数据库

SHUJUKU

202X

微信公众号：灰灰考研

第三章 数据库的基本操作

01 数据库和数据表的创建、查看、修改和删除等操作。

02 MySQL的数据类型，掌握基本数据类型的使用以及表的约束，以及给表添加约束的命令。

一、数据库的基本操作

（1）创建数据库

创建数据库就是在数据库系统中划分一块存储数据的空间，方便数据的分配、放置和管理。在MySQL中使用CREATE DATABASE命令创建数据库，语法格式如下：

CREATE DATABASE数据库名称;

注：“数据库名称”必须是唯一的！

（2）查看数据库

在MySQL中，查看数据库的语法格式如下：

SHOW DATABASE

注：使用该命令可以查询在MySQL中已经存在的所有数据库！

```
mysql> SHOW DATABASES;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| performance_schema |
| test               |
| xxgc               |
+--------------------+
5 rows in set (0.00 sec)
```

（3）修改数据库

数据库创建之后，数据库编码方式就确定了。修改数据库的编码方式，可以使用ALTER DATABASE语句，具体语法如下：

ALTER DATABASE数据库名称DEFAULT CHARACTER SET编码方式COLLATE编码方式_bin;

其中“数据库名称”是要修改的数据库的名字，“编码方式”是修改后的数据库编码方式。

例 将数据库xxgc的编码方式修改为gbk.

SQL语句如下：

```
ALTER DATABASE xxgc DEFAULT CHARACTER SET gbk COLLATE gbk_bin;
```

结果：

```
mysql> SHOW CREATE DATABASE xxgc;
+----------+-----------------------------------------------------------------------------------+
| Database | Create Database                                                                   |
+----------+-----------------------------------------------------------------------------------+
| xxgc     | CREATE DATABASE `xxgc` /*!40100 DEFAULT CHARACTER SET gbk COLLATE gbk_bin */ |
+----------+-----------------------------------------------------------------------------------+
1 row in set (0.00 sec)
```

（4）删除数据库

删除数据库可以使用DROP DATABASE命令，具体语法格式如下：

DROP DATABASE数据库名称;

其中“数据库名称”是要删除的数据库的名字。需要注意的是，如果要删除的数据库不存在，则会出现错误。

例：删除名为xxgc的数据库。

SQL语句如下：

```
DROP DATABASE xxgc;
```

结果：

```
mysql> SHOW DATABASES;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| performance_schema |
| test               |
+--------------------+
4 rows in set (0.00 sec)
```

二、数据类型

MySQL支持多种数据类型，大致分为四类：数值类型，日期和时间类型、字符串类型和二进制类型。

(1)数值类型

MySQL支持所有标准SQL数值类型，包括精确数值类型和近似数值类型。

类型	字节数	范围（有符号）	范围（无符号）	用途
TINYINT	1	(-128，127)	(0，255)	小整数值
SMALLINT	2	(-32 768，32 767)	(0，65 535)	大整数值
MEDIUMINT	3	(-8 388 608，8 388 607)	(0，16 777 215)	大整数值
INT 或 INTEGER	4	(-2 147 483 648，2 147 483 647)	(0，4 294 967 295)	大整数值
BIGINT	8	(-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	(-3.402 823 466 E+38，1.175 494 351 E-38)	(1.175 494 351 E-38，3.402 823 466 E+38)	单精度浮点数值
DOUBLE	8	(-1.797 693 134 862 315 7 E+308，2.225 073 858 507 201 4 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(M,D)	M+2	(-1.797 693 134 862 315 7 E+308，2.225 073 858 507 201 4 E-308)	(2.225 073 858 507 201 4 E-308，1.797 693 134 862 315 7 E+308)	小数值

（2）日期和时间类型

表示日期和时间值的日期和时间类型有DATETIME、DATE、TIMESTAMP、TIME和YEAR。

每个时间类型有一个有效值范围和一个“零”值，当输入不合法的值时，MySQL使用“零”值插入。

类型	字节数	范围	格式	零值
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
YEAR	1	(1901,2155)	YYYY	0000
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 00:00:00, 2038-01-19 03:14:07)	YYYY-MM-DD HH:MM:SS	0000-00-00 00:00:00

（3）字符串和二进制类型

为了存储字符串，图片和声音等数据，MySQL提供了字符串和二进制类型。

类型	大小	使用
CHAR（n）	0～255 字符	定长字符串，n 为字符串的最大长度。若输入数据的长度超过了 n 值，超出部分将会被截断；否则，不足部分用空格填充。例如，对于 CHAR(4)，若插入值为'abc'，则其占用的存储空间为 4 个字节
VARCHAR（n）	0～65536 字符	变长字符串，n 为字符串的最大长度。占用字节数随输入数据的实际长度而变化，最大长度不得超过 n+1。例如，对于 VARCHAR(4)，若插入值为'abc'，则其占用的存储空间为 4 个字节，若插入值为'abcd'，则其占用的存储空间为 5 个字节
BINARY（n）	0～255 字节	固定长度的二进制数据，n 为字节长度，若输入数据的字节长度超过了 n 值，超出部分将会被截断；否则，不足部分用字符'\0'填充。例如，对于 BINARY(3)，插入值为'a\0'时，会变成'a\0\0'值存入
VARBINARY（n）	0～65536 字节	可变长度的二进制数据，n 为字节长度
ENUM	1～65535 个值	枚举类型，语法格式为：ENUM ('值 1'，'值 2'，…，'值 n')。该类型的字段值只能为枚举值中的某一个。例如，性别字段数据类型可以设为 ENUM('男'，'女')
SET	1～64 个值	集合类型，语法格式为：SET('值 1'，'值 2'，…，'值 n')。例如，人的兴趣爱好字段的数据类型可以设为 SET（'听音乐'，'看电影'，'购物'，'游泳'，'旅游'），该字段的值从集合中取值，且可以取多个值
BIT（n）	1～64 位	位字段类型。如果输入的值的长度小于 n 位，在值的左边用 0 填充。例如，为 BIT(6)分配值'101'的效果与分配值'000101'的效果相同
TINYBLOB	0～255 字节	不超过 255 个字符的二进制字符串
BLOB	0～65535 字节	二进制形式的文本数据，主要存储图片、音频等信息
MEDIUMBLOB	0～16777215 字节	二进制形式的中等长度文本数据
LONGBLOB	0～4294967295 字节	二进制形式的极大长度文本数据
TINYTEXT	0～255 字节	短文本字符串
TEXT	0～65535 字节	文本数据。如新闻内容、博客、日志等
MEDIUMTEXT	0～16777215 字节	中等长度文本数据
LONGTEXT	0～4294967295 字节	极大长度文本数据

三、数据表的基本操作

（1）创建数据表

数据库创建成功之后，接下来需要在数据库中创建数据表，因为数据表是数据库中存放数据的对象实体。没有数据表，数据库中其他的数据对象就没有意义。

注：在创建数据表之前，一定要使用“USE数据库名”明确是在哪个数据库中创建的，否则系统会抛出“NO database selected”错误。创建数据表的语法格式如下：

```
CREATE TABLE数据表名称
(
字段名1数据类型[完整性约束条件],
字段名2数据类型[完整性约束条件],
    ...
字段名3数据类型[完整性约束条件]
);
```

在上述语法格式中，“数据表名称”是创建的数据表的名字，“字段名”是数据表的列名，“完整性约束条件”是字段的特殊约束条件。

如：在xxgc数据库中创建一个用于存储教师信息的teacher表，其结构如表：

teacher表结构

字段名	数据类型	备注说明
id	INT（5）	教师的工号
name	VARCHAR（10）	教师的姓名
email	VARCHAR（30）	教师的邮箱地址

选择创建表的数据库，创建表的SQL语句如下：

```
USE xxgc;
CREATE TABLE teacher
(
Id INT(5),
name VARCHAR(10),
email VARCHAR(30)
);
```

执行后显示：Query OK, 0 rows affected（0.34sec），说明teacher表创建成功。

（2）查看数据表

1. 使用SHOW CREATE TABLE语句查看数据表语法格式如下：

```
SHOW CREATE TABLE数据表名称;
```

例 使用SHOW CREATE TABLE语句查看teacher表。

SQL语句如下：

```
SHOW CREATE TABLE teacher;
```

结果如下：

```
mysql> SHOW CREATE TABLE teacher;
+---------+------------------------------------------------------------------
------------------------------------------------------------------------------+
| Table   | Create Table
                                                                             |
+---------+------------------------------------------------------------------
------------------------------------------------------------------------------+
| teacher | CREATE TABLE `teacher` (
  `id` int(5) DEFAULT NULL,
  `name` varchar(10) DEFAULT NULL,
  `email` varchar(30) DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8 |
+---------+------------------------------------------------------------------
------------------------------------------------------------------------------+
1 row in set (0.00 sec)
```

```
mysql> SHOW CREATE TABLE teacher\G
*************************** 1. row ***************************
       Table: teacher
Create Table: CREATE TABLE `teacher` (
  `id` int(5) DEFAULT NULL,
  `name` varchar(10) DEFAULT NULL,
  `email` varchar(30) DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8
1 row in set (0.00 sec)
```

2. 使用DESCRIBE语句查看数据表使用DESCRIBE语句查看数据表，可以查看到数据表的字段名、类型、是否为空，是否为主键等信息。语法格式如下：

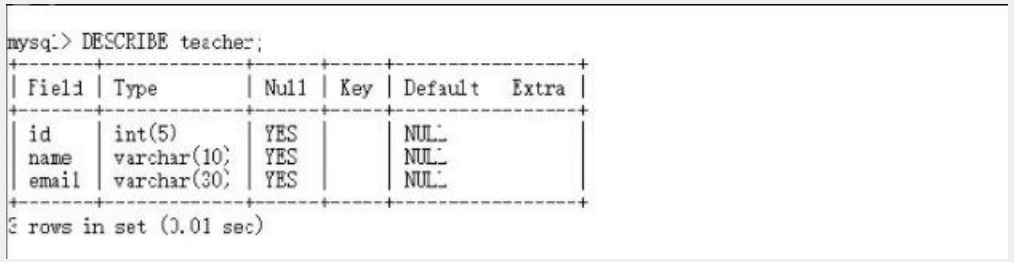
```
DESCRIBE表名;
```

```
DESC表名;
```

【例】使用DESCRIBE语句查看teacher表。

```
DESCRIBE teacher;
```

结果如下：



```
mysql> DESCRIBE teacher;
+-------+------------+------+-----+---------+-------+
| Field | Type       | Null | Key | Default | Extra |
+-------+------------+------+-----+---------+-------+
| id    | int(5)     | YES  |     | NULL    |       |
| name  | varchar(10)| YES  |     | NULL    |       |
| email | varchar(30)| YES  |     | NULL    |       |
+-------+------------+------+-----+---------+-------+
3 rows in set (0.01 sec)
```

☆ Field表示该表的字段名；Type表示对应字段的数据类型；Null表示对应字段是否可以存储NULL值；Key表示对应字段是否编制索引和约束；Default表示对应字段是否有默认值；Extra表示获取到的与对应字段相关的附加信息。

（3）修改数据表

数据表创建之后，用户还可以对表中的某些信息进行修改，包括修改数据表的结构以及表中字段的信息，如修改表名、修改字段名、修改字段的数据类型等。

1. 修改表名语法格式如下：

```
ALTER TABLE旧表名RENAME [TO]新表名;
```

【例】将数据库xxgc中teacher表的表名改为xxgc_teacher。

修改表名之前，先用SHOW TABLES语句查看数据库中的表，结果如图3-9所示。

执行下述命令，将teacher表名改为xxgc_teacher。

```
    ALTER TABLE teacher RENAME xxgc_teach
er;
```

上述命令执行成功后，再用SHOW TABLES语句查看数据库中的表，结果：

```
mysql> SHOW TABLES;
+----------------+
| Tables_in_xxgc |
+----------------+
| teacher        |
+----------------+
1 row in set (0.00 sec)
```

```
mysql> SHOW TABLES;
+----------------+
| Tables_in_xxgc |
+----------------+
| xxgc_teacher   |
+----------------+
1 row in set (0.00 sec)
```

2. 修改字段名和数据类型

语法格式如下：

```
ALTER TABLE表名CHANGE旧字段名 新字段名 新数据类型;
```

其中，“旧字段名”是修改之前的字段名称，“新字段名”是修改之后的字段名称，“新数据类型”是修改后的数据类型。注意，修改后的数据类型不能为空。如果只修改字段名，不修改数据类型，可以将新数据类型写为字段原来的数据类型。

【例】将xxgc_teacher表中的id字段改名为workid，数据类型保持不变。修改字段之前，首先查看表的信息，执行结果如图：

```
mysql> DESC xxgc_teacher;
+--------+-------------+------+-----+---------+-------+
| Field  | Type        | Null | Key | Default | Extra |
+--------+-------------+------+-----+---------+-------+
| workid | int(5)      | YES  |     | NULL    |       |
| name   | varchar(10) | YES  |     | NULL    |       |
| email  | varchar(30) | YES  |     | NULL    |       |
+--------+-------------+------+-----+---------+-------+
3 rows in set (0.01 sec)
```

执行下述命令，将

```
xgc_teacher CHANGE id wo
```

3. 修改字段的数据类型

语法格式如下：

```
ALTER TABLE表名MODIFY字段名 新数据类型;
```

【例】将xxgc_teacher表中的workid字段的数据类型由INT（5）修改为TINYINT。

执行修改命令之前，先查看xxgc_teacher表的结构，如图：

```
mysql> DESC xxgc_teacher;
+--------+-------------+------+-----+---------+-------+
| Field  | Type        | Null | Key | Default | Extra |
+--------+-------------+------+-----+---------+-------+
| workid | int(5)      | YES  |     | NULL    |       |
| name   | varchar(10) | YES  |     | NULL    |       |
| email  | varchar(30) | YES  |     | NULL    |       |
+--------+-------------+------+-----+---------+-------+
3 rows in set (0.01 sec)
```

执行修改命令，SQL语句如下：命令成功执行后，再查看一下xxgc_teacher表的结构。结果如图：

```
ALTER TABLE xxgc_teacher MODIFY workid TINYINT;
```

```
mysql> DESC xxgc_teacher;
+--------+-------------+------+-----+---------+-------+
| Field  | Type        | Null | Key | Default | Extra |
+--------+-------------+------+-----+---------+-------+
| workid | tinyint(4)  | YES  |     | NULL    |       |
| name   | varchar(10) | YES  |     | NULL    |       |
| email  | varchar(30) | YES  |     | NULL    |       |
+--------+-------------+------+-----+---------+-------+
3 rows in set (0.01 sec)
```

4、添加字段
语法格式如下：

```
    ALTER TABLE表名
    ADD新字段名 数据类型[约束条件] [FIRST
|AFTER已经存在的字段名];
```

其中，“新字段名”是新添加的字段名称，“FIRST”是可选参数，用于将新添加的字段设置为表的第一个字段，“AFTER已经存在的字段名”也是可选参数，用于将新添加的字段添加到指定字段的后面。如不指定位置，则默认将新添加字段追加到表末尾。
【例】在xxgc_teacher表中添加一个没有约束条件的INT（4）类型的age字段。
SQL语句如下：

```
    ALTER TABLE xxgc_teacher ADD age INT
(4);
```

为了验证age字段是否添加成功，使用DESC语句查看xxgc_teacher表的结构，执行结果如图：

```
mysql> DESC xxgc_teacher;
+--------+------------+------+-----+---------+-------+
| Field  | Type       | Null | Key | Default | Extra |
+--------+------------+------+-----+---------+-------+
| workid | tinyint(4) | YES  |     | NULL    |       |
| name   | varchar(10)| YES  |     | NULL    |       |
| email  | varchar(30)| YES  |     | NULL    |       |
| age    | int(4)     | YES  |     | NULL    |       |
+--------+------------+------+-----+---------+-------+
4 rows in set (0.01 sec)
```

5、删除字段
语法格式如下：

```
ALTER TABLE表名DROP字段名;
```

【例】删除xxgc_teacher表中的email字段。SQL语句如下：

```
ALTER TABLE xxgc_teacher DROP email;
```

为了验证email字段是否删除成功，使用DESC语句查看xxgc_teacher表的结构，执行结果如图：

```
mysql> DESC xxgc_teacher;
+--------+------------+------+-----+---------+-------+
| Field  | Type       | Null | Key | Default | Extra |
+--------+------------+------+-----+---------+-------+
| workid | tinyint(4) | YES  |     | NULL    |       |
| name   | varchar(10)| YES  |     | NULL    |       |
| age    | int(4)     | YES  |     | NULL    |       |
+--------+------------+------+-----+---------+-------+
3 rows in set (0.01 sec)
```

6、修改字段的位置
语法结构：

```
ALTER TABLE表名MODIFY字段名1新数据类型FIRST|AFTER字段名2;
```

其中“FIRST”是可选参数，用于将“字段名1”设置为表的第一个字段，“AFTER字段名2”也是可选参数，用于将“字段名1”移动到“字段名2”的后面。此命令可以同时修改字段的数据类型和位置。如果只修改位置，不修改数据类型，可以将新数据类型写为字段原来的数据类型。

【例】将xxgc_teacher表中的name字段修改为表中的第一个字段。SQL语句如下：

```
    ALTER TABLE xxgc_teacher MODIFY name
VARCHAR(10) FIRST;
```

使用DESC语句查看xxgc_teacher表的结构，执行结果如图：

```
mysql> DESC xxgc_teacher;
+--------+------------+------+-----+---------+-------+
| Field  | Type       | Null | Key | Default | Extra |
+--------+------------+------+-----+---------+-------+
| name   | varchar(10)| YES  |     | NULL    |       |
| workid | tinyint(4) | YES  |     | NULL    |       |
| age    | int(4)     | YES  |     | NULL    |       |
+--------+------------+------+-----+---------+-------+
3 rows in set (0.01 sec)
```

6、删除数据表

删除数据表的语法格式如下：

```
DROP TABLE表名;
```

【例】删除xxgc_teacher表。SQL语句如下：

```
DROP TABLE xxgc_teacher;
```

为了验证xxgc_teacher表是否删除成功，使用DESC语句查看，执行结果如图：

```
mysql> DESC xxgc_teacher;
ERROR 1146 (42S02): Table 'xxgc.xxgc_teacher' doesn't exist
```

四、数据表的约束

约束的目的是保证数据库中数据的完整性和一致性。在MySQL中，常见的表约束有以下几种：PRIMARY KEY CONSTRAINT（主键约束）、FOREIGN KEYCONSTRAINT（外键约束）、NOT NULL CONSTRAINT（非空约束）、UNIQUE CONSTRAINT（唯一约束）、DEFAULT CONSTRAINT（默认约束）。

1、PRIMARY KEY CONSTRAINT（主键约束）主键，又称主码，由表中的一个字段或多个字段组成，能够唯一地标识表中的一条记录。主键约束要求主键字段中的数据唯一，并且不允许为空。主键分为两种类型：单字段主键和复合主键。注意，每个数据表中最多只能有一个主键。

单字段主键（1）创建表时指定主键语法格式如下：

```
字段名  数据类型  PRIMARY KEY ;
```

【例】创建company表，并设置company_id字段为主键。
SQL语句如下：

```
CREATE TABLE company
(
company_id INT(11) PRIMARY KEY,
company_name VARCHAR(50),
company_address VARCHAR(200)
);
```

执行上述命令之后，用DESC语句查看company表的结构，执行结果如图

```
mysql> DESC company;
+-----------------+--------------+------+-----+---------+-------+
| Field           | Type         | Null | Key | Default | Extra |
+-----------------+--------------+------+-----+---------+-------+
| company_id      | int(11)      | NO   | PRI | NULL    |       |
| company_name    | varchar(50)  | YES  |     | NULL    |       |
| company_address | varchar(200) | YES  |     | NULL    |       |
+-----------------+--------------+------+-----+---------+-------+
3 rows in set (0.01 sec)
```

从图可以看出，company_id字段的“Key”显示为PRI，表示此字段为主键。

（2）为已存在的表添加主键约束　语法格式如下：

```
ALTER TABLE表名MODIFY字段名 数据类型PRIMARY KEY;
```

【例】将company表的company_id字段修改为主键。首先将前面创建的company表删除，再新建company表，SQL语句如下：

```
DROP TABLE company;
CREATE TABLE company
(
company_id INT(11),
company_name VARCHAR(50),
company_address VARCHAR(200)
);
```

执行上述命令，用DESC语句查看company表的结构，执行结果如图

```
mysql> DESC company;
+-----------------+--------------+------+-----+---------+-------+
| Field           | Type         | Null | Key | Default | Extra |
+-----------------+--------------+------+-----+---------+-------+
| company_id      | int(11)      | YES  |     | NULL    |       |
| company_name    | varchar(50)  | YES  |     | NULL    |       |
| company_address | varchar(200) | YES  |     | NULL    |       |
+-----------------+--------------+------+-----+---------+-------+
3 rows in set (0.11 sec)
```

接下来，使用ALTER语句将company_id字段修改为主键，SQL语句如下：

```
ALTER TABLE company MODIFY company_id INT(11) PRIMARY KEY;
```

为了验证company_id字段的主键约束是否添加成功，再次使用DESC语句查看company表的结构，执行结果如图

```
mysql> DESC company;
+-----------------+-------------+------+-----+---------+-------+
| Field           | Type        | Null | Key | Default | Extra |
+-----------------+-------------+------+-----+---------+-------+
| company_id      | int(11)     | NO   | PRI | NULL    |       |
| company_name    | varchar(50) | YES  |     | NULL    |       |
| company_address | varchar(200)| YES  |     | NULL    |       |
+-----------------+-------------+------+-----+---------+-------+
3 rows in set (0.01 sec)
```

从图3-22可以看出，company_id字段的“Key”显示为PRI，表示此字段为主键。

（3）删除主键约束　语法格式如下：

```
ALTER TABLE表名DROP PRIMARY KEY;
```

【例3-19】删除company表的company_id字段的主键约束。SQL语句如下：

```
ALTER TABLE company DROP PRIMARY KEY;
```

为了验证company_id字段的主键约束是否删除，使用DESC语句查看company表的结构，执行结果如图

```
mysql> DESC company;
+-----------------+-------------+------+-----+---------+-------+
| Field           | Type        | Null | Key | Default | Extra |
+-----------------+-------------+------+-----+---------+-------+
| company_id      | int(11)     | NO   |     | NULL    |       |
| company_name    | varchar(50) | YES  |     | NULL    |       |
| company_address | varchar(200)| YES  |     | NULL    |       |
+-----------------+-------------+------+-----+---------+-------+
3 rows in set (0.01 sec)
```

从图可以看出，company_id字段的“Key”为空，表示此字段已经不是主键了。

2. 复合主键复合主键指主键由多个字段组成。

（1）创建表时指定复合主键其语法格式如下：

```
PRIMARY KEY (字段名1, 字段名2,…,字段名3);
```

其中“字段名1,字段名2,…,字段名3”指的是构成主键的多个字段的名称。

【例】创建sales表，设置product_id、region_code字段为复合主键。SQL语句如下：

```
CREATE TABLE sales
(
product_id INT(11),
region_code VARCHAR(10),
quantity INT(11),
price FLOAT,
PRIMARY KEY (product_id,region_code)
 );
```

执行上述命令，用DESC语句查看sales表的结构，执行结果如图

```
mysql> DESC sales;
+-------------+-------------+------+-----+---------+-------+
| Field       | Type        | Null | Key | Default | Extra |
+-------------+-------------+------+-----+---------+-------+
| product_id  | int(11)     | NO   | PRI | 0       |       |
| region_code | varchar(10) | NO   | PRI |         |       |
| quantity    | int(11)     | YES  |     | NULL    |       |
| price       | float       | YES  |     | NULL    |       |
+-------------+-------------+------+-----+---------+-------+
4 rows in set (0.01 sec)
```

（2）为已存在的表添加复合主键　语法格式如下：

```
ALTER TABLE表名ADD PRIMARY KEY (字段名1, 字段名2,…,字段名3);
```

【例】将sales表的product_id字段和region_code字段作为复合主键。首先将前面创建的sales表删除，再新建sales表。SQL语句如下：

```
DROP TABLE sales;
CREATE TABLE sales
(
product_id INT(11),
region_code VARCHAR(10),
quantity INT(11),
price FLOAT
);
```

执行上述命令，用DESC语句查看sales表的结构，执行结果如图

```
mysql> DESC sales;
+-------------+-------------+------+-----+---------+-------+
| Field       | Type        | Null | Key | Default | Extra |
+-------------+-------------+------+-----+---------+-------+
| product_id  | int(11)     | YES  |     | NULL    |       |
| region_code | varchar(10) | YES  |     | NULL    |       |
| quantity    | int(11)     | YES  |     | NULL    |       |
| price       | float       | YES  |     | NULL    |       |
+-------------+-------------+------+-----+---------+-------+
4 rows in set (0.01 sec)
```

接下来，使用ALTER语句将sales表的product_id字段和region_code字段设为复合主键。SQL语句如下：

```
ALTER TABLE sales ADD PRIMARY KEY (product_id, region_code);
```

为了验证product_id字段和region_code字段作为复合主键是否添加成功，再次使用DESC语句查看sales表的结构，执行结果如图

```
mysql> DESC sales;
+-------------+-------------+------+-----+---------+-------+
| Field       | Type        | Null | Key | Default | Extra |
+-------------+-------------+------+-----+---------+-------+
| product_id  | int(11)     | NO   | PRI | 0       |       |
| region_code | varchar(10) | NO   | PRI |         |       |
| quantity    | int(11)     | YES  |     | NULL    |       |
| price       | float       | YES  |     | NULL    |       |
+-------------+-------------+------+-----+---------+-------+
4 rows in set (0.01 sec)
```

（3）删除复合主键约束的语法格式如下：

```
ALTER TABLE表名DROP PRIMARY KEY;
```

2、FOREIGN KEY CONSTRAINT（外键约束）

外键在两个表的数据之间建立关联，它可以是一个字段或者多个字段。一个表可以有一个或者多个外键。一个表的外键可以为空值，若不为空值，则每一个外键值必须等于另一个表中主键的某个值。注意，关联指的是在关系数据库中，相关表之间的联系。它是通过相同或者相容的字段或字段组来表示的。子表的外键必须关联父表的主键，且关联字段的数据类型必须匹配。定义外键后，不允许在主表中删除与子表具有关联关系的记录。主表（父表）：对于两个具有关联关系的表而言，相关联字段中主键所在的那个表即主表。从表（子表）：对于两个具有关联关系的表而言，相关联字段中外键所在的那个表即从表。

1. 创建表时添加外键约束语法格式如下：

```
CONSTRAINT外键名FOREIGN KEY（从表的外键字段名）REFERENCES主表名（主表的主键字段名）
```

其中，“外键名”是指从表创建的外键约束的名字。

【例】创建部门表dept和员工表emp，并在员工表上创建外键。

Step1：先创建主表dept

```
CREATE TABLE dept
(
 id INT(11) PRIMARY KEY,
name VARCHAR(22),
location VARCHAR(50),
description VARCHAR(200)
);
```

Step2：再创建从表emp

```
CREATE TABLE emp
(
 id INT(11) PRIMARY KEY,
name VARCHAR(25),
dept_id INT(11),
salary FLOAT,
CONSTRAINT fk_emp_dept FOREIGN KEY(dept_id) REFERENCES dept(id)
);
```

执行上述命令，用SHOW CREATE TABLE语句查看dept表的结构，执行结果如图

```
mysql> SHOW CREATE TABLE dept;
+-------+----------------------------------------------------------------------------
------------------------------------------------------------------------------------
------------+
| Table | Create Table

          |
+-------+----------------------------------------------------------------------------
------------------------------------------------------------------------------------
------------+
| dept  | CREATE TABLE `dept` (
  `id` int(11) NOT NULL,
  `name` varchar(22) DEFAULT NULL,
  `location` varchar(50) DEFAULT NULL,
  `description` varchar(200) DEFAULT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 |
+-------+----------------------------------------------------------------------------
------------------------------------------------------------------------------------
------------+
1 row in set (0.00 sec)
```

用SHOW CREATE TABLE语句查看emp表的结构，执行结果如图

```
mysql> SHOW CREATE TABLE emp;
+-------+---------------------------------------------------------------------------
----------------------------------------------------------------------------------
-----------------------------------------------------------------------+
| Table | Create Table

                                                                       |
+-------+---------------------------------------------------------------------------
----------------------------------------------------------------------------------
-----------------------------------------------------------------------+
| emp   | CREATE TABLE `emp` (
  `id` int(11) NOT NULL,
  `name` varchar(25) DEFAULT NULL,
  `dept_id` int(11) DEFAULT NULL,
  `salary` float DEFAULT NULL,
  PRIMARY KEY (`id`),
  KEY `fk_emp_dept` (`dept_id`),
  CONSTRAINT `fk_emp_dept` FOREIGN KEY (`dept_id`) REFERENCES `dept` (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 |
+-------+---------------------------------------------------------------------------
----------------------------------------------------------------------------------
-----------------------------------------------------------------------+
1 row in set (0.00 sec)
```

从图3-28可以看出已经成功地创建了dept表和emp表的主外键关联。要特别注意，主表dept的主键字段id和从表emp的外键字段dept_id的数据类型必须兼容或者一致，且含义一样。在创建表时创建表的主外键关联，必须先创建主表，再创建从表。

2. 为已存在的表添加外键约束语法格式如下：
语法格式如下：

```
ALTER TABLE从表名ADD CONSTRAINT外键名FOREIGN KEY（从表的外键字段名）REFERENCES
主表名（主表的主键字段名）；
```

其中，“外键名”是指从表创建的外键约束的名字。
【例】已存在部门表dept和员工表emp，为员工表emp创建外键。
Step1：删除emp表和dept表

```
DROP TABLE emp;
DROP TABLE dept;
```

Step2：先创建主表dept

```
CREATE TABLE dept
(
id INT(11) PRIMARY KEY,
name VARCHAR(22),
location VARCHAR(50),
description VARCHAR(200)
);
```

Step3：再创建从表emp

```
CREATE TABLE emp
(
id INT(11) PRIMARY KEY,
name VARCHAR(25),
dept_id INT(11),
salary FLOAT
);
```

执行上述命令，用SHOW CREATE TABLE语句查看dept表，执行结果如图

```
mysql> SHOW CREATE TABLE dept;
+-------+------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------
------------+
| Table | Create Table
            |
+-------+------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------
------------+
| dept  | CREATE TABLE `dept` (
  `id` int(11) NOT NULL,
  `name` varchar(22) DEFAULT NULL,
  `location` varchar(50) DEFAULT NULL,
  `description` varchar(200) DEFAULT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 |
+-------+------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------
------------+
1 row in set (0.00 sec)
```

用SHOW CREATE TABLE语句查看emp表，执行结果如图

```
mysql> SHOW CREATE TABLE emp;
+-------+---------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------
| Table | Create Table

+-------+---------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------
| emp   | CREATE TABLE `emp` (
  `id` int(11) NOT NULL,
  `name` varchar(25) DEFAULT NULL,
  `dept_id` int(11) DEFAULT NULL,
  `salary` float DEFAULT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 |
+-------+---------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------
1 row in set (0.00 sec)
```

接下来，使用ALTER语句为员工表emp创建外键，SQL语句如下：

```
    ALTER TABLE emp ADD CONSTRAINT fk_emp_dept FOREIGN KEY(dept_id)
REFERENCES
dept (id);
```

3. 删除外键约束　语法格式如下：

```
ALTER TABLE从表名DROP FOREIGN KEY外键名;
```

【例3-24】删除emp表dept_id字段的外键约束，外键约束名是fk_emp_dept。
SQL语句如下：

```
ALTER TABLE emp DROP FOREIGN KEY fk_emp_dept;
```

为了验证emp表dept_id字段的外键约束是否删除，使用SHOW CREATE TABLE语句查看emp表，执行结果如图

```
mysql> SHOW CREATE TABLE emp;
+-------+--------------------------------------------------------------------------------
--------------------------------------------------------------------------------------
----------------------------------------------------+
| Table | Create Table
                                                    |
+-------+--------------------------------------------------------------------------------
--------------------------------------------------------------------------------------
----------------------------------------------------+
| emp   | CREATE TABLE `emp` (
  `id` int(11) NOT NULL,
  `name` varchar(25) DEFAULT NULL,
  `dept_id` int(11) DEFAULT NULL,
  `salary` float DEFAULT NULL,
  PRIMARY KEY (`id`),
  KEY `fk_emp_dept` (`dept_id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 |
+-------+--------------------------------------------------------------------------------

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

3 NOT NULL CONSTRAINT（非空约束）：

非空约束指字段的值不能为空。在同一个数据表中可以定义多个非空字段。

1. 创建表时添加非空约束语法格式如下：

```
字段名 数据类型NOT NULL;
```

【例】创建company表，并设置company_id字段为主键，company_address字段为非空约束。SQL语句如下所示：

```
DROP TABLE IF EXISTS company;
CREATE TABLE company
(
company_id INT(11) PRIMARY KEY,
company_name VARCHAR(50),
company_address VARCHAR(200) NOT NULL
);
```

执行上述命令，用DESC语句查看company表的结构，执行结果如图

```
mysql> DESC company;
+-----------------+--------------+------+-----+---------+-------+
| Field           | Type         | Null | Key | Default | Extra |
+-----------------+--------------+------+-----+---------+-------+
| company_id      | int(11)      | NO   | PRI | NULL    |       |
| company_name    | varchar(50)  | YES  |     | NULL    |       |
| company_address | varchar(200) | NO   |     | NULL    |       |
+-----------------+--------------+------+-----+---------+-------+
3 rows in set (0.14 sec)
```

2. 为已经存在的表添加非空约束语法格式如下：

```
ALTER TABLE表名MODIFY字段名 新数据类型NOT NULL;
```

此命令可以同时修改字段的数据类型和增加非空约束。如果不修改字段的数据类型，将“新数据类型”写为字段原来的数据类型即可。

【例】将company表的company_address字段设置为非空约束。

首先创建company表，SQL语句如下：

```
DROP TABLE IF EXISTS company;
CREATE TABLE company
(
company_id INT(11) PRIMARY KEY,
company_name VARCHAR(50),
company_address VARCHAR(200)
);
```

执行上述命令，用DESC语句查看company表的结构，执行结果如图

```
mysql> DESC company;
+-----------------+--------------+------+-----+---------+-------+
| Field           | Type         | Null | Key | Default | Extra |
+-----------------+--------------+------+-----+---------+-------+
| company_id      | int(11)      | NO   | PRI | NULL    |       |
| company_name    | varchar(50)  | YES  |     | NULL    |       |
| company_address | varchar(200) | YES  |     | NULL    |       |
+-----------------+--------------+------+-----+---------+-------+
3 rows in set (0.01 sec)
```

接下来，使用ALTER语句将company_address字段设置为非空约束，SQL语句如下：

```
    ALTER TABLE company MODIFY company_address VARCHAR(200) NOT NUL
L;
```

为了验证company_address字段的非空约束是否添加成功，再次使用DESC语句查看company表的结构，执行结果如图

```
mysql> DESC company;
+-----------------+--------------+------+-----+---------+-------+
| Field           | Type         | Null | Key | Default | Extra |
+-----------------+--------------+------+-----+---------+-------+
| company_id      | int(11)      | NO   | PRI | NULL    |       |
| company_name    | varchar(50)  | YES  |     | NULL    |       |
| company_address | varchar(200) | NO   |     | NULL    |       |
+-----------------+--------------+------+-----+---------+-------+
3 rows in set (0.01 sec)
```

从图可以看出，company_address字段的“Null”列的值为NO，表示这个字段不允许为空。

3. 删除非空约束语法格式如下：

```
ALTER TABLE表名MODIFY字段名 数据类型;
```

【例】删除company表的company_address字段的非空约束。
SQL语句如下：

```
ALTER TABLE company MODIFY company_address VARCHAR(200);
```

为了验证company_address字段的非空约束是否删除成功，使用DESC语句查看company表的结构，执行结果如图所示。

```
mysql> DESC company;
+-----------------+--------------+------+-----+---------+-------+
| Field           | Type         | Null | Key | Default | Extra |
+-----------------+--------------+------+-----+---------+-------+
| company_id      | int(11)      | NO   | PRI | NULL    |       |
| company_name    | varchar(50)  | YES  |     | NULL    |       |
| company_address | varchar(200) | YES  |     | NULL    |       |
+-----------------+--------------+------+-----+---------+-------+
3 rows in set (0.01 sec)
```

从图可以看出，company_address字段的“Null”列的值为YES，表示这个字段允许为空。

4 UNIQUE CONSTRAINT（唯一约束）

唯一约束要求该列值唯一，不能重复。唯一约束可以确保一列或者几列不出现重复值。

1. 创建表时添加唯一约束语法格式如下：

```
字段名  数据类型  UNIQUE;
```

【例】创建company表，并将company_id字段设置为主键，company_address字段设置为非空约束，company_name字段设置为唯一约束。

SQL语句如下：

```
DROP TABLE IF EXISTS company;
CREATE TABLE company
(
company_id INT(11) PRIMARY KEY,
company_name VARCHAR(50) UNIQUE,
company_address VARCHAR(200) NOT NULL
);
```

执行上述命令，用DESC语句查看company表的结构，执行结果如图

```
mysql> DESC company;
+-----------------+--------------+------+-----+---------+-------+
| Field           | Type         | Null | Key | Default | Extra |
+-----------------+--------------+------+-----+---------+-------+
| company_id      | int(11)      | NO   | PRI | NULL    |       |
| company_name    | varchar(50)  | YES  | UNI | NULL    |       |
| company_address | varchar(200) | NO   |     | NULL    |       |
+-----------------+--------------+------+-----+---------+-------+
3 rows in set (0.01 sec)
```

从图可以看出，company_name字段的“Key”列的值为UNI，表示这个字段具有唯一约束。
注意，一个表中可以有多个字段声明为唯一约束，但是只能有一个主键；声明为主键的字段不允许有空值，但是声明为唯一约束的字段允许存在空值。

2. 为已存在的表添加唯一约束语法格式如下：

```
ALTER TABLE表名MODIFY字段名 新数据类型UNIQUE;
```

此命令可以同时修改字段的数据类型和增加唯一约束。如果不修改字段的数据类型，将“新数据类型”写为字段原来的数据类型即可。

【例】将company表的company_name字段修改为唯一约束。首先创建company表
SQL语句如下：

```
DROP TABLE IF EXISTS company;
CREATE TABLE company
(
company_id INT(11) PRIMARY KEY,
company_name VARCHAR(50),
company_address VARCHAR(200) NOT NULL
);
```

执行上述命令，用DESC语句查看company表的结构，执行结果如图

```
mysql> DESC company;
+-----------------+-------------+------+-----+---------+-------+
| Field           | Type        | Null | Key | Default | Extra |
+-----------------+-------------+------+-----+---------+-------+
| company_id      | int(11)     | NO   | PRI | NULL    |       |
| company_name    | varchar(50) | YES  |     | NULL    |       |
| company_address | varchar(200)| NO   |     | NULL    |       |
+-----------------+-------------+------+-----+---------+-------+
3 rows in set (0.13 sec)
```

为了验证company_name字段的唯一约束是否添加成功，再次使用DESC语句查看company表的结构，执行结果如图

```
mysql> DESC company;
+-----------------+--------------+------+-----+---------+-------+
| Field           | Type         | Null | Key | Default | Extra |
+-----------------+--------------+------+-----+---------+-------+
| company_id      | int(11)      | NO   | PRI | NULL    |       |
| company_name    | varchar(50)  | YES  | UNI | NULL    |       |
| company_address | varchar(200) | NO   |     | NULL    |       |
+-----------------+--------------+------+-----+---------+-------+
3 rows in set (0.01 sec)
```

从图可以看出，company_name字段的“Key”列的值为UNI，表示这个字段具有唯一约束。

3. 删除唯一约束　语法格式如下：

```
ALTER TABLE表名DROP INDEX字段名;
```

【例】删除company表的company_name字段的唯一约束。SQL语句如下：

```
ALTER TABLE company DROP INDEX company_name;
```

为了验证company_name字段的唯一约束是否删除成功，使用DESC语句查看company表的结构，执行结果如图所示。从图可以看出，company_name字段的“Key”列的值为空，表示这个字段已没有唯一约束。

```
mysql> DESC company;
+-----------------+--------------+------+-----+---------+-------+
| Field           | Type         | Null | Key | Default | Extra |
+-----------------+--------------+------+-----+---------+-------+
| company_id      | int(11)      | NO   | PRI | NULL    |       |
| company_name    | varchar(50)  | YES  |     | NULL    |       |
| company_address | varchar(200) | NO   |     | NULL    |       |
+-----------------+--------------+------+-----+---------+-------+
3 rows in set (0.01 sec)
```

5 DEFAULT CONSTRAINT（默认约束）

若将数据表中某列定义为默认约束，在用户插入新的数据行时，如果没有为该列指定数据，那么数据库系统会自动将默认值赋给该列，默认值也可以是空值（NULL）。

1. 创建表时添加默认约束语法格式如下：

```
字段名  数据类型  DEFAULT默认值;
```

【例】创建company表，并将company_id字段设置为主键，company_address字段设置为非空约束，company_name字段设置为唯一约束，company_tel字段的默认值为“0371-”，
SQL语句如下：

```
DROP TABLE IF EXISTS company;
CREATE TABLE company
(
company_id INT(11) PRIMARY KEY,
company_name VARCHAR(50) UNIQUE,
company_address VARCHAR(200) NOT NULL,
company_tel VARCHAR(20) DEFAULT '0371-'
);
```

执行上述命令，用DESC语句查看company表的结构，执行结果如图

```
mysql> DESC company;
+-----------------+-------------+------+-----+---------+-------+
| Field           | Type        | Null | Key | Default | Extra |
+-----------------+-------------+------+-----+---------+-------+
| company_id      | int(11)     | NO   | PRI | NULL    |       |
| company_name    | varchar(50) | YES  | UNI | NULL    |       |
| company_address | varchar(200)| NO   |     | NULL    |       |
| company_tel     | varchar(20) | YES  |     | 0371-   |       |
+-----------------+-------------+------+-----+---------+-------+
4 rows in set (0.01 sec)
```

从图可以看出，company_tel字段的“Default”列的值为“0371-”，表示这个字段具有默认值“0371-”。

2. 为已存在的表添加默认约束语法格式如下：

```
ALTER TABLE表名MODIFY字段名 新数据类型DEFAULT默认值;
```

此命令可以同时修改字段的数据类型和增加默认约束。如果不修改字段的数据类型，将“新数据类型”写为字段原来的数据类型即可。

【例】为company表的company_tel字段添加默认约束，默认值为“0371-”。首先创建company表，SQL语句如下：

```
DROP TABLE company;
CREATE TABLE IF EXISTS company
(company_id INT(11) PRIMARY KEY,
company_name VARCHAR(50) UNIQUE,
company_address VARCHAR(200) NOT NULL,
company_tel VARCHAR(20) );
```

执行上述命令，用DESC语句查看company表的结构，执行结果如图

```
mysql>  DESC company;
+-----------------+-------------+------+-----+---------+-------+
| Field           | Type        | Null | Key | Default | Extra |
+-----------------+-------------+------+-----+---------+-------+
| company_id      | int(11)     | NO   | PRI | NULL    |       |
| company_name    | varchar(50) | YES  | UNI | NULL    |       |
| company_address | varchar(200)| NO   |     | NULL    |       |
| company_tel     | varchar(20) | YES  |     | NULL    |       |
+-----------------+-------------+------+-----+---------+-------+
4 rows in set (0.02 sec)
```

接下来，使用ALTER语句为company_tel字段添加默认约束，SQL语句如下：

```
    ALTER TABLE company MODIFY company_tel VARCHAR(20) DEFAULT '0371
-';
```

为了验证company_tel字段的默认约束是否添加成功，再次使用DESC语句查看company表的结构，执行结果如图所示。

```
mysql> DESC company;
+-----------------+-------------+------+-----+---------+-------+
| Field           | Type        | Null | Key | Default | Extra |
+-----------------+-------------+------+-----+---------+-------+
| company_id      | int(11)     | NO   | PRI | NULL    |       |
| company_name    | varchar(50) | YES  | UNI | NULL    |       |
| company_address | varchar(200)| NO   |     | NULL    |       |
| company_tel     | varchar(20) | YES  |     | 0371-   |       |
+-----------------+-------------+------+-----+---------+-------+
4 rows in set (0.01 sec)
```

3. 删除默认约束语法格式如下：

```
ALTER TABLE表名MODIFY字段名 数据类型;
```

【例】删除company表的company_tel字段的默认约束。SQL语句如下：

```
ALTER TABLE company MODIFY company_tel VARCHAR(20);
```

为了验证company_tel字段的默认约束是否删除，使用DESC语句查看company表的结构，执行结果如图

```
mysql> DESC company;
+-----------------+--------------+------+-----+---------+-------+
| Field           | Type         | Null | Key | Default | Extra |
+-----------------+--------------+------+-----+---------+-------+
| company_id      | int(11)      | NO   | PRI | NULL    |       |
| company_name    | varchar(50)  | YES  | UNI | NULL    |       |
| company_address | varchar(200) | NO   |     | NULL    |       |
| company_tel     | varchar(20)  | YES  |     | NULL    |       |
+-----------------+--------------+------+-----+---------+-------+
4 rows in set (0.01 sec)
```

从图可以看出，company_tel字段已经没有默认值了。

学生表

列名	数据类型	允许 NULL 值	约束	备注
sno	CHAR(8)	不允许	主键	学号
sname	VARCHAR(10)	不允许		姓名
sgender	CHAR(2)			性别
sbirth	DATE			出生日期
sclass	VARCHAR(20)			班级

教师表

列名	数据类型	允许 NULL 值	约束	备注
tno	CHAR(4)	不允许	主键	工号
tname	VARCHAR(10)	不允许		姓名
tgender	CHAR(2)			性别
tedu	VARCHAR(10)			学历
tpro	VARCHAR(8)		默认为“讲师”	职称

课程表

列名	数据类型	允许 NULL 值	约束	备注
cno	CHAR(4)	不允许	主键	课程号
cname	VARCHAR(40)		唯一约束	课程名
cperiod	INT			学时
credit	DECIMAL(3,1)			学分
ctno	CHAR(4)		是教师表的外键	授课教师工号

选课表

列名	数据类型	允许 NULL 值	约束	备注
cno	CHAR(8)		主键（课程号，学号），其中课程号是课程表的外键，学号是学生表的外键	课程号
sno	CHAR(4)			学号
score	INT			成绩

本章小结

数据库的基本操作：创建数据库、查看数据库、修改数据库、删除数据库。

- 数据表的基本操作：创建数据表、查看数据表、修改数据表、删除数据表。
- 四类数据类型：数值类型、日期/时间类型、字符串（字符）类型、二进制类型。
- MySQL中的五种约束：PRIMARY KEY约束、FOREIGN KEY约束、NOT NULL约束、UNIQUE约束、DEFAUL约束。

小练习：网上书店项目
在此数据库中创建：
会员表、图书表、图书类别表、
订购表！

列名	数据类型	允许 NULL 值	约束	备注
uid	CHAR (4)	不允许	主键	会员编号
uname	VARCHAR(20)			会员昵称
email	VARCHAR(20)			电子邮箱
tnum	VARCHAR(15)			联系电话
score	INT			积分

列名	数据类型	允许 NULL 值	约束	备注
bid	INT	不允许	主键	图书编号
bname	VARCHAR(50)	不允许		图书名称
author	CHAR(8)			作者
price	FLOAT			价格
publisher	VARCHAR(50)			出版社
discount	FLOAT			折扣
cid	INT		图书类别表的外键	图书类别

列名	数据类型	允许 NULL 值	约束	备注
cid	INT	不允许	主键	类别编号
cname	VARCHAR(16)			类别名称

列名	数据类型	允许 NULL 值	约束	备注
bid	INT	不允许		图书编号
uid	CHAR(4)	不允许		会员编号
ordernum	INT		默认值为 1	订购量
orderdate	DATETIME			订购日期
deliverydate	DATETIME			发货日期