**MySQL数据类型和常用字段属性总结**

这篇文章主要介绍了MySQL数据类型和常用字段属性总结,本文总结了日期和时间数据类型、数值数据类型、字符串数据类型等,需要的朋友可以参考下

**前言**

好比C++中，定义int类型需要多少字节，定义double类型需要多少字节一样，MySQL对表每个列中的数据也会实行严格控制，这是数据驱动应用程序成功的关键。M前言

好比C++中，定义int类型需要多少字节，定义double类型需要多少字节一样，MySQL对表每个列中的数据也会实行严格控制，这是数据驱动应用程序成功的关键。MySQL提供了一组可以赋给表中各个列的数据类型，每个类型都强制数据满足为该数据类型预先确定的一组规则，例如大小、类型及格式。

这里先总结数据类型。MySQL中的数据类型大的方面来分，可以分为：日期和时间、数值，以及字符串。下面就分开来进行总结。

**日期和时间数据类型**

|  |  |
| --- | --- |
| MySQL数据类型 | 含义 |
| date | 3字节，日期，格式：2014-09-18 |
| time | 3字节，时间，格式：08:42:30 |
| datetime | 8字节，日期时间，格式：2014-09-18 08:42:30 |
| timestamp | 4字节，自动存储记录修改的时间 |
| year | 1字节，年份 |

**数值数据类型**

**整型**

|  |  |
| --- | --- |
| MySQL数据类型 | 含义（有符号） |
| tinyint | 1字节，范围（-128~127） |
| smallint | 2字节，范围（-32768~32767） |
| mediumint | 3字节，范围（-8388608~8388607） |
| int | 4字节，范围（-2147483648~2147483647） |
| bigint | 8字节，范围（+-9.22\*10的18次方） |

上面定义的都是有符号的，当然了，也可以加上unsigned关键字，定义成无符号的类型，那么对应的取值范围就要翻翻了，比如：

tinyint unsigned的取值范围为0~255。

**浮点型**

|  |  |
| --- | --- |
| MySQL数据类型 | 含义 |
| float(m, d) | 4字节，单精度浮点型，m总个数，d小数位 |
| double(m, d) | 8字节，双精度浮点型，m总个数，d小数位 |
| decimal(m, d) | decimal是存储为字符串的浮点数 |

我在MySQL中建立了一个表，有一列为float(5, 3)；做了以下试验：

1.插入123.45678，最后查询得到的结果为99.999；  
2.插入123.456，最后查询结果为99.999；  
3.插入12.34567，最后查询结果为12.346；

所以，在使用浮点型的时候，还是要注意陷阱的，要以插入数据库中的实际结果为准。

**字符串数据类型**

|  |  |
| --- | --- |
| MySQL数据类型 | 含义 |
| char(n) | 固定长度，最多255个字符 |
| varchar(n) | 可变长度，最多65535个字符 |
| tinytext | 可变长度，最多255个字符 |
| text | 可变长度，最多65535个字符 |
| mediumtext | 可变长度，最多2的24次方-1个字符 |
| longtext | 可变长度，最多2的32次方-1个字符 |

1.char（n）和varchar（n）中括号中n代表字符的个数，并不代表字节个数，所以当使用了中文的时候(UTF8)意味着可以插入m个中文，但是实际会占用m\*3个字节。  
2.同时char和varchar最大的区别就在于char不管实际value都会占用n个字符的空间，而varchar只会占用实际字符应该占用的空间+1，并且实际空间+1<=n。  
3.超过char和varchar的n设置后，字符串会被截断。  
4.char的上限为255字节，varchar的上限65535字节，text的上限为65535。  
5.char在存储的时候会截断尾部的空格，varchar和text不会。  
6.varchar会使用1-3个字节来存储长度，text不会。

**其它类型**

1.enum(“member1″, “member2″, … “member65535″)  
enum数据类型就是定义了一种枚举，最多包含65535个不同的成员。当定义了一个enum的列时，该列的值限制为列定义中声明的值。如果列声明包含NULL属性，则NULL将被认为是一个有效值，并且是默认值。如果声明了NOT NULL，则列表的第一个成员是默认值。

2.set(“member”, “member2″, … “member64″)  
set数据类型为指定一组预定义值中的零个或多个值提供了一种方法，这组值最多包括64个成员。值的选择限制为列定义中声明的值。

**数据类型属性**

上面大概总结了MySQL中的数据类型，当然了，上面的总结肯定是不全面的，如果要非常全面的总结这些内容，好几篇文章都不够的。下面就再来总结一些常用的属性。

**1.auto\_increment**

auto\_increment能为新插入的行赋一个唯一的整数标识符。为列赋此属性将为每个新插入的行赋值为上一次插入的ID+1。

MySQL要求将auto\_increment属性用于作为主键的列。此外，每个表只允许有一个auto\_increment列。例如：

代码如下:

id smallint not null auto\_increment primary key

**2.binary**

binary属性只用于char和varchar值。当为列指定了该属性时，将以区分大小写的方式排序。与之相反，忽略binary属性时，将使用不区分大小写的方式排序。例如：

代码如下:

hostname char(25) binary not null

**3.default**

default属性确保在没有任何值可用的情况下，赋予某个常量值，这个值必须是常量，因为MySQL不允许插入函数或表达式值。此外，此属性无法用于BLOB或TEXT列。如果已经为此列指定了NULL属性，没有指定默认值时默认值将为NULL，否则默认值将依赖于字段的数据类型。例如：

代码如下:

subscribed enum('0', '1') not null default '0'

**4.index**

如果所有其他因素都相同，要加速数据库查询，使用索引通常是最重要的一个步骤。索引一个列会为该列创建一个有序的键数组，每个键指向其相应的表行。以后针对输入条件可以搜索这个有序的键数组，与搜索整个未索引的表相比，这将在性能方面得到极大的提升。

代码如下:

create table employees  
(  
id varchar(9) not null,  
firstname varchar(15) not null,  
lastname varchar(25) not null,  
email varchar(45) not null,  
phone varchar(10) not null,  
index lastname(lastname),  
primary key(id)  
);

我们也可以利用MySQL的create index命令在创建表之后增加索引：

代码如下:

create index lastname on employees (lastname(7));

这一次只索引了名字的前7个字符，因为可能不需要其它字母来区分不同的名字。因为使用较小的索引时性能更好，所以应当在实践中尽量使用小的索引。

**5.not null**

如果将一个列定义为not null，将不允许向该列插入null值。建议在重要情况下始终使用not null属性，因为它提供了一个基本验证，确保已经向查询传递了所有必要的值。

**6.null**

为列指定null属性时，该列可以保持为空，而不论行中其它列是否已经被填充。记住，null精确的说法是“无”，而不是空字符串或0。

**7.primary key**

primary key属性用于确保指定行的唯一性。指定为主键的列中，值不能重复，也不能为空。为指定为主键的列赋予auto\_increment属性是很常见的，因为此列不必与行数据有任何关系，而只是作为一个唯一标识符。主键又分为以下两种：

(1)单字段主键

如果输入到数据库中的每行都已经有不可修改的唯一标识符，一般会使用单字段主键。注意，此主键一旦设置就不能再修改。

(2)多字段主键

如果记录中任何一个字段都不可能保证唯一性，就可以使用多字段主键。这时，多个字段联合起来确保唯一性。如果出现这种情况，指定一个auto\_increment整数作为主键是更好的办法。

**8.unique**

被赋予unique属性的列将确保所有值都有不同的值，只是null值可以重复。一般会指定一个列为unique，以确保该列的所有值都不同。例如：

代码如下:

email varchar(45) unique

**9.zerofill**

zerofill属性可用于任何数值类型，用0填充所有剩余字段空间。例如，无符号int的默认宽度是10；因此，当“零填充”的int值为4时，将表示它为0000000004。例如：

代码如下:

orderid int unsigned zerofill not null

总结完毕！！！