第5章　数据库和表的操作

MySQL 数据库是有组织的数据的集合，由 DBMS 统一管理和维护。数据库由包含数据的基本表和其他对象（如视图、索引、存储过程和触发器等）组成，其主要用途是处理数据管理活动产生的信息。表是数据库的主要对象，是存放数据的基本单位。

对数据库和表的操作是开发人员的一项重要工作。本章首先对 MySQL 数据库进行简单介绍，然后以实例的形式介绍数据库的创建、修改和删除操作；接着以在 teaching 数据库中表的操作为例，介绍表的基本操作（包括表的创建、修改和删除操作）、完整性约束的实现，以及表中数据的插入、修改、删除操作等内容。

扫一扫

视频讲解

5.1 MySQL 数据库简介

数据库是存储数据的容器，是 MySQL 存放表、视图和索引等数据库对象的逻辑实体。

组成数据库的逻辑成分称为数据库对象，MySQL 中的逻辑对象主要包括数据表、视图、索引、存储过程、函数、触发器、事件等。

每个 MySQL 都包含两种类型的数据库：系统数据库和用户数据库。

系统数据库存储有关 MySQL 的信息，MySQL 使用系统数据库管理系统，安装 MySQL 时会自动生成，如下面要介绍的 performance_schema、information_schema、mysql 和 sys。而用户数据库由用户建立，如 teaching 教学数据库。MySQL 可以包含一个或多个用户数据库。

1. performance_schema

performance_schema 是内存型数据库，使用 performance_schema 存储引擎，通过事件机制将 MySQL 服务的运行时状态采集并存储在 performance_schema 数据库，主要用于性能分析。

2. information_schema

在 MySQL 中，把 information_schema 看作一个信息数据库，用于存储数据库元数据（关于数据的数据），如数据库名、表名、列的数据类型、访问权限等。

3. mysql

mysql 数据库中包含事件、存储引擎状态、主从信息、日志、时区信息、用户权限配置等信息，如在 mysql 数据库的 user 表中修改 root 用户密码。

4. sys

sys 数据库主要提供一些视图，数据都来自 performance_schema 数据库，主要是让开发者和使用者更方便地查看性能问题。

另外，MySQL 还自动创建了一个叫作 world 的数据库，该数据库中只包括 3 张数据表，分别保存城市、国家和国家使用的语言等内容。

MySQL 还提供了一个样例数据库 sakila，该数据库中共有 16 张表。

5.2 数据库操作

在 MySQL 中，用户可以自己创建数据库，即用户数据库，并且可以对数据库进行修改、删除等操作。

扫一扫

视频讲解

■ 5.2.1 创建数据库

在 MySQL 中可以直接采用命令行方式利用 SQL 语句创建数据库；也可以通过可视化管理工具创建数据库，这种方式既可以采用通过菜单和界面创建数据库，也可以利用 SQL 语句创建数据库。

SQL 提供的数据库创建语句格式如下。

```
CREATE{DATABASE|SCHEMA} [IF NOT EXISTS] database_name
[[DEFAULT] CHARACTER SET charset_name]
[[DEFAULT] COLLATE collation_name];
```

其中，“[]”括起来的为可选语法项；“{}”括起来的为必选语法项；“|”分隔的语法项，只能选择其一。

参数说明：

（1）SCHEMA：在 MySQL 中 SCHEMA 与 DATABASE 一样，也是数据库的意思。

（2）database_name：新数据库的名称。数据库名称在服务器中必须唯一，并且要符合标识符的命名规则。

（3）IF NOT EXISTS：在创建数据库前进行判断，只有该数据库目前尚不存在时才执行 CREATE 语句。

（4）[DEFAULT] CHARACTER SET：指定数据库默认字符集，charset_name 为字符集名称。

（5）[DEFAULT] COLLATE：指定字符集的默认校对规则，collation_name 为校对规则名称。

注意：建立数据库时应尽量显式指定使用的字符集，而不是依赖于 MySQL 的默认设置，否则 MySQL 升级时可能带来很大困扰。一般情况下，将数据库默认字符集设置为 UTF-8 是较好的选择。关于各种字符集的区别，读者可自行查阅相关资料。

1. 命令行方式创建数据库

【例 5-1】 要创建一个名为 teaching 的数据库，可以在 MySQL 命令行客户端窗口输入以下语句。

```
CREATE DATABASE teaching CHARACTER SET utf8mb4;
```

创建结果如图 5-1 所示，使用 SHOW DATABASES 语句显示 MySQL 的所有数据库。

2. 利用可视化管理工具创建数据库

下面以 Navicat 为例分别通过菜单界面和 SQL 语句创建数据库。

1）菜单界面

双击已经建立好的 mine 连接，右击 mine，在弹出的快捷菜单中选择“新建数据库”，弹出“新建数据库”对话框，在“数据库名”文本框输入要创建的数据库的名称，如 teaching，在“字符集”下拉列表框中选择 utf8mb4，然后单击“确定”按钮，即可创建成功，如图 5-2 所示。

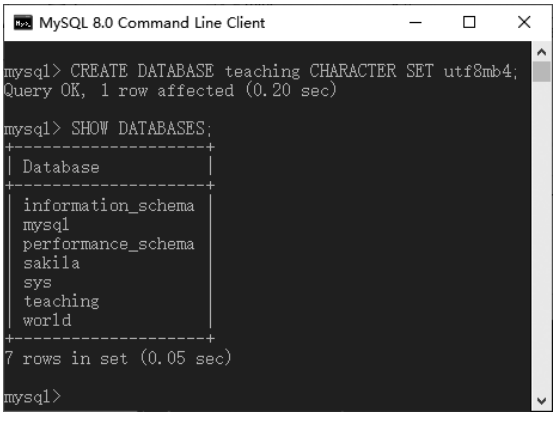

图 5-1 命令行方式创建数据库

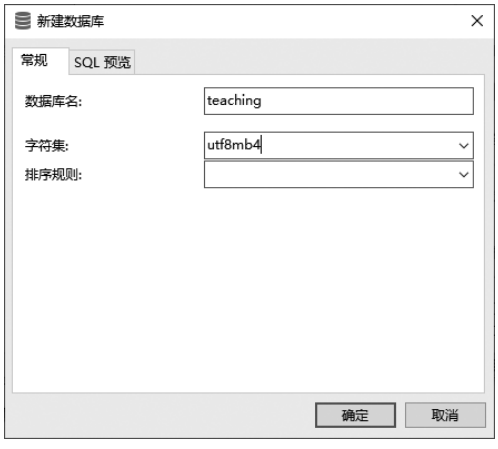

图 5-2 “新建数据库”对话框

在 Navicat 界面的 mine 连接下可以看到已经创建好的 teaching 数据库，右侧窗口显示字符集为 utf8mb4。展开此数据库可以看到其中的表、视图、函数等数据库对象，如图 5-3 所示。

2）SQL 语句

单击 Navicat 工具栏“新建查询”按钮，会出现一个新的查询页面，默认名称为“无标题”，在这个页面中可以输入要让 MySQL 执行的 SQL 语句。

【例 5-2】 输入创建数据库的 SQL 语句。

```
CREATE DATABASE teaching1 CHARACTER SET utf8mb4;
```

单击工具栏中的“执行”按钮，当系统给出的提示信息为 OK 时，说明此数据库创建成功，如图 5-4 所示。

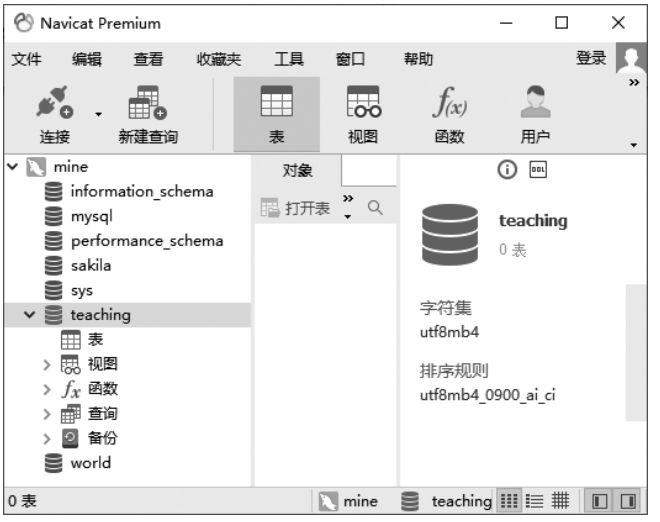

图 5-3　创建数据库成功

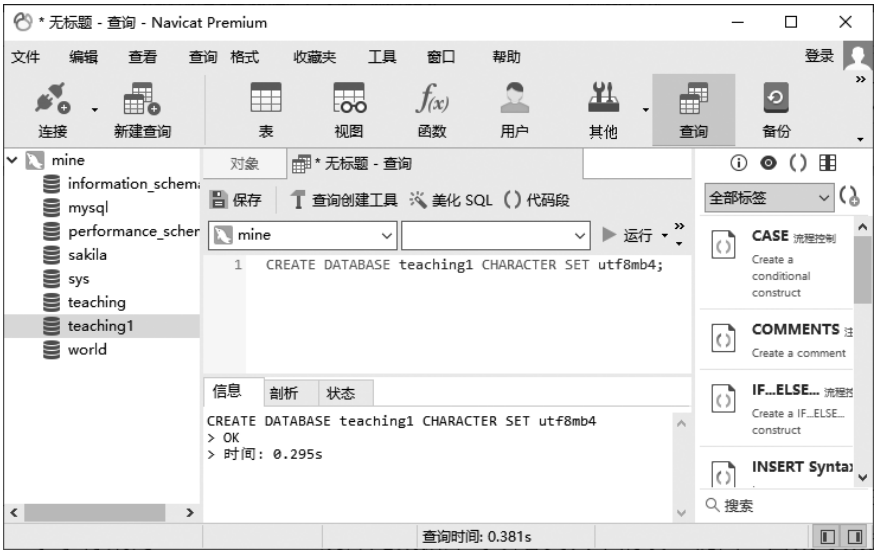

图 5-4　SQL 语句创建数据库

扫一扫

视频讲解

■ 5.2.2　选择和修改数据库

创建好数据库后，此数据库不会自动成为当前数据库，需要使用 USE 语句使其成为当前数据库。语法格式如下。

```
USE database_name;
```

【例 5-3】　选择 teaching 为当前数据库。

```
USE teaching;
```

在命令行客户端或 Navicat 的查询窗口都可以执行以上语句。

命令行客户端的执行结果为 Database changed，Navicat 的执行结果为 OK。

如果要修改数据库的默认字符集或校对规则，可直接通过 Navicat 的“编辑”菜单完成修改。也可以通过命令行客户端或 Navicat 查询窗口执行 SQL 语句完成。

修改数据库的 SQL 语句语法如下。

```
ALTER {DATABASE|SCHEMA} [database_name]
[[DEFAULT] CHARACTER SET charset_name]
[[DEFAULT] COLLATE collation_name];
```

说明：

(1) 数据库名称如果省略，表示修改当前数据库。

(2) 其他参数与创建时相同。

【例 5-4】 修改 teaching1 数据库的默认字符集和校对规则。

```
ALTER DATABASE teaching1
DEFAULT CHARACTER SET gb2321
DEFAULT COLLATE gb2321_chinese_ci;
```

扫一扫

视频讲解

■ 5.2.3 删除数据库

不再使用的数据库可以删除，和创建数据库一样，也可以直接采用命令行方式或通过 Navicat 删除数据库。

1. 命令行方式删除数据库

SQL 提供的数据库删除语句为 DROP DATABASE，其语法格式如下。

```
DROP DATABASE database_name;
```

例如，要删除已创建的数据库 teaching1，就可以在 MySQL 命令行客户端窗口输入以下语句。

```
DROP DATABASE teaching1;
```

2. 利用 Navicat 删除数据库

1) 菜单界面

例如，要删除 teaching1 数据库，首先右击 teaching1，在弹出的快捷菜单中选择“删除数据库”，如图 5-5 所示。

2) SQL 语句

与创建数据库相同，在 Navicat 的查询页面输入 SQL 语句：

```
DROP DATABASE teaching1;
```

单击“执行”按钮即可完成删除。

说明： 用户只能根据自己的权限删除用户数据库；不能删除当前正在使用(正打开供用户读写)的数据库；不能删除系统数据库(performance_schema、information_schema、mysql)，以免影响 MySQL 服务器的使用。

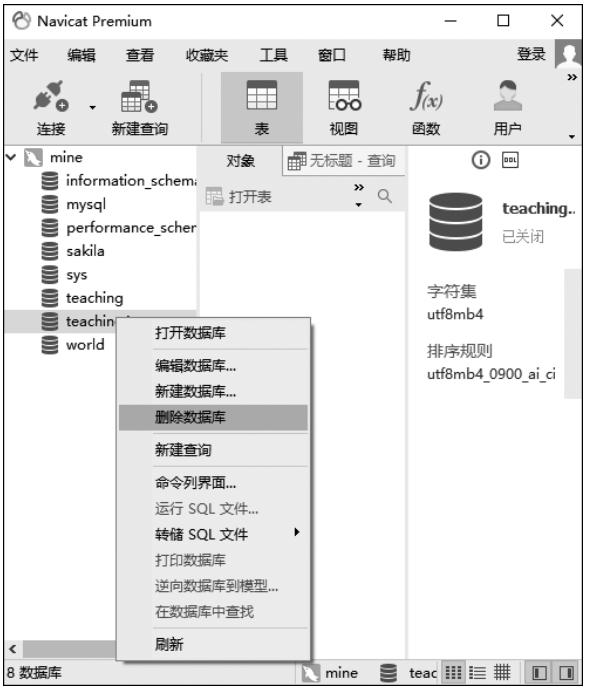

图 5-5　菜单界面方式删除数据库

5.3 创建和修改表

在数据库中，表是由数据按一定的顺序和格式构成的数据集合，是存放数据的基本单位，是数据库的主要对象。表的数据组织形式是行、列结构，表中每行代表一条记录，每列代表记录的一个字段，没有记录的表称为空表。每张表通常都有一个主关键字（又称为主码），用于唯一地确定一条记录。在同一张表中不允许有相同名称的字段。

创建好数据库后，数据库是空的，逻辑上就像建造了一个空的房子（仓库），存入数据后，才成为真正的数据库。对于关系数据库，用于存储数据的当然是关系表，所以首先要在空数据库中创建表。

注意：表必须建在某一数据库中，不能单独存在，也不能以操作系统文件形式存在。

扫一扫

视频讲解

■ 5.3.1　数据类型

我们定义数据表的字段、声明程序中的变量等时，都需要为它们设置一个数据类型，目的是指定该字段或变量所存放的数据是字符串型、整型、定点型、浮点型、日期时间型或其他类型的数据，以及会用多少空间存储数据。

数据类型决定了数据的存储格式，代表了各种不同的信息类型。MySQL 提供系统数据类型集，该类型集定义了可与 MySQL 一起使用的所有数据类型。MySQL 中的数据类型是预先定义好的，可以直接使用。

1. 字符串型

字符串数据类型包括 char、varchar、text，它们的取值都是由任何英文字母、符号、数字、汉字等任意组合而成的数据。

1）定长字符串 char(n)

按固定长度存储字符数据，字符个数不满 n 时自动补空格，查询时会去掉空格。n 的取值范围为 0～255，最大存储字节数为 nw，w 为单个字符所占用的字节数，与字符的编码标准有关，如字符编码为 utf8mb4，每个字符占 4 字节，即 $w=4$。最大存储字节数不能超过 65535。

2）变长字符串 varchar(n)

按变长存储字符数据，存储大小为输入数据的字节的实际长度，若输入的字符个数超过 n，则截断后存储。因为该类型最大存储字节数为 65535，所以 n 的最大值与字符的编码标准有关，如 utf8，每个字符占 3 字节，n 的取值范围为 0～21845；而 utf8mb4，每个字符占 4 字节，n 的取值范围为 0～16383。在处理变长字符串数据时，MySQL 会把数据内容和数据长度都存储起来，这些额外的“长度字节”前缀，会被当作无符号整数来对待。

char 类型的字符串查询速度快，但为了节省存储空间，当有空值或字符串数据长度不固定时可以使用 varchar 数据类型。

3）文本字符串

文本字符串用于存储可变长文本。如果是长度无法确定的字符串数据，如表单数据的收集，可以采用表 5-1 中的某个文本数据类型。

表 5-1 文本字符串类型

数据类型	最大字符数	适用情况	数据类型	最大字符数	适用情况
tinytext	255	短文本数据存储	mediumtext	$2^{24}-1$	中等长度文本数据存储
text	65535	一般长度文本数据存储	longtext	$2^{32}-1$	长文本数据存储

字符串类型中 char 与 varchar 使用频率最高，它们的区别在于是否为定长字符串，在描述某一确定值时，如身份证号，在我国固定为 18 位，可使用 char；如姓名，字符长度不一，可使用 varchar。

text 文本字符串类型可以存储比较大的文本段，搜索速度稍慢，因此如果不是特别大的内容，建议使用 char 或 varchar 来代替。而且，text 类型的数据删除后容易导致“空洞”，使文件碎片比较多，所以频繁使用的表不建议包含 text 类型字段，建议独立出来单独使用一张表。

2. 整型

(1) bigint(大整数)：-2^{63}～$2^{63}-1$ 的整型数据，存储大小为 8 字节。

(2) int(整型)：-2^{31}～$2^{31}-1$ 的整型数据，存储大小为 4 字节。

(3) smallint(短整型)：−32768～ 32767 的整型数据，存储大小为 2 字节。

(4) tinyint(微短整型)：−128～127 的整型数据，存储大小为 1 字节。

(5) bit(位)：只存储 NULL、0 或 1，只占据 1 位空间。bit 数据类型非常适用于开关标

记，在大多数应用程序中被转换为 TRUE 或 FALSE，非 0 为 TRUE，0 为 FALSE。

3. 定点型

decimal（精确数值型）数据由整数部分和小数部分构成，所有数字都是有效位，能够以完整的精度存储十进制数。

表达方式：decimal[(*m*,*d*)]，*m* 指定存放数据的总数字个数，*d* 指定可放到小数点右边的小数位数，*m* 可指定的范围为 1～65，*d* 可指定的范围为 0～30，而且一定要小于 *m*。

例如，decimal(8,6)取值范围是－99.999999～99.999999。

4. 浮点型

浮点型数据包含 float 和 double 两种，与定点型 decimal 相比是不精确类型。

(1) 单精度浮点型 float[(*m*,*d*)]：*m* 代表可以使用的数字位数，*d* 代表小数点后的小数位数。float 数据的取值范围为－3.402823466E＋38～3.402823466E＋38。

(2) 双精度浮点型 double[(*m*,*d*)]：*m* 代表可以使用的数字位数，*d* 代表小数点后的小数位数。double 数据的取值范围比 float 数据大，为－1.7976931348623157E＋308～1.7976931348623157E＋308。

存储 float 数据使用 4 字节，double 数据使用 8 字节。

另外，decimal(*m*,*d*)和 float、double 数据的区别在于，float、double 数据在不指定 *m*、*d* 时默认按照实际精度来处理，而 decimal 数据在不指定 *m*、*d* 时默认为 decimal(10，0)。

5. 日期时间型

MySQL 支持 5 种形式的日期时间类型：date（日期）、time（时间）、year（年）、datetime（日期时间）、timestamp（时间戳），如表 5-2 所示。

表 5-2　日期时间类型

数据类型	字节数	格　式	取值范围
date	3	YYYY-MM-DD	1000-01-01—9999-12-31
time	3	HH:MM:SS	－838:58:59—835:59:59
year	1	YYYY	1901—2155
datetime	8	YYYY-MM-DD HH:MM:SS	1000-01-01 00:00:00—9999-12-31 23:59:59
timestamp	4	YYYY-MM-DD HH:MM:SS	1970-01-01 00:00:00—2037 年某个时间

简单介绍一下 datetime 与 timestamp 的区别：由于所占存储空间的不同，datetime 与 timestamp 能存储的日期时间范围也不同。另外，datetime 默认值为空，当插入的值为 NULL 时，该列的值就是 NULL；timestamp 默认值不为空，当插入的值为 NULL 时，MySQL 会取系统当前日期时间。datetime 存储的时间与时区无关，timestamp 存储的时间及显示的时间都依赖于当前时区。

6. 二进制类型

1）binary(*n*)和 varbinary(*n*)

binary 和 varbinary 类型类似于 char 和 varchar 类型，但不同的是，它们存储的不是字符串，而是二进制串。binary 与 varbinary 唯一的差别在于，当 binary 类型输入的数据长度小于 *n* 时会补\0。

2）blob

blob 数据类型是一个二进制大对象，可以容纳可变长数据，可以存储数量很大的二进制数据，如图片、音频、视频等。

blob 类似于 text，也有 4 种类型：tinyblob、blob、mediumblob 和 longblob。它们之间的区别只是可存储二进制数据值的最大长度不同。

7. 枚举和集合类型

1）enum 类型

enum 类型定义的是枚举集合，赋给 enum 列的值只能是在创建表时指定的值列表中的一个成员。枚举类型通常用于表示类别值，如对于某个定义为 enum ('N', 'Y')的列，其值可以是'N'或 'Y'。另外，也可以将 enum 类型用于表示某种产品的尺寸、颜色，或者用于表示某次调查问卷中单选题的答案等。例如，size enum('S','M','L','XL','XXL','XXXL')表示服装尺寸字段的取值范围。

枚举列表值所允许的成员值从 1 开始编号，MySQL 内部存储的就是这个索引编号，枚举最多可以有 65535 个元素。enum 列总有一个默认值。如果将 enum 列声明为 NULL，NULL 值则为该列的一个有效值，并且默认值为 NULL。如果 enum 列被声明为 NOT NULL，其默认值为允许的值列表的第 1 个元素。

2）set 类型

set 类型叫作集合类型，与 enum 类型有相似之处，如在创建 set 列时，同样需要为它指定一个所允许的集合成员列表。set 类型最多允许有 64 个成员，set 值在内部也是用整数表示，列表中每个值也都有一个索引编号。

与 enum 类型不同的是，赋给 set 列的值都可以由集合的任何成员构成，可以包含成员中的零个或多个值。如果有一组固定值，但它们不是互斥的，这时就可以使用 set 类型。指定包括多个 set 成员的值时，各成员之间用逗号隔开，所以不要把一个包含逗号的字符串用作 set 成员。

如果插入 set 字段中的值有重复，MySQL 自动删除重复的值；插入 set 字段的值的顺序不重要，MySQL 会在存入时按照定义的顺序显示。

8. 其他数据类型

除了前面介绍的数据类型之外，MySQL 还提供了 geometry（几何）、point（点）、linestring(线段)、polygon(多边形)以及 JSON 等数据类型。

扫一扫

视频讲解

■ 5.3.2 创建表

对于具体的某个表，在创建之前，需要确定表的以下特征。

（1）表要包含的数据。

（2）表中的列数，每列中数据的类型和长度，哪些列允许空值。

（3）是否要使用以及何处使用约束、默认值设置。

（4）所需索引的类型，哪里需要索引，哪些列是主键，哪些列是外键。

MySQL 创建表的语法格式如下。

```
CREATE[TEMPORARY] TABLE [IF NOT EXISTS] table_name
( { <column_definition>|<index_definition>| <table_constraint>} [, ...n] )
[table_option] [SELECT_statement];
```

参数说明：

（1）TEMPORARY：表示新建的为临时表。不加该关键字创建的表通常称为持久表，持久表一旦创建将一直存在，多个用户或多个应用程序可以同时使用持久表。有时需要临时存放数据，如临时存储复杂的 SELECT 语句的结果。用户可以像操作持久表一样操作临时表，只不过临时表只对创建它的用户可见，当断开与该数据库的连接时，MySQL 会自动删除它们。

（2）IF NOT EXISTS：创建表时也可使用 IF NOT EXISTS 语句判断创建的表是否已存在，避免出现错误。

（3）table_name：用于指定新建表的名称。表名必须符合标识符规则而且必须是唯一的。

（4）table_option：表的选项，包括存储引擎、默认字符集等，这里的默认字符集如果省略，就按创建数据库时的设置。

（5）SELECT_statement：查询语句，用于从其他表查询出的数据定义表。

上述创建表的语法中＜column_definition＞（列定义）包含的内容如下。

```
<column_definition>::={ column_name data_type }
[ <column_constraint>] [ ...n ]
```

其中，＜column_constraint＞（列约束）包含的内容如下。

```
<column_constraint>::={[NULL | NOT NULL]
  | [PRIMARY KEY | UNIQUE]
  | [CONSTRAINT constraint_name] [CHECK (logical_expression)]
  | [DEFAULT default_expression]}
```

参数说明：

（1）column_name：列名。

（2）data_type：数据类型。

（3）CONSTRAINT constraint_name：为列约束命名。

（4）NULL 和 NOT NULL：如果表的某列被指定具有 NULL 属性，那么就允许在插入数据时省略该列的值；反之，如果表的某列被指定具有 NOT NULL 属性，那么就不允许