

MySQL学习笔记 (Day008 : 数据类型)

MySQL 学习

MySQL 学习 笔记 (Day008 : 数据类型)

一. INT 类型

1. INT 类型的分类

2. INT 类型的使用

3. INT(N)

4. AUTO_INCREMENT

二. 数字类型

1. 数字类型的分类

三. 字符串类型

1. 字符串类型介绍

2. N 和字符集

3. BLOB 和 TEXT

四. 字符集

1. 常见的字符集

2. collation

五. 集合类型

六. 日期类型

一. INT 类型

1. INT 类型的分类

- TINYINT
- 存储空间：1 字节
- 取值范围
- 有符号(signed)：[-128, 127]
- 无符号(unsigned)：[0, 255]
- SMALLINT
- 存储空间：2 字节
- 取值范围
- 有符号(signed)：[-32768, 32767]
- 无符号(unsigned)：[0, 65535]
- MEDIUMINT
- 存储空间：3 字节
- 取值范围
- 有符号(signed)：[-8388608, 8388607]
- 无符号(unsigned)：[0, 16777215]
- INT
- 存储空间：4 字节
- 取值范围
- 有符号(signed)：[-2147483648, 2147483647]
- 无符号(unsigned)：[0, 4294967295]
- BIGINT
- 存储空间：8 字节
- 取值范围
- 有符号(signed)：[-9223372036854775808, 9223372036854775807]
- 无符号(unsigned)：[0, 18446744073709551615]

2. INT 类型的使用

- 自增长ID
- 推荐 使用 BIGINT，而不是INT；
- unsigned or signed
- 根据实际情况使用，一般情况下推荐 默认 的 signed
- unsigned 的注意事项

```
mysql> create table test_unsigned(a int unsigned, b int unsigned);
Query OK, 0 rows affected (0.14 sec)

mysql> insert into test_unsigned values(1, 2);
Query OK, 1 row affected (0.03 sec)

mysql> select a - b from test_unsigned;
ERROR 1690 (22003): BIGINT UNSIGNED value is out of range in '(`burn_test`.`test_unsigned`.`a` - `burn_test`.`test_unsigned`.`b`)'

mysql> select b - a from test_unsigned;
+-----+
| b - a |
+-----+
| 1 |
+-----+
1 row in set (0.00 sec)

mysql> set sql_mode = 'no_unsigned_subtraction'; -- 这样就可以得到负数
Query OK, 0 rows affected (0.00 sec)

mysql> select a - b from test_unsigned;
+-----+
| a - b |
+-----+
| -1 |
+-----+
1 row in set (0.00 sec)
```

一般情况下使用 int 时，推荐有符号数 (signed)，使用无符号数只是比原来多一倍的取值，数量级上没有改变。

如果需要取值范围很大，直接选择用 BIGINT

3. INT(N)

```
mysql> show create table test_unsigned;
+-----+-----+
| Table | Create Table |
+-----+-----+
| test_unsigned | CREATE TABLE `test_unsigned` (
  `a` int(10) unsigned DEFAULT NULL,
  `b` int(10) unsigned DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 |
+-----+-----+
1 row in set (0.00 sec)

int(N) 和 zerofill
int(N)中的 N 是显示宽度，不表示 存储的数字的 长度 的上限。
zerofill 表示当存储的数字 长度 < N 时，用 数字0 填充左边，直至补满长度 N
当存储数字的长度 超过N时，按照 实际存储 的数字显示

mysql> create table test_int_n(a int(3) zerofill); -- 显示宽度N=3
Query OK, 0 rows affected (0.11 sec)

mysql> insert into test_int_n values(1);
Query OK, 1 row affected (0.04 sec)

mysql> select * from test_int_n;
+-----+
| a |
+-----+
| 001 | -- 不满 N=3时，左边用0填充
+-----+
1 row in set (0.00 sec)

mysql> insert into test_int_n values(1111);
Query OK, 1 row affected (0.03 sec)

mysql> select * from test_int_n;
+-----+
| a |
+-----+
| 001 |
| 1111 | -- 超过N=3的长度时，是什么数字，显示什么数字
+-----+
2 rows in set (0.00 sec)

mysql> select a, HEX(a) from test_int_n\G
***** 1. row *****
a: 001
HEX(a): 1 -- 实际存储的还是1
***** 2. row *****
a: 1111
HEX(a): 457 -- 1111对应的16进制就是457
2 rows in set (0.00 sec)
```

int(N)中的 N 和 zerofill 配合才有意义，且仅仅是显示的时候才有意义，和实际存储没有关系，不会去截取数字的长度。

4. AUTO_INCREMENT

- 自增
- 每张表一个
- 必须是索引的一部分

```
mysql> create table test_auto_increment(a int auto_increment);
ERROR 1075 (42000): Incorrect table definition; there can be only one auto column and it must be defined as a key
-- 没有指定为key，报错了

mysql> create table test_auto_increment(a int auto_increment primary key); -- 指定为key后有效
Query OK, 0 rows affected (0.11 sec)

mysql> insert into test_auto_increment values(NULL); -- 插入NULL值
Query OK, 1 row affected (0.03 sec)

mysql> select * from test_auto_increment;
+----+
| a |
+----+
| 1 | -- 插入NULL值，便可以让其自增，且默认从1开始
+----+
1 row in set (0.00 sec)

mysql> insert into test_auto_increment values(0); -- 插入 0
Query OK, 1 row affected (0.03 sec)

mysql> select * from test_auto_increment;
+----+
| a |
+----+
| 1 |
| 2 | -- 插入 0，自增长为2
+----+
2 rows in set (0.00 sec)

mysql> insert into test_auto_increment values(-1); -- 插入 -1
Query OK, 1 row affected (0.02 sec)

mysql> select * from test_auto_increment;
+----+
| a |
+----+
| -1 | -- 刚刚插入的-1
| 1 |
| 2 |
+----+
3 rows in set (0.00 sec)

mysql> insert into test_auto_increment values(NULL); -- 继续插入NULL
Query OK, 1 row affected (0.02 sec)

mysql> select * from test_auto_increment;
+----+
| a |
+----+
| -1 |
| 1 |
| 2 |
| 3 | -- 刚刚插入NULL，自增为3
+----+
4 rows in set (0.00 sec)

mysql> insert into test_auto_increment values('0'); -- 插入字符0
Query OK, 1 row affected (0.04 sec)

mysql> select * from test_auto_increment;
+----+
| a |
+----+
| -1 |
| 1 |
| 2 |
| 3 |
| 4 | -- 插入字符'0'后，自增长为4
+----+
5 rows in set (0.00 sec)

mysql> update test_auto_increment set a = 0 where a = -1; -- 更新为0
Query OK, 1 row affected (0.03 sec)
Rows matched: 1 Changed: 1 Warnings: 0

mysql> select * from test_auto_increment;
+----+
| a |
+----+
| 0 | -- 原来的 -1 更新为0
| 1 |
| 2 |
| 3 |
| 4 |
+----+
5 rows in set (0.00 sec)

--
-- 数字 0 这个值比较特殊，插入0和插入NULL的效果是一样的，都是代表自增
--

-----

mysql> insert into test_auto_increment values(NULL), (100), (NULL);
Query OK, 3 rows affected (0.02 sec)
Records: 3 Duplicates: 0 Warnings: 0

mysql> select * from test_auto_increment;
+----+
| a |
+----+
| 0 |
| 1 |
| 2 |
| 3 |
| 4 |
| 5 | -- 第一个NULL
| 100 | -- 100
| 101 | -- 第二个NULL，按当前最大的值 + 1 来设置，之前是100，所以这里101
+----+
8 rows in set (0.00 sec)

mysql> insert into test_auto_increment values(99); -- 插入99
Query OK, 1 row affected (0.02 sec)

mysql> select * from test_auto_increment;
+----+
| a |
+----+
| 0 |
| 1 |
| 2 |
| 3 |
| 4 |
| 5 |
| 99 | -- 刚刚插入的 99
| 100 |
| 101 |
+----+
9 rows in set (0.00 sec)
```

AUTO_INCREMENT 是实例启动时，取当前表的最大值，然后 +1 则为下次自增的值。（MAX + 1）

TIPS:

insert into tablename select NULL; 等价与 insert into tablename values (NULL);

二. 数字类型

1. 数字类型的分类

- 单精度类型：FLOAT
 - 存储空间：4 字节
 - 精确性：低
- 双精度类型：DOUBLE
 - 占用空间：8 字节
 - 精确性：低，比FLOAT高
- 高精度类型：DECIMAL
 - 占用空间：变长
 - 精确性：非常高

注意：财务系统 必须使用DECIMAL

三. 字符串类型

1. 字符串类型介绍

类型	说明	N的含义	是否有字符集	最大长度
CHAR(N)	定长字符	字符	是	255
VARCHAR(N)	变长字符	字符	是	16384
BINARY(N)	定长二进制字节	字节	否	255
VARBINARY(N)	变长二进制字节	字节	否	16384
TINYBLOB(N)	二进制大对象	字节	否	256
BLOB(N)	二进制大对象	字节	否	16K
MEDIUMBLOB(N)	二进制大对象	字节	否	16M
LONGBLOB(N)	二进制大对象	字节	否	4G
TINYTEXT(N)	大对象	字节	是	256
TEXT(N)	大对象	字节	是	16K
MEDIUMTEXT(N)	大对象	字节	是	16M
LONGTEXT(N)	大对象	字节	是	4G

2. N和字符集

- char(N)
 - 假设当前table的字符集的最大长度为w, 则char(N)的最大存储空间为 (N X W)Byte;假设使用UTF-8, 则char(10)可以最小存储10个字节的字符, 最大存储30个字节的字符, 其实是另一种意义上的 varchar
 - 当存储的字符数 小于N 时, 尾部使用空格填充, 并且填充最小字节的空格

```
mysql> create table test_char(a char(10));
Query OK, 0 rows affected (0.12 sec)

mysql> show create table test_char;
+-----+
| Table      | Create Table |
+-----+
| test_char | CREATE TABLE `test_char` (
  `a` char(10) DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 |
+-----+
1 row in set (0.00 sec)

mysql> insert into test_char values('abc');
Query OK, 1 row affected (0.02 sec)

mysql> insert into test_char values('你好吗');
Query OK, 1 row affected (0.05 sec)

mysql> insert into test_char values('大家好ab');
Query OK, 1 row affected (0.02 sec)

mysql> insert into test_char values('大家ab好');
Query OK, 1 row affected (0.03 sec)

mysql> insert into test_char values('大家ab好吗');
Query OK, 1 row affected (0.03 sec)

mysql> select a, length(a) from test_char;
+-----+
| a          | length(a) |
+-----+
| abc        | 3         |
| 你好吗     | 9         |
| 大家好ab   | 11        |
| 大家ab好   | 11        |
| 大家ab好吗 | 14        |
+-----+
5 rows in set (0.00 sec)

mysql> select a, hex(a) from test_char;
+-----+
| a          | hex(a)     |
+-----+
| abc        | 616263     |
| 你好吗     | E4BDAGE5A58DE59097 |
| 大家好ab   | E5A4A7ESAEB6E5A5BD6162 |
| 大家ab好   | E5A4A7ESAEB6E5A5BD0 |
| 大家ab好吗 | E5A4A7ESAEB6E5A5BD0E59097 |
+-----+
5 rows in set (0.00 sec)

mysql> select hex(' ');
+-----+
| hex(' ') |
+-----+
| 20       | -- 注意 空格对应的16进制数字是 20
+-----+
1 row in set (0.00 sec)
```

test_char 表实际二进制存储文件

```
--
-- shell> hexdump -C test_char.idb
--

-- 1:abc
-- 2:你好吗
-- 3:大家好ab
-- 4:大家ab好
-- 5:大家ab好吗

-- ---省略---
00006070 73 75 70 72 65 6d 75 6d 0a 00 00 00 10 00 24 00 |supremum.....$.|
00006080 00 00 00 02 03 00 00 00 00 1f 33 a8 00 00 00 26 |.....3...&|
00006090 01 10 61 62 63 20 20 20 20 20 20 20 0a 00 00 00 |..abc      ...| -- 1:后面补了7个空格
000060a0 18 00 24 00 00 00 00 02 04 00 00 00 00 1f 34 a9 |..$......4.|
000060b0 00 00 00 25 01 10 e4 bd a0 e5 a5 bd e5 90 97 20 |...%......| -- 2:补充了1个空格
000060c0 0b 00 00 00 20 00 25 00 00 00 00 02 05 00 00 00 |....%......|
000060d0 00 1f 39 ac 00 00 00 26 01 10 e5 a4 a7 e5 ae b6 |..9...&.....| -- 3:没有补充空格
000060e0 e5 a5 bd 61 62 0b 00 00 00 28 00 25 00 00 00 00 |...ab...{%....| --
000060f0 02 06 00 00 00 00 1f 3a ad 00 00 00 28 01 10 e5 |.....!...{...| --
00006100 a4 a7 e5 ae b6 61 62 e5 a5 bd 0e 00 00 00 30 ff |....ab.....0.| -- 4: 没有补充空格
00006110 5f 00 00 00 00 02 07 00 00 00 00 1f 3f b0 00 00 |.....?...|
00006120 00 29 01 10 e5 a4 a7 e5 ae b6 61 62 e5 a5 bd e5 |.).....ab.....| --
00006130 90 97 00 00 00 00 00 00 00 00 00 00 00 00 00 |.....| -- 5: 没有补充空格
00006140 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |.....|
-- ---省略---
```

• varchar(N)

```
mysql> create table test_varchar(a varchar(10));
Query OK, 0 rows affected (0.12 sec)

mysql> show create table test_varchar;
+-----+
| Table      | Create Table |
+-----+
| test_varchar | CREATE TABLE `test_varchar` (
  `a` varchar(10) DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 |
+-----+
1 row in set (0.00 sec)

mysql> insert into test_varchar values('abc');
Query OK, 1 row affected (0.03 sec)

mysql> insert into test_varchar values('你好吗');
Query OK, 1 row affected (0.02 sec)

mysql> insert into test_varchar values('大家好ab');
Query OK, 1 row affected (0.03 sec)

mysql> insert into test_varchar values('大家ab好');
Query OK, 1 row affected (0.02 sec)

mysql> insert into test_varchar values('大家ab好吗');
Query OK, 1 row affected (0.03 sec)

mysql> select a, hex(a) from test_varchar;
+-----+
| a          | hex(a)     |
+-----+
| abc        | 616263     |
| 你好吗     | E4BDAGE5A58DE59097 |
| 大家好ab   | E5A4A7ESAEB6E5A5BD6162 |
| 大家ab好   | E5A4A7ESAEB6E5A5BD0 |
| 大家ab好吗 | E5A4A7ESAEB6E5A5BD0E59097 |
+-----+
5 rows in set (0.00 sec)

mysql> select a, length(a) from test_varchar;
+-----+
| a          | length(a) |
+-----+
| abc        | 3         |
| 你好吗     | 9         |
| 大家好ab   | 11        |
| 大家ab好   | 11        |
| 大家ab好吗 | 14        |
+-----+
5 rows in set (0.00 sec)
```

test_varchar 表实际二进制存储文件

```
--
-- shell> hexdump -C test_char.idb
--

-- 1:abc
-- 2:你好吗
-- 3:大家好ab
-- 4:大家ab好
-- 5:大家ab好吗

-- 和char一样观察, 都没有进行空格的填充

00006070 73 75 70 72 65 6d 75 6d 03 00 00 00 10 00 1d 00 |supremum.....|
00006080 00 00 00 02 08 00 00 00 00 1f 44 b5 00 00 00 29 |.....D....|
00006090 01 10 61 62 63 09 00 00 00 18 00 23 00 00 00 00 |..abc.....#.|
000060a0 02 09 00 00 00 00 1f 45 b6 00 00 00 2b 01 10 e4 |.....E....|
000060b0 bd a0 e5 a5 bd e5 90 97 0b 00 00 00 20 00 25 00 |......%.|
000060c0 00 00 00 02 0a 00 00 00 00 1f 4a b9 00 00 00 2c |.....J....|
000060d0 01 10 e5 a4 a7 e5 ae b6 e5 a5 bd 61 62 0b 00 00 |.....ab...|
000060e0 00 29 00 25 00 00 00 00 02 0b 00 00 00 1f 4b |.(%......R|
000060f0 ba 00 00 00 2c 01 10 e5 a4 a7 e5 ae b6 61 62 e5 |.....ab...|
00006100 a5 bd 0e 00 00 00 30 ff 67 00 00 00 02 0c 00 |.....0.g....|
00006110 00 00 00 1f 50 bd 00 00 00 2d 01 10 e5 a4 a7 e5 |...P.....|
00006120 ae b6 61 62 e5 a5 bd e5 90 97 00 00 00 00 00 |...ab.....|
00006130 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |.....|
```

• 插入数据尾部带空格


```
mysql> insert into test_char values('好好好  '); -- 后面有3个空格
Query OK, 1 row affected (0.03 sec)

mysql> insert into test_varchar values('好好好  '); -- 后面有3个空格
Query OK, 1 row affected (0.02 sec)

--
-- test_char 表
--
mysql> select a, length(a) from test_char;
+-----+
| a          | length(a) |
+-----+
| abc        | 3         |
| 你好吗    | 9         |
| 大家好ab   | 11        |
| 大家ab好   | 11        |
| 大家ab好吗 | 14        |
| 好好好    | 9         | -- 只有9个字节
+-----+
6 rows in set (0.00 sec)

mysql> select a, hex(a) from test_char;
+-----+
| a          | hex(a)    |
+-----+
| abc        | 616263    |
| 你好吗    | E4BDAAE5A58DE59097 |
| 大家好ab   | E5A4A7E5AE86E5A58D6162 |
| 大家ab好   | E5A4A7E5AE86E6162E5A58D |
| 大家ab好吗 | E5A4A7E5AE86E6162E5A58DE59097 |
| 好好好    | E5A58DE5A58DE5A58D | -- 无填充空格
+-----+
6 rows in set (0.00 sec)

--
-- test_varchar表
--
mysql> select a, length(a) from test_varchar;
+-----+
| a          | length(a) |
+-----+
| abc        | 3         |
| 你好吗    | 9         |
| 大家好ab   | 11        |
| 大家ab好   | 11        |
| 大家ab好吗 | 14        |
| 好好好    | 12        | -- (好好好)9个字节 + 3个字节的空格
+-----+
7 rows in set (0.00 sec)

mysql> select a, hex(a) from test_varchar;
+-----+
| a          | hex(a)    |
+-----+
| abc        | 616263    |
| 你好吗    | E4BDAAE5A58DE59097 |
| 大家好ab   | E5A4A7E5AE86E5A58D6162 |
| 大家ab好   | E5A4A7E5AE86E6162E5A58D |
| 大家ab好吗 | E5A4A7E5AE86E6162E5A58DE59097 |
| 好好好    | E5A58DE5A58DE5A58D2020 | -- 后面有20 20 20，表示3个自己的空格
+-----+
7 rows in set (0.00 sec)
```

上面的现象无法用统一的规则进行表述，但是[官方文档](#)给出的解释是，这样的安排是为了避免索引的碎片

3.BLOB和TEXT

- 在BLOB和TEXT上创建索引时，必须指定索引前缀的长度

```
mysql> create table test_text(a int primary key, b text, key(b));
ERROR 1170 (42000): BLOB/TEXT column 'b' used in key specification without a key length

mysql> create table test_text(a int primary key, b text, key(b(64)));
Query OK, 0 rows affected (0.13 sec)
```

- BLOB和TEXT列不能有默认值
- BLOB和TEXT列排序时只使用该列的前max_sort_length个字节

```
mysql> select @@max_sort_length;
+-----+
| @@max_sort_length |
+-----+
| 1024              |
+-----+
1 row in set (0.00 sec)
```

不建议在MySQL中存储大型的二进制数据，比如歌曲，视频

四. 字符集

1. 常见的字符集

- utf8
- utf8mb4
- gbk
- gb18030

```
mysql> show character set;
+-----+
| Charset | Description                               | Default collation | Maxlen |
+-----+
| big5    | Big5 Traditional Chinese                 | big5_chinese_ci  | 2      |
| dec8    | DEC West European                       | dec8_swedish_ci  | 1      |
| cp850   | DOS West European                       | cp850_general_ci | 1      |
| hp8     | HP West European                        | hp8_english_ci   | 1      |
| ko18r   | KOI8-R Belcom Russian                   | ko18r_general_ci | 1      |
| latin1   | cp1252 West European                     | latin1_swedish_ci | 1      |
| latin2   | ISO 8859-2 Central European              | latin2_general_ci | 1      |
| swe7    | 7bit Swedish                             | swe7_swedish_ci  | 1      |
| ascii   | US ASCII                                  | ascii_general_ci  | 1      |
| ujis    | EUC-JP Japanese                         | ujis_japanese_ci | 3      |
| sjis    | Shift-JIS Japanese                      | sjis_japanese_ci | 2      |
| hebrew  | ISO 8859-8 Hebrew                       | hebrew_general_ci | 1      |
| tis620  | TIS620 Thai                              | tis620_thai_ci   | 1      |
| euckr   | EUC-KR Korean                           | euckr_korean_ci  | 2      |
| ko18u   | KOI8-U Ukrainian                       | ko18u_general_ci | 1      |
| gb2312  | GB2312 Simplified Chinese                | gb2312_chinese_ci | 2      |
| greek   | ISO 8859-7 Greek                         | greek_general_ci  | 1      |
| cp1250  | Windows Central European                 | cp1250_general_ci | 1      |
| gbk     | GBK Simplified Chinese                   | gbk_chinese_ci   | 2      | -- gbk, 表示的字符有限
| latin5  | ISO 8859-9 Turkish                       | latin5_turkish_ci | 1      |
| armSCII8| ARMSCII-8 Armenian                      | armSCII8_general_ci | 1      |
| utf8    | UTF-8 Unicode                            | utf8_general_ci   | 3      | -- utf8, 最长3字节
| ucs2    | UCS-2 Unicode                            | ucs2_general_ci   | 2      |
| cp866   | DOS Russian                              | cp866_general_ci  | 1      |
| keybcs2 | DOS Kamenicky Czech-Slovak               | keybcs2_general_ci | 1      |
| macce   | Mac Central European                     | macce_general_ci  | 1      |
| macroman | Mac West European                       | macroman_general_ci | 1      |
| cp852   | DOS Central European                     | cp852_general_ci  | 1      |
| latin7   | ISO 8859-13 Baltic                       | latin7_general_ci | 1      |
| utf8mb4 | UTF-8 Unicode                            | utf8mb4_general_ci | 4      | -- utf8 + mb4 4个字节
| cp1251  | Windows Cyrillic                         | cp1251_general_ci | 1      |
| utf16   | UTF-16 Unicode                           | utf16_general_ci  | 4      |
| utf16le | UTF-16LE Unicode                         | utf16le_general_ci | 4      |
| cp1256  | Windows Arabic                           | cp1256_general_ci | 1      |
| cp1257  | Windows Baltic                           | cp1257_general_ci | 1      |
| utf32   | UTF-32 Unicode                           | utf32_general_ci  | 4      |
| binary  | Binary pseudo charset                    | binary            | 1      |
| geostd8 | GEOSTD8 Georgian                         | geostd8_general_ci | 1      |
| cp932   | SJIS for Windows Japanese                | cp932_japanese_ci | 2      |
| eucjpms | UJIS for Windows Japanese                | eucjpms_japanese_ci | 3      |
| gb18030 | China National Standard GB18030         | gb18030_chinese_ci | 4      | -- gb18030, 最长4个字节
+-----+
41 rows in set (0.00 sec)
```

2. collation

collation的含义是指排序规则，ci(case insensitive) 结尾的排序集是不区分大小写的

```
mysql> select 'a' = 'A';
+-----+
| 'a' = 'A' |
+-----+
| 1         | -- 因为大小写无关，所以返回1
+-----+
1 row in set (0.00 sec)

mysql> create table test_ci (a varchar(10), key(a));
Query OK, 0 rows affected (0.13 sec)

mysql> insert into test_ci values('a');
Query OK, 1 row affected (0.02 sec)

mysql> insert into test_ci values('A');
Query OK, 1 row affected (0.03 sec)

mysql> select * from test_ci where a = 'a';
+-----+
| a      |
+-----+
| a      |
| A      | -- A也被我们查到了
+-----+
2 rows in set (0.00 sec)
```

上面的情况如果从业务的角度上看，可以很好理解，比如创建一个用户叫做Tom，你是不希望再创建一个叫做tom的用户的

- 修改默认的collation

```
mysql> set names utf8mb4 collate utf8mb4_bin; -- 当前会话有效
Query OK, 0 rows affected (0.00 sec)

mysql> select 'a' = 'A';
+-----+
| 'a' = 'A' |
+-----+
|          0 |
+-----+
1 row in set (0.00 sec)
```

字符集的指定，可以在创建数据库的时候指定，也可以在创建表的时候单独指定，也可以创建列的时候进行指定

五. 集合类型

- 集合类型ENUM和SET
- ENUM类型都多允许65536个值
- SET类型最多允许64个值
- 通过sql_mode参数可以用户约束检查

```
mysql> create table test_col (
-> user varchar(10),
-> sex enum('male', 'female') -- 虽然写的是字符串，单其实存储的整型，效率还是可以的
-> );

mysql> insert into test_col values ("tom", "male");
Query OK, 1 row affected (0.02 sec)

mysql> insert into test_col values ("tom", "xmale"); -- 不是male 和 female
Query OK, 1 row affected, 1 warning (0.03 sec) -- 有warning

mysql> set sql_mode='strict_trans_tables'; -- 设置为严格模式
Query OK, 0 rows affected, 2 warnings (0.00 sec)

mysql> insert into test_col values ("tom", "xmale");
ERROR 1265 (01000): Data truncated for column 'sex' at row 1
```

强烈建议新业务上都设置成严格模式

六. 日期类型

日期类型	占用空间	表示范围
DATETIME	8	1000-01-01 00:00:00 ~ 9999-12-31 23:59:59
DATE	3	1000-01-01 ~ 9999-12-31
TIMESTAMP	4	1970-01-01 00:00:00UTC ~ 2038-01-19 03:14:07UTC
YEAR	1	YEAR(2):1970-2070, YEAR(4):1901-2155
TIME	3	-838:59:59 ~ 838:59:59

TIMESTAMP 带时区功能

```
mysql> create table test_time(a timestamp, b datetime);
Query OK, 0 rows affected (0.12 sec)

mysql> insert into test_time values (now(), now());
Query OK, 1 row affected (0.03 sec)

mysql> select * from test_time;
+-----+-----+
| a          | b          |
+-----+-----+
| 2015-11-28 10:00:39 | 2015-11-28 10:00:39 |
+-----+-----+
1 row in set (0.00 sec)

mysql> select @@time_zone;
+-----+
| @@time_zone |
+-----+
| SYSTEM      |
+-----+
1 row in set (0.00 sec)

mysql> set time_zone='+00:00';
Query OK, 0 rows affected (0.00 sec)

mysql> select @@time_zone;
+-----+
| @@time_zone |
+-----+
| +00:00      |
+-----+
1 row in set (0.00 sec)

mysql> select * from test_time;
+-----+-----+
| a          | b          |
+-----+-----+
| 2015-11-20 2:00:39 | 2015-11-20 10:00:39 | -- 时区的差别体现出来了
+-----+-----+
1 row in set (0.00 sec)
```