第5章 数据类型与表的约束

CHAPTER

学习目标：

- 掌握 MySQL 中的各种数据类型；
- 掌握 MySQL 中表的各种约束。

在数据库中，数据表用来组织和保存数据，它由表结构和表中数据构成。在设计表结构时，经常需要根据实际需求，选择合适的数据类型和约束。本章介绍数据类型和表中的各种约束。

5.1 数据类型

使用 MySQL 数据库存储数据时，不同的数据类型决定了 MySQL 存储数据方式的不同。MySQL 数据库提供了多种数据类型，本节对这些数据类型进行讲解。

5.1.1 数字类型

在数据表中，经常需要存储一些数字，如学生的年龄、商品的价格等，这需要数字类型进行存储。数字类型包括整数类型、浮点数类型、定点数类型、bit(位)类型等。

1. 整数类型

MySQL 提供的整数类型包括 tinyint、smallint、mediumint、int 和 bigint。整数类型的属性字段可以添加 auto_increment 自增约束条件。不同整数类型所对应的字节大小和取值范围不同，如表 5-1 所示。

表 5-1 整数类型

数据类型	字节数	无符号取值范围	有符号取值范围
tinyint	1	$0\sim2^{8}-1$	$-2^{7}\sim2^{7}-1$
smallint	2	$0\sim2^{16}-1$	$-2^{15}\sim2^{15}-1$
mediumint	3	$0\sim2^{24}-1$	$-2^{23}\sim2^{23}-1$
int	4	$0\sim2^{32}-1$	$-2^{31}\sim2^{31}-1$
bigint	8	$0\sim2^{64}-1$	$-2^{63}\sim2^{63}-1$

从表 5-1 中可以看出，不同整数类型所占用的字节数和取值范围都是不同的。其中，占用字节数最小的是 tinyint，占用字节数最大的是 bigint。不同整数类型的取值范围可以根据字节数计算出来，如 tinyint 类型的整数占用 1 字节，1 字节是 8 位，那么 tinyint 类型无符号数的最大值就是 2^8-1（即 255），有符号数的最大值就 2^7-1（即 127）。同理，可以算出其他不同整数类型的取值范围。

需要注意的是，若使用无符号数据类型，需要在数据类型右边加上 unsigned 关键字来修饰，如 int unsigned 表示无符号 int 类型。

下面通过案例来演示整数类型的使用。

【例 5-1】 创建名称为 int_test 的数据表，字段类型用 int 和 int unsigned 来定义。具体 SQL 语句与执行结果如下。

```
mysql> create table int_test(
    -> i1 int,
    -> i2 int unsigned
    -> );
Query OK, 0 rows affected (0.63 sec)
```

在数据表 int_test 中，i1 是有符号类型，i2 是无符号类型。

【例 5-2】 插入整型数据。当数据在合法范围内，可以插入，反之提示错误信息。具体 SQL 语句与执行结果如下。

```
mysql> insert into int_test values(500,500);
Query OK, 1 row affected (0.20 sec)
mysql> insert into int_test values(-500,-500);
ERROR 1264 (22003): Out of range value for column 'i2' at row 1
```

从以上执行结果可以看出，－500 超出了无符号 int 类型 i2 的取值范围，插入失败，MySQL 显示了错误信息。

2. 浮点数类型

在 MySQL 中，存储的小数都是使用浮点数或定点数来表示的。浮点数的类型有两种，分别是单精度浮点数（float）和双精度浮点数（double），对应的字节大小及其取值范围如表 5-2 所示。

表 5-2 浮点数类型

数据类型	字节数	负数取值范围	非负数取值范围
float	4	－3.402823466E＋38～ －1.175494351E－38	0 和 1.175494351E－38～ 3.402823466E＋38
double	8	－1.7976931348623157E＋308～ －2.2250738585072014E－308	0 和 2.2250738585072014E－308～ 1.7976931348623157E＋308

表 5-2 中列举的取值范围是理论上的极限值，但根据不同的硬件或操作系统，实际范围可能会小。另外，当浮点数类型使用 unsigned 修饰为无符号时，取值范围将不包含负数。

浮点数类型虽然取值范围很大，但是精度并不高。float 的精度大约是 6 位，double 的精度大约是 15 位。如果超出精度，可能会导致给定的数值与实际保存的数值不一致，发生精度损失。

为了更好地理解，下面通过案例演示浮点数类型的使用。

【例 5-3】 创建名称为 float_test 的数据表，具体 SQL 语句与执行结果如下。

```
mysql> create table float_test(
    -> f1 float,
    -> f2 float);
Query OK, 0 rows affected (0.27 sec)
```

数据表创建成功。

【例 5-4】 插入 float 类型的数据，具体 SQL 语句与执行结果如下。

```
mysql> insert into float_test values(1234567,1.234567);
Query OK, 1 row affected (0.04 sec)
```

数据插入成功后，可以使用 select 语句查询表中的数据，执行结果如下。

```
mysql> select * from float_test;
+----------+----------+
| f1       | f2       |
+----------+----------+
| 1234570  | 1.23457  |
+----------+----------+
1 row in set (0.00 sec)
```

从以上结果中可以看出，当一个数字的整数部分和小数部分加起来达到 7 位时，第 7 位就会被四舍五入。

3. 定点数类型

定点数类型(decimal)通过 decimal(M,D)设置位数和精度，其中 M 表示数值总位数(不包括“.”和“－”)，最大值为 65，默认值为 10；D 表示小数点后的位数，最大值为 30，默认值为 0。例如，decimal(6,3)表示的取值范围是－999.999～999.999。系统会自动根据存储的数据来分配存储空间。如果不允许保存负数，可以通过 unsigned 修饰。

为了更好地理解，下面通过案例演示定点数类型的使用。

【例 5-5】 创建名称为 decimal_test 的数据表，具体 SQL 语句与执行结果如下。

```
mysql> create table decimal_test(
    -> d1 decimal(5,2),
    -> d2 decimal(5,2));
Query OK, 0 rows affected (0.60 sec)
```

数据表创建成功。

【例 5-6】 插入一条数据(123.124,123.125),具体 SQL 语句与执行结果如下。

```
mysql> insert into decimal_test values(123.124,123.125);
Query OK, 1 row affected, 2 warnings (0.05 sec)
```

数据插入成功后,出现了两条警告信息,查看警告信息,结果如下。

```
mysql> show warnings;
+-------+------+-----------------------------------------+
| Level | Code | Message                                 |
+-------+------+-----------------------------------------+
| Note  | 1265 | Data truncated for column 'd1' at row 1 |
| Note  | 1265 | Data truncated for column 'd2' at row 1 |
+-------+------+-----------------------------------------+
2 rows in set (0.04 sec)
```

警告信息显示出现了数据截断,可以使用 select 语句查询表中数据,执行结果如下。

```
mysql> select * from decimal_test;
+--------+--------+
| d1     | d2     |
+--------+--------+
| 123.12 | 123.13 |
+--------+--------+
1 row in set (0.00 sec)
```

从以上执行结果可以看出,小数部分被四舍五入。

【例 5-7】 插入一条数据(999.99,999.999),具体 SQL 语句与执行结果如下。

```
mysql> insert into decimal_test values(999.99,999.999);
ERROR 1264 (22003): out of range value for column 'd2' at row 1
```

从以上执行结果可以看到,数据插入失败。

从例 5-6 和例 5-7 可以看出,若小数部分超出范围,会进行四舍五入,并出现数据截断(data truncated)警告;若整数部分超出范围,数据会插入失败,提示超出取值范围(out of range value)错误。

注意:浮点数类型也可以设置位数和精度,但仍有可能损失精度,在实际使用过程中应避免使用浮点数类型,以免出现不能人为控制的问题。因此,对于小数类型的数据,建议使用定点数类型。

4. bit(位)类型

bit(位)类型用于存储二进制数据,语法为 bit(M),M 表示位数,范围为 1～64。

为了更好地理解,下面通过案例演示 bit 类型的使用。

【例 5-8】 以保存字符 A 为例,A 的 ASCII 码对应十进制 65,对应二进制 1000001,总共有 7 位,因此需要 bit(7)保存。具体 SQL 语句与执行结果如下。

```
mysql> select ASCII('A');
+-------------+
| ASCII('A')  |
+-------------+
|        65   |
+-------------+
1 row in set (0.05 sec)
```

获取字符 A 的 ASCII 码，结果是 65。

```
mysql> select bin(65),length(bin(65));
+---------+------------------+
| bin(65) | length(bin(65))  |
+---------+------------------+
| 1000001 |              7   |
+---------+------------------+
1 row in set (0.10 sec)
```

获取字符 A 的二进制数，并计算长度，结果为 1000001 和 7。

```
mysql> create table bit_test(b bit(7));
Query OK, 0 rows affected (0.25 sec)
```

数据表创建成功。

```
mysql> insert into bit_test values(65);
Query OK, 1 row affected (0.59 sec)
```

插入数据成功。

```
mysql> select bin(b) from bit_test;
+----------+
| bin(b)   |
+----------+
| 1000001  |
+----------+
1 row in set (0.00 sec)
```

查询数据并转换为二进制数显示，结果为 1000001。

从以上执行结果可以看出，利用 MySQL 中的 ASCII()、bin()、length()函数可以方便地查询 ASCII 码、二进制值和数字长度。bit 类型字段在数字插入时转换为二进制保存，但在利用 select 查询时，可以转换为对应的字符显示。

5.1.2　时间和日期类型

在处理日期和时间类型的值时，MySQL 有不同的数据类型供用户选择。它们可以被

分为简单的日期和时间类型、混合的日期和时间类型。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，可以将多样化的输入格式变为一个标准格式。日期和时间类型同样有对应的字节数和取值范围，如表 5-3 所示。

表 5-3 日期和时间类型

数据类型	字节数	取值范围	日期格式	零值
year	1	1901～2155	YYYY	0000
date	4	1000-01-01～9999-12-31	YYYY-MM-DD	0000-00-00
time	3	－838:59:59～838:59:59	HH:MM:SS	00:00:00
datetime	8	1000-01-01 00:00:00～ 9999-12-31 23:59:59	YYYY-MM-DD HH:MM:SS	0000-00-00 00:00:00
timestamp	4	1970-01-01 00:00:01～ 2038-01-19 03:14:07	YYYY-MM-DD HH:MM:SS	0000-00-00 00:00:00

在表 5-3 中，日期格式 YYYY 表示年，MM 表示月，DD 表示日。每种日期和时间类型的取值范围都是不同的。需要注意的是，如果插入的数值不合法，系统会自动将对应的零值插入数据表中。

下面详细讲解日期和时间类型。

1. year 类型

year 类型用于年份。

【例 5-9】 创建名称为 year_test 的数据表，并插入数据。具体 SQL 语句与执行结果如下。

```
mysql> create table year_test(y year);
Query OK, 0 rows affected (0.17 sec)
mysql> insert into year_test values(2022);
Query OK, 1 row affected (0.14 sec)
```

在 MySQL 中，可以使用以下 3 种格式指定 year 类型的值。

(1) 使用 4 位字符串或数字表示，为'1901'～'2155'或 1901～2155。例如输入'2022'或 2022，插入数据库中的值均为 2022。

(2) 使用两位字符串表示，为'00'～'99'，其中，'00'～'69'的值会被转换为 2000～2069 的 year 值，'70'～'99'的值会被转换为 1970～1999 的 year 值。例如输入'22'，插入数据表中的值为 2022。

(3) 使用两位数字表示，为 1～99，其中，1～69 的值会被转换为 2001～2069 的 year 值，70～99 的值会被转换为 1970～1999 的 year 值。例如输入 22，插入数据表中的值为 2022。

需要注意的是，当使用 year 类型时，一定要区分'0'和 0。因为字符串格式'0'表示的 year 值是 2000，而数字格式的 0 表示的 year 值是 0000。

2. date 类型

date 类型用于表示日期值，不包含时间部分。

【例 5-10】 创建名称为 date_test 的数据表，并插入数据。具体 SQL 语句与执行结果如下。

```
mysql> create table date_test(d date);
Query OK, 0 rows affected (0.19 sec)
mysql> insert into date_test values('2022-04-03');
Query OK, 1 row affected (0.15 sec)
```

在 MySQL 中，可以使用以下 4 种格式指定 date 类型的值。

(1) 以'YYYY-MM-DD'或者'YYYYMMDD'字符串格式表示。例如，输入'2022-04-03'或'20220403'，插入数据库中的日期都为 2022-04-03。

(2) 以'YY-MM-DD'或者'YYMMDD'字符串格式表示。YY 表示的是年，为'00'～ '99'，其中，'00'～'69'的值会被转换为 2000～2069 的 year 值，'70'～'99'的值会被转换为 1970～1999 的 year 值。例如输入'22-04-03'或'220403'，插入数据库中的日期都为 2022-04-03。

(3) 以 YY-MM-DD 或者 YYMMDD 数字格式表示。例如输入 22-04-03 或 220403，插入数据库中的日期都为 2022-04-03。

(4) 使用 current_date 输入当前系统日期。

3. time 类型

time 类型用于表示时间值，它的显示形式一般为 HH:MM:SS，其中 HH 表示小时，MM 表示分，SS 表示秒。在 MySQL 中，可以使用以下 3 种格式指定 time 类型的值。

(1) 以'HHMMSS'字符串或者 HHMMSS 数字格式表示。例如输入'121212'或 121212，插入数据库中的时间为 12:12:12。

(2) 以'D HH:MM:SS'字符串格式表示。其中，D 表示日，可以取 0～34 的值，插入数据时，小时的值等于(D×24＋HH)。例如，输入'2 11:30:50'，插入数据库中的时间为 59:30:50；输入'11:30:50'，插入数据库中的时间为'11:30:50'；输入'30 22:59:59'，插入数据库中的时间为 742:59:59。

(3) 使用 current_time 输入当前系统时间。

4. datetime 类型

datetime 类型用于表示日期和时间，它的显示形式为'YYYY-MM-DD HH:MM:SS'，其中 YYYY 表示年，MM 表示月，DD 表示日，HH 表示小时，MM 表示分，SS 表示秒。在 MySQL 中可以使用以下 4 种格式指定 datetime 类型的值。

(1) 以'YYYY-MM-DD HH:MM:SS'或者'YYYYMMDDHHMMSS'字符串格式表示的日期和时间，取值范围为'1000-01-01 00:00:00'～'9999-12-31 23:59:59'。例如，输入'2014-01-22 09:01:23'或'20140122090123'，插入数据库中的 datetime 值都为 2014-01-22 09:01:23。

(2) 以'YY-MM-DD HH:MM:SS'或者'YYMMDDHHMMSS'字符串格式表示的日期和时间，其中 YY 表示年，取值范围为'00'～ '99'，与 date 类型中的 YY 相同，'00'～'69'的值会被转换为 2000～2069 的 year 值，'70'～'99'的值会被转换为 1970～1999 的 year 值。

(3) 以 YYYYMMDDHHMMSS 或者 YYMMDDHHMMSS 数字格式表示的日期和时间。例如，插入 20140122090123 或者 140122090123，插入数据库中的 datetime 值都为 2014-01-22 09:01:23。

(4) 使用 now()来输入当前系统的日期和时间。

5. timestamp 类型

timestamp(时间戳)类型用于表示日期和时间,它的显示形式与 datetime 相同,但取值范围比 datetime 小,下面介绍几种 timestamp 类型与 datetime 类型不同的形式,具体如下。

(1) 使用 current_timestamp 来输入系统当前日期和时间。

(2) 无任何输入,或输入 null 时,实际保存的是系统当前日期和时间。

注意:在 MySQL 中,timestamp 字段默认情况下会自动设置 not null default current_timestamp on update current timestamp 属性,具体解释如下。

(1) not null 表示非空约束,该字段将不允许保存 null 值。

(2) default 表示默认约束,当字段无任何输入时,自动设置某个值作为默认值。此处设为 current_timestamp 表示使用系统当前日期和时间作为默认值。

(3) on update 用于当一条记录中的其他字段被 update 语句修改时,自动更改该字段为某个值。此处设为 current_timestamp 表示每次修改时保存修改时的系统日期和时间。

若为 timestamp 字段手动设置 default 属性时,该字段将不会自动设置 on update 属性。

5.1.3 字符串类型

字符串类型用来存储字符串数据,除了可以存储字符串数据之外,还可以存储其他数据,如图片和声音的二进制数据。MySQL 支持两类字符型数据:文本字符串和二进制字符串。文本字符串类型包括 char、varchar、text、enum 和 set,二进制字符串类型包括 binary、varbinary 和 blob。

1. char 和 varchar 类型

char 和 varchar 类型的语法格式如下。

```
char(M)
```

或

```
varchar(M)
```

char(M)为固定长度字符串,在定义时指定字符串列长。不足指定长度时,在右侧填充空格,以达到指定的长度。M 表示列长度,M 的范围是 0～255 个字符。例如 char(4)定义了一个固定长度的字符串列,其包含的字符个数最大是 4。当检索到 char 值时,尾部的空格将被删除。

varchar(M)是长度可变的字符串,M 表示最大列长度。M 的范围是 0～65535。varchar 的最大实际长度由最长的行的大小和使用的字符集确定,而其实际占用的空间为字符串的实际长度加 1。例如,varchar(50)定义了一个最大长度为 50 的字符串,如果插入的字符串只有 10 个字符,则实际存储的字符串为 10 个字符和一个字符串结束字符。varchar 在保存和检索值时尾部的空格仍保留。

为了对比 char 和 varchar 之间的区别,下面以 char(4)和 varchar(4)为例进行说明,如表 5-4 所示。

表 5-4　char(4)和 varchar(4)的对比

插　入　值	char(4)存储需求	varchar(4) 存储需求
''	4 字节	1 字节
'ab'	4 字节	3 字节
'abc'	4 字节	4 字节
'abcd'	4 字节	5 字节

从对比结果可以看出，char(4)定义了固定长度为 4 的列，不管存储的数据长度为多少，所占用的空间均为 4 字节；varchar(4)定义的列所占的字节数为实际长度加 1。

2. text 类型

text 类型保存大文本数据，如文章内容、评论等。当保存或查询 text 列的值时，不删除尾部空格。text 类型分为 4 种，不同的 text 类型的存储范围和数据长度不同，如表 5-5 所示。

表 5-5　text 类型

数 据 类 型	存 储 范 围	数 据 类 型	存 储 范 围
tinytext	$0\sim2^8-1$ 字节	mediumtext	$0\sim2^{24}-1$ 字节
text	$0\sim2^{16}-1$ 字节	longtext	$0\sim2^{32}-1$ 字节

text 类型所保存的最大字符数量取决于字符串实际占用的字节数。

注意：在使用“=”等运算符对 char、varchar、text 进行比较时，字符串末尾的空格会被忽略。例如，使用 where 查询'a'字符串，查询结果中可能包含 a 后面有空格的情况，反之，如查询条件字符串末尾有空格，如'a '，空格也会被忽略。

由于数据库对大小写不敏感，因此，char、varchar、text、enum、set 类型都不区分大小写。例如，使用 where 查询'a'字符串，则'A'和'a'都会被查询出来。而 binary、varbinary、blob 类型区分大小写，这是因为它们使用二进制方式保存数据。

3. enum 类型

enum 类型又称为枚举类型，定义 enum 类型的语法格式如下。

```
enum('值 1','值 2','值 3',…, '值 n')
```

在上述格式中，('值 1','值 2','值 3',…, '值 n')称为枚举列表，enum 类型的数据只能从枚举列表中获取，并且只能取一个。

【例 5-11】 创建名称为 enum_test 的数据表，并插入数据。具体 SQL 语句与执行结果如下。

```
mysql> create table enum_test(sex enum('male','female'));
Query OK, 0 rows affected (0.36 sec)
```

数据表创建成功，插入两条数据并查看。

```
mysql> insert into enum_test values('male'),('female');
Query OK, 2 rows affected (0.10 sec)
Records: 2  Duplicates: 0  Warnings: 0
mysql> select * from enum_test;
+---------+
| sex     |
+---------+
| male    |
| female  |
+---------+
2 rows in set (0.04 sec)
```

插入枚举列表中没有的数据，从执行结果中可以看出，插入失败。

```
mysql> insert into enum_test values('ma');
ERROR 1265 (01000): Data truncated for column 'sex' at row 1
```

enum 值在内部用整数表示，并且每个枚举值均有一个索引值：列表值所允许的成员值从 1 开始编号，MySQL 存储的就是这个索引编号。枚举最多可以有 65535 个元素。

4. set 类型

set 类型用于保存字符串对象，其定义格式与 enum 类型相似，语法格式如下。

```
set('值 1','值 2','值 3',…, '值 n');
```

set 类型的列表中最多可以有 64 个值，且列表中的每个值都有一个顺序编号，为了节省空间，实际保存在记录中的也是顺序编号，但在 select、insert 等语句进行操作时，仍然要使用列表中的值。

set 类型与 enum 类型的区别在于，可以从列表中选择一个或多个值来保存，多个值之间用逗号分隔。

【例 5-12】 创建名称为 set_test 的数据表，并插入数据。具体 SQL 语句与执行结果如下。

```
mysql> create table set_test(t set('book','game','code'));
Query OK, 0 rows affected (0.15 sec)
```

数据表创建成功，插入 3 条数据并查看。

```
mysql> insert into set_test values(''),('book'),('book,game');
Query OK, 3 rows affected (0.04 sec)
Records: 3  Duplicates: 0  Warnings: 0
mysql> select * from set_test;
+------------+
| t          |
```