在区别。BLOB 类型区分大小写，而 TEXT 不区分大小写。大小修饰符不用于各种 BLOB 和 TEXT 子类型。比指定类型支持的最大范围大的值将被自动截短。

日期和时间类型

在处理日期和时间类型的值时，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
DATE	3	1000-01-01/9999-12-31	YYYY-MM-DD	日期值
TIME	3	-838:59:59'/'838:59:59'	HH:MM:SS	时间值或持续时间
YEAR	1	1901/2155	YYYY	年份值
DATETIME	8	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	8	1970-01-01 00:00:00/2037 年某时	YYYYMMDD HHMMSS	混合日期和时间值，时间戳

DATE、TIME 和 TEAR 类型

MySQL 用 DATE 和 TEAR 类型存储简单的日期值，使用 TIME 类型存储时间值。这些类型可以描述为字符串或不带分隔符的整数序列。如果描述为字符串，DATE 类型的值应该使用连字号作为分隔符分开，而 TIME 类型的值应该使用冒号作为分隔符分开。

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。

MySQL 还对日期的年份中的两个数字的值，或是 SQL 语句中为 TEAR 类型输入的两个数字进行最大限度的通译。因为所有 TEAR 类型的值必须用 4 个数字存储。MySQL 试图将 2 个数字的年份转换为 4 个数字的值。把在 00-69 范围内的值转换到 2000-2069 范围内。把 70-99 范围内的值转换到 1970-1979 之内。如果 MySQL 自动转换后的值并不符合我们的需要，请输入 4 个数字表示的年份。

DATEYIME 和 TIMESTAMP 类型

除了日期和时间数据类型，MySQL 还支持 DATEYIME 和 TIMESTAMP 这两种混合类型。它们可以把日期和时间作为单个的值进行存储。这两种类型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。

如果我们对 TIMESTAMP 类型的字段没有明确赋值，或是被赋与了 null 值。MySQL 会自动使用系统当前的日期和时间来填充它。

复合类型

MySQL 还支持两种复合数据类型 ENUM 和 SET，它们扩展了 SQL 规范。虽然这些类型在技术上是字符串类型，但是可以被视为不同的数据类型。一个 ENUM 类型只允许从一个集合中取得一个值；而 SET 类型允许从一个集合中取得任意多个值。

ENUM 类型

ENUM 类型因为只允许在集合中取得一个值，有点类似于单选项。在处理相互排拆的数据时容易让人理解，比如人类的性别。ENUM 类型字段可以从集合中取得一个值或使用 null 值，除此之外的输入将会使 MySQL 在这个字段中插入一个空字符串。另外如果插入值的大小写与集合中值的大小写不匹配，MySQL 会自动使用插入值的大小写转换成与集合中大小写一致的值。

ENUM 类型在系统内部可以存储为数字，并且从 1 开始用数字做索引。一个 ENUM 类型最多可以包含 65536 个元素，其中一个元素被 MySQL 保留，用来存储错误信息，这个错误值用索引 0 或者一个空字符串表示。

MySQL 认为 ENUM 类型集合中出现的值是合法输入，除此之外其它任何输入都将失败。这说明通过搜索包含空字符串或对应数字索引为 0 的行就可以很容易地找到错误记录的位置。

SET 类型

SET 类型与 ENUM 类型相似但不相同。SET 类型可以从预定义的集合中取得任意数量的值。并且与 ENUM 类型相同的是任何试图在 SET 类型字段中插入非预定义的值都会使 MySQL 插入一个空字符串。如果插入一个即有合法的元素又有非法的元素的记录，MySQL 将会保留合法的元素，除去非法的元素。

一个 SET 类型最多可以包含 64 项元素。在 SET 元素中值被存储为一个分离的“位”序列，这些“位”表示与它相对应的元素。“位”是创建有序元素集合的一种简单而有效的方式。并且它还去除了重复的元素，所以 SET 类型中不可能包含两个相同的元素。

希望从 SET 类型字段中找出非法的记录只需查找包含空字符串或二进制值为 0 的行。