

4. 数据类型

4.1 数据类型分类

分类	数据类型	说明
数值类型	<code>BIT(M)</code>	位类型。M指定位数，默认值1，范围1-64
	<code>TINYINT [UNSIGNED]</code>	带符号的范围-128~127，无符号范围0~255. 默认有符号
	<code>BOOL</code>	使用0和1表示真和假
	<code>SMALLINT [UNSIGNED]</code>	带符号是-2^15次方 到 2^15-1, 无符号是2^16-1
	<code>INT [UNSIGNED]</code>	带符号是-2^31次方 到 2^31-1, 无符号是2^32-1
	<code>BIGINT [UNSIGNED]</code>	带符号是-2^63次方 到 2^63-1, 无符号是2^64-1
	<code>FLOAT[(M,D)] [UNSIGNED]</code>	M指定显示长度，d指定小数位数，占用4字节
	<code>DOUBLE[(M,D)][UNSIGNED]</code>	表示比float精度更大的小数，占用空间8字节
	<code>DECIMAL(M,D) [UNSIGNED]</code>	定点数M指定长度，D表示小数点的位数
文本、二进制类型	<code>CHAR(size)</code>	固定长度字符串，最大255
	<code>VARCHAR(SIZE)</code>	可变长度字符串，最大长度65535
	<code>BLOB</code>	二进制数据
	<code>TEXT</code>	大文本，不支持全文索引，不支持默认值
时间日期	<code>DATE/DATETIME/TIMESTAMP</code>	日期类型(yyyy-mm-dd) (yyyy-mm-dd hh:mm:ss) timestamp时间戳
String类型	<code>ENUM</code> 类型	ENUM是一个字符串对象，其值来自表创建时在列规定中显示枚举的一列值
	<code>SET</code> 类型	SET是一个字符串对象，可以有零或多个值，其值来自表创建时规定的允许的一列值。指定包括多个set成员的set列值时各成员之间用逗号间隔开。这样set成员值本身不能包含逗号。

4.2 数值类型

类型	字节	最小值	最大值
		(带符号的/无符号的)	(带符号的/无符号的)
TINYINT	1	-128	127
		0	255
SMALLINT	2	-32768	32767
		0	65535
MEDIUMINT	3	-8388608	8388607
		0	16777215
INT	4	-2147483648	2147483647
		0	4294967295
BIGINT	8	-9223372036854775808	9223372036854775807
		0	18446744073709551615

4.2.1 tinyint类型

数值越界测试：

```
mysql> create table tt1(num tinyint);
Query OK, 0 rows affected (0.02 sec)

mysql> insert into tt1 values(1);
Query OK, 1 row affected (0.00 sec)

mysql> insert into tt1 values(128); -- 越界插入, 报错
ERROR 1264 (22003): Out of range value for column 'num' at row 1
mysql> select * from tt1;
```

```
+-----+
| num |
+-----+
|   1  |
+-----+
1 row in set (0.00 sec)
```

说明:

- 在MySQL中，整型可以指定是有符号的和无符号的，默认是有符号的。
- 可以通过UNSIGNED来说明某个字段是无符号的
- 无符号案例

```
mysql> create table tt2(num tinyint unsigned);
mysql> insert into tt2 values(-1); -- 无符号，范围是： 0 - 255
ERROR 1264 (22003): Out of range value for column 'num' at row 1
mysql> insert into tt2 values(255);
Query OK, 1 row affected (0.02 sec)

mysql> select * from tt2;
+-----+
| num |
+-----+
| 255 |
+-----+
1 row in set (0.00 sec)
```

- 其他类型自己推导

注意：尽量不使用unsigned，对于int类型可能存放不下的数据，int unsigned同样可能存放不下，与其如此，还不如设计时，将int类型提升为bigint类型。

4.2.2 bit类型

基本语法：

bit[(M)] : 位字段类型。M表示每个值的位数，范围从1到64。如果M被忽略，默认为1。

举例：

```
mysql> create table tt4 ( id int, a bit(8));
Query OK, 0 rows affected (0.01 sec)

mysql> insert into tt4 values(10, 10);
Query OK, 1 row affected (0.01 sec)

mysql> select * from tt4; #发现很怪异的现象，a的数据10没有出现
+-----+
| id  | a    |
+-----+
| 10  |      |
+-----+
1 row in set (0.00 sec)
```

bit使用的注意事项：

- bit字段在显示时，是按照ASCII码对应的值显示。

```
mysql> insert into tt4 values(65, 65);
mysql> select * from tt4;
+----+---+
| id | a   |
+----+---+
| 10 |      |
| 65 | A   |
+----+---+
```

- 如果我们有这样的值，只存放0或1，这时可以定义bit(1)。这样可以节省空间。

```
mysql> create table tt5(gender bit(1));
mysql> insert into tt5 values(0);
Query OK, 1 row affected (0.00 sec)

mysql> insert into tt5 values(1);
Query OK, 1 row affected (0.00 sec)

mysql> insert into tt5 values(2); -- 当插入2时，已经越界了
ERROR 1406 (22001): Data too long for column 'gender' at row 1
```

4.2.3 小数类型

4.2.3.1 float

语法：

`float[(m, d)] [unsigned]` : m指定显示长度, d指定小数位数, 占用空间4个字节

案例：

小数：float(4,2)表示的范围是-99.99 ~ 99.99, MySQL在保存值时会进行四舍五入。

```
mysql> create table tt6(id int, salary float(4,2));
Query OK, 0 rows affected (0.01 sec)

mysql> insert into tt6 values(100, -99.99);
Query OK, 1 row affected (0.00 sec)

mysql> insert into tt6 values(101, -99.991); #多的这一点被拿掉了
Query OK, 1 row affected (0.00 sec)

mysql> select * from tt6;
+----+-----+
| id | salary |
+----+-----+
| 100 | -99.99 |
| 101 | -99.99 |
+----+-----+
2 rows in set (0.00 sec)
```

问题：当我们的float(4,2)如果是一个有符号的，则表示范围是-99.99 ~ 99.99，如果float(6,3)，请同学们说说范围是多少？

案例：

如果定义的是float(4,2) unsigned 这时，因为把它指定为无符号的数，范围是 0 ~ 99.99

```
mysql> create table tt7(id int, salary float(4,2) unsigned);
Query OK, 0 rows affected (0.01 sec)

mysql> insert into tt7 values(100, -0.1);
Query OK, 1 row affected, 1 warning (0.00 sec)

mysql> show warnings;
+-----+-----+
| Level | Code | Message
+-----+-----+
| Warning | 1264 | Out of range value for column 'salary' at row 1 |
+-----+-----+
1 row in set (0.00 sec)

mysql> insert into tt7 values(100, -0);
Query OK, 1 row affected (0.00 sec)

mysql> insert into tt7 values(100, 99.99);
Query OK, 1 row affected (0.00 sec)
```

4.2.3.2 decimal

语法：

`decimal(m, d) [unsigned]`：定点数m指定长度，d表示小数点的位数

- decimal(5,2) 表示的范围是 -999.99 ~ 999.99
- decimal(5,2) unsigned 表示的范围 0 ~ 999.99

decimal和float很像，但是有区别：

float和decimal表示的精度不一样

```
mysql> create table tt8 ( id int, salary float(10,8), salary2
decimal(10,8));
mysql> insert into tt8 values(100,23.12345612, 23.12345612);
Query OK, 1 row affected (0.00 sec)

mysql> select * from tt8;
+----+-----+-----+
| id | salary      | salary2      |
+----+-----+-----+
| 100 | 23.12345695 | 23.12345612 | # 发现decimal的精度更准确，因此如果我们希望某个数据表示高精度，选择decimal
+----+-----+-----+
```

说明：float表示的精度大约是7位。

- decimal整数最大位数m为65。支持小数最大位数d是30。如果d被省略，默认为0.如果m被省略，默认是10。

建议：如果希望小数的精度高，推荐使用decimal。

4.3 字符串类型

4.3.1 char

语法：

`char(L)`：固定长度字符串，L是可以存储的长度，单位为字符，最大长度值可以为255

案例（char）：

```
mysql> create table tt9(id int, name char(2));
Query OK, 0 rows affected (0.00 sec)

mysql> insert into tt9 values(100, 'ab');
Query OK, 1 row affected (0.00 sec)

mysql> insert into tt9 values(101, '中国');
Query OK, 1 row affected (0.00 sec)

mysql> select * from tt9;
+----+----+
| id | name |
+----+----+
| 100 | ab   |
| 101 | 中国 |
+----+----+
```

说明：

`char(2)` 表示可以存放两个字符，可以是字母或汉字，但是不能超过2个，最多只能是255

```
mysql> create table tt10(id int ,name char(256));
ERROR 1074 (42000): Column length too big for column 'name' (max = 255); use
BLOB or TEXT instead
```

4.3.2 varchar

语法：

`varchar(L)`：可变长度字符串，L表示字符长度，最大长度65535个字节

案例：

```

mysql> create table tt10(id int ,name varchar(6)); --表示这里可以存放6个字符

mysql> insert into tt10 values(100, 'hello');

mysql> insert into tt10 values(100, '我爱你, 中国');

mysql> select * from tt10;
+----+-----+
| id | name      |
+----+-----+
| 100 | hello     |
| 100 | 我爱你, 中国 |
+----+-----+

```

说明：

关于varchar(len),len到底是多大，这个len值，和表的编码密切相关：

- varchar长度可以指定为0到65535之间的值，但是有1 - 3个字节用于记录数据大小，所以说有效字节数是65532。
- 当我们的表的编码是utf8时，varchar(n)的参数n最大值是 $65532/3=21844$ [因为utf中，一个字符占用3个字节]，如果编码是gbk，varchar(n)的参数n最大是 $65532/2=32766$ (因为在gbk中，一个字符占用2字节)。

```

mysql> create table tt11(name varchar(21845))charset=utf8; --验证了utf8确实是不能超过21844

ERROR 1118 (42000): Row size too large. The maximum row size for the used
table type, not counting BLOBS, is 65535. You have to change some columns to
TEXT or BLOBS

mysql> create table tt11(name varchar(21844)) charset=utf8;
Query OK, 0 rows affected (0.01 sec)

```

4.3.3 char和varchar比较

实际存储	char(4)	varchar(4)	char占用字节	varchar占用字节
abcd	abcd	abcd	$4*3=12$	$4*3+1=13$
A	A	A	$4*3=12$	$1*3+1=4$
Abcde	×	×	数据超过长度	数据超过长度

如何选择定长或变长字符串？

- 如果数据确定长度都一样，就使用定长 (char)，比如：身份证，手机号，md5
- 如果数据长度有变化，就使用变长(varchar)，比如：名字，地址，但是你要保证最长的能存的进去。
- 定长的磁盘空间比较浪费，但是效率高。
- 变长的磁盘空间比较节省，但是效率低。
- 定长的意义是，直接开辟好对应的空间
- 变长的意义是，在不超过自定义范围的情况下，用多少，开辟多少。

4.4 日期和时间类型

常用的日期有如下三个：

- `date`: 日期 'yyyy-mm-dd'，占用三字节
- `datetime` 时间日期格式 'yyyy-mm-dd HH:ii:ss' 表示范围从 1000 到 9999，占用八字节
- `timestamp`: 时间戳，从1970年开始的 yyyy-mm-dd HH:ii:ss 格式和 `datetime` 完全一致，占用四字节

案例：

```
//创建表

mysql> create table birthday (t1 date, t2 datetime, t3 timestamp);
Query OK, 0 rows affected (0.01 sec)

//插入数据:

mysql> insert into birthday(t1,t2) values('1997-7-1','2008-8-8 12:1:1'); --插入两种时间
Query OK, 1 row affected (0.00 sec)

mysql> select * from birthday;
+-----+-----+-----+
| t1   | t2      | t3          |
+-----+-----+-----+
| 1997-07-01 | 2008-08-08 12:01:01 | 2017-11-12 18:28:55 | --添加数据时，时间戳自动补上当前时间
+-----+-----+-----+


//更新数据:

mysql> update birthday set t1='2000-1-1';
Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> select * from birthday;
+-----+-----+-----+
| t1   | t2      | t3          |
+-----+-----+-----+
| 2000-01-01 | 2008-08-08 12:01:01 | 2017-11-12 18:32:09 | -- 更新数据，时间戳会更新成当前时间
+-----+-----+-----+
```

4.5 enum和set

语法：

- `enum`: 枚举，“单选”类型；
`enum('选项1','选项2','选项3',...);`

该设定只是提供了若干个选项的值，最终一个单元格中，实际只存储了其中一个值；而且出于效率考虑，这些值实际存储的是“数字”，因为这些选项的每个选项值依次对应如下数字：1,2,3,...最多65535个；当我们添加枚举值时，也可以添加对应的数字编号。

- `set`: 集合，“多选”类型；
`set('选项值1','选项值2','选项值3', ...);`

该设定只是提供了若干个选项的值，最终一个单元格中，设计可存储了其中任意多个值；而且出于效率考虑，这些值实际存储的是“数字”，因为这些选项的每个选项值依次对应如下数字：1,2,4,8,16,32, 最多64个。

说明：不建议在添加枚举值，集合值的时候采用数字的方式，因为不利于阅读。

案例：

有一个调查表votes，需要调查人的喜好，比如（登山，游泳，篮球，武术）中去选择（可以多选），
（男，女）[单选]

```
mysql> create table votes(
->     username varchar(30),
->     hobby set('登山','游泳','篮球','武术'), --注意：使用数字标识每个爱好的时候，想想
Linux权限，采用比特位位置来个set中的爱好对应起来
->     gender enum('男','女'));      --注意：使用数字标识的时候，就是正常的数组下标
Query OK, 0 rows affected (0.02 sec)
```

插入数据：

```
insert into votes values('雷锋', '登山,武术', '男');
insert into votes values('Juse', '登山,武术', 2);
select * from votes where gender=2;
+-----+-----+-----+
| username | hobby       | gender |
+-----+-----+-----+
| Juse    | 登山,武术   | 女     |
+-----+-----+-----+
```

有如下数据，想查找所有喜欢登山的人：

```
+-----+-----+-----+
| username | hobby       | gender |
+-----+-----+-----+
| 雷锋    | 登山,武术   | 男     |
| Juse    | 登山,武术   | 女     |
| LiLei   | 登山       | 男     |
| LiLei   | 篮球       | 男     |
| HanMeiMei | 游泳     | 女     |
+-----+-----+-----+
```

使用如下查询语句：

```
mysql> select * from votes where hobby='登山';
+-----+-----+-----+
| username | hobby   | gender |
+-----+-----+-----+
| LiLei   | 登山   | 男     |
+-----+-----+-----+
```

不能查询出所有，爱好为登山的人。

集合查询使用find_in_set函数：

`find_in_set(sub,str_list)`：如果 `sub` 在 `str_list` 中，则返回下标；如果不在，返回0；
`str_list` 用逗号分隔的字符串。

```
mysql> select find_in_set('a', 'a,b,c');
+-----+
| find_in_set('a', 'a,b,c') |
+-----+
|          1          |
+-----+  
  
mysql> select find_in_set('d', 'a,b,c');
+-----+
| find_in_set('d', 'a,b,c') |
+-----+
|          0          |
+-----+
```

查询爱好登山的人：

```
mysql> select * from votes where find_in_set('登山', hobby);
+-----+-----+-----+
| username | hobby      | gender |
+-----+-----+-----+
| 雷锋     | 登山,武术   | 男     |
| Juse     | 登山,武术   | 女     |
| LiLei    | 登山       | 男     |
+-----+-----+-----+
```