第5章 MySQL数据表管理

本章要点

- 掌握 MySQL 支持的各种数据类型的特点，重点掌握数值数据类型、字符数据类型和日期时间数据类型。在此基础上，能够在创建表时正确选择列的数据类型。
- 理解表的结构并能够根据需要设计表。
- 理解主键和自增列的特点并掌握它们的使用方法。
- 掌握定义表的 SQL 语句的使用方法，包括创建表、修改表和删除表。
- 掌握操作表中数据的方法，包括插入数据、更新数据和删除数据。

注意：运行脚本文件 Chapter5-booksale.sql 创建数据库 booksale。本章例题均在该数据库下运行。

数据表是数据库中最重要的数据库对象之一，是实际数据存储的地方。其他的数据库对象，如视图、索引等都是依赖于数据表对象而存在的。数据表的操作是 MySQL 数据库的基础操作之一，数据表质量是直接影响数据库性能的重要因素之一。数据表中的数据以一条条记录的形式存在，对记录的增加、删除、修改和查询为数据表记录的四大操作。


视频讲解

5.1 MySQL 支持的数据类型

数据类型是 MySQL 的重要组成部分，表中的每个列、局部变量、参数等都离不开数据类型的支持。以表中的列为例，数据类型是列的属性，决定了列中可以存放的数据的类型，甚至决定了列的取值范围。给列选择数据类型是创建表的关键步骤之一，所以理解数据类型对创建表非常重要。

5.1.1 数值数据类型

数值数据类型以“数字”的形式存储数据，主要包括整数数据类型、实数数据类型、浮点数据类型、二进制位数据类型和逻辑数据类型。

1. 整数数据类型

整数数据类型简称整型，用于存储整数。根据所存储数据的范围不同，MySQL 提供了如表 5-1 所示的五种整数数据类型。

表 5-1 整数数据类型

数 据 类 型	存储	带符号数值域	无符号数值域
TINYINT[(m)][UNSIGNED | SIGNED][ZEROFILL]	1B	−128～127	0～255
SMALLINT[(m)][UNSIGNED | SIGNED][ZEROFILL]	2B	−32768～32767	0～65535
MEDIUMINT [(m)][UNSIGNED | SIGNED][ZEROFILL]	3B	−8388608～8388607	0～16777215
INT [(m)][UNSIGNED | SIGNED][ZEROFILL]	4B	$-2^{31}\sim 2^{31}-1$	0～4294967295
INTEGER [(m)][UNSIGNED | SIGNED][ZEROFILL]			
BIGINT [(m)] [UNSIGNED | SIGNED] [ZEROFILL]	8B	$-2^{63}\sim 2^{63}-1$	0～18446744073709551615

说明如下。

- m 表示数据的显示宽度,每种整数数据类型都有默认的显示宽度。如果实际数据的位数超过了显示宽度,那么这个数仍然可以正确存储并且可以显示所有位。
- UNSIGNED 表示定义无符号数; SIGNED 表示定义带符号数。默认情况下,变量的符号属性为 SIGNED。
- ZEROFILL 参数表示数字不足的显示空间由 0 来填充。使用 ZEROFILL 参数时,变量自动增加 UNSIGNED 属性。

📖**提示**:从 MySQL 8.0.17 开始,数值数据类型不推荐使用 ZEROFILL 属性,在未来的 MySQL 版本中将删除对它的支持。

从表 5-1 中可以看到,INT 类型和 INTEGER 类型的存储和值域都是一样的,其实,在 MySQL 中 INT 类型和 INTEGER 类型是一样的。

列选择哪个整数数据类型取决于该列的范围。如果列的最大值不超过 255,选择 TINYINT UNSIGNED 就足够了。

【例 5-1】 查看 INT 的数据范围。

```
help INT;
```

执行结果如下所示。

```
Name: 'INT'
Description:
INT[(M)] [UNSIGNED] [ZEROFILL]
A normal - size integer. The signed range is - 2147483648 to 2147483647.
The unsigned range is 0 to 4294967295.
URL: https://dev.mysql.com/doc/refman/8.0/en/numeric - type - syntax.html
```

2. 实数数据类型

实数数据类型不仅可以存储整数,也可以存储小数,并且每一位数字都是精确存储的。MySQL 提供了如表 5-2 所示的实数数据类型。

表 5-2 实数数据类型

数据类型	存储情况
DECIMAL[(m[,d])][UNSIGNED | SIGNED][ZEROFILL]	所占用字节数是 m+2 存储数据的范围取决于 m 和 d 的定义
DEC[(m[,d])][UNSIGNED | SIGNED][ZEROFILL]	
NUMERIC[(m[,d])][UNSIGNED | SIGNED][ZEROFILL]	

说明如下。

- DEC 和 NUMERIC 是 DECIMAL 的别名。
- m(精度)：最多可以存储的十进制数字的总位数，包括小数点左边和右边的位数。m 必须是 1～65 之间的值，默认值为 10。
- d(标度)：小数部分的最大位数。从 m 中减去此数字可确定整数部分的最大位数。d 必是 0～30 之间的值，但是不能超过 m，d 的默认值是 0。精度与标度之间的关系是 0≤d≤m。

例如：DECIMAL (12,4)定义一个实数数据类型的变量，该变量可以存放 12 位数字，其中小数点后面可以存放 4 位小数。

3. 浮点数据类型

浮点数据类型的变量用于存储数的大致数值。浮点数据为近似值，因此它并不能精确地表示数据类型范围内的所有值。根据所存储数据的范围不同，MySQL 提供了如表 5-3 所示的两种浮点数据类型。

表 5-3 浮点数据类型

数据类型	存储	值域
FLOAT[(m,d)][UNSIGNED | SIGNED][ZEROFILL]	4B	-3.402823466E+38～-1.175494351E-38、1.175494351E-38～3.402823466E+38 和 0
DOUBLE[(m,d)][UNSIGNED | SIGNED][ZEROFILL]	8B	-1.7976931348623157E+308～-2.2250738585072014E-308、2.2250738585072014E-308～1.7976931348623157E+308 和 0

说明如下。

- m 表示数的总位数，d 表示小数位数。
- 如果没有定义 m 和 d，则其位数取决于计算机的硬件及操作系统。

提示：由于浮点数据类型的变量存储的数据是不精确的，所以在设计数据库表时很少使用它们作为列的数据类型。

4. 二进制位数据类型

二进制位数据类型变量用于存放二进制数据，设置的基本形式如下所示。

```
BIT[(m)]
```

提示：m 表示二进制数的位数(比特数)，其范围是 1～64(即 1～64bit)，默认值是 1，即存放 1bit 数据。

5. 逻辑数据类型

逻辑数据类型变量用于存放逻辑数据 0 或 1,0 表示假(FALSE),1 表示真(TRUE)。设置的基本形式如下所示。

```
BOOL | BOOLEAN
```

逻辑类型又被称为布尔类型。实际上,布尔类型 BOOL 或 BOOLEAN 的功能等同于微整型 TINYINT(1)。

5.1.2 字符数据类型

字符数据类型是设计数据库表时经常会使用到的数据类型。MySQL 提供了如表 5-4 所示的字符数据类型。

表 5-4 字符数据类型

数据类型	存储	作用
CHAR[(m)]	0～255B	存储固定宽度的字符数据,m 表示字符个数,默认值为 1
VARCHAR(m)	0～65535B	存储可变宽度的字符数据,m 表示字符个数
BINARY (m)	0～255B	存储固定宽度的二进制数据,m 表示数据的字节数
VARBINARY (m)	0～65535B	存储可变宽度的二进制数据,m 表示数据的字节数
TINYBLOB	0～255B	存储最大容量为 255B 的二进制数据
BLOB [(m)]	0～65535B	存储最大容量为 65535B(约 64KB)的二进制数据,可以通过 m 指定最大存储容量
MEDIUMBLOB	0～(2^{24}－1)B	存储最大容量为(2^{24}－1)B(约 16MB)的二进制数据
LONGBLOB	0～(2^{32}－1)B	存储最大容量为(2^{32}－1)B(约 4GB)的二进制数据
TINYTEXT	0～255B	存储最大容量为 255B 的字符数据
TEXT [(m)]	0～65535B	存储最大容量为 65535B(约 64KB)的字符数据,可以通过 m 指定最大存储容量
MEDIUMTEXT	0～(2^{24}－1)B	存储最大容量为(2^{24}－1)B(约 16MB)的字符数据
LONGTEXT	0～(2^{32}－1)B	存储最大容量为(2^{32}－1)B(约 4GB)的字符数据
ENUM	1B 或 2B	以列表形式存储的枚举型,列表中最多有 65535 个值,只能取出列表中的一个枚举字符串值
SET	1B、2B、3B、4B 或 8B	以列表形式存储的集合,可以取列表中的一个成员或多个成员(最多由 64 个成员构成)的组合

1. CHAR 和 VARCHAR

CHAR 和 VARCHAR 类型都用于存储字符数据,但是它们存储以及检索的方式不同。

- CHAR 类型最多允许存储 255B 的字符数据;VARCHAR 类型最多允许存储 65535B(约 64KB)的字符数据。
- CHAR 类型所占存储空间的大小是固定的,当实际存储字符数据不足以占满时,将在右侧使用空格补齐,而检索该字符数据时自动去除右侧空格;VARCHAR 类型所占存储空间的大小是根据实际所存储数据变化的。两种数据类型的比较如表 5-5 所示。

表 5-5 固定宽度和可变宽度数据类型的比较

类 型	数据类型	字符数据	存储情况	实际占用的存储空间
固定宽度	CHAR(4)	'OK'	'OK '	4B
可变宽度	VARCHAR(4)	'OK'	'OK'	2B+1B(前缀)

提示：存储字符数据的数据类型需要设置字符集(character set)和排序规则(collate)属性。在设计 MySQL 数据库表时，CHAR 和 VARCHAR 常被用作字符型列的数据类型。当某一列的数据值宽度相对固定时，应该使用 CHAR 类型；而当某一列数据值的宽度差别较大时，应该使用 VARCHAR 类型。

2. BINARY 和 VARBINARY

BINARY 和 VARBINARY 数据类型用于存储二进制数据，即它们存储字节字符串，而不是字符串。其中 BINARY 类型为固定宽度，最多存储 255B 二进制数据；而 VARBINARY 类型为可变宽度，最多存储 65535B(约 64KB)二进制数据。

3. BLOB 和 TEXT

BLOB(Binary Large Object，二进制大对象)类型用于存储二进制类型的数据。根据最大存储容量的不同，具体分为 TINYBLOB、BLOB、MEDIUMBLOB 和 LONGBLOB 四种类型。

TEXT 类型用于存储字符类型的数据。根据最大存储容量的不同，具体分为 TINYTEXT、TEXT、MEDIUMTEXT 和 LONGTEXT 四种类型。

与其他数据类型不同的是，每一个 BLOB 或 TEXT 类型的值都作为独立的对象存在。

4. ENUM

ENUM 类型又称为枚举类型，设置的基本形式如下所示。

```
ENUM('值 1','值 2', ..., '值 n')
```

取值时只能取列表中的一个元素。其取值列表中最多能有 65535 个值。列表中的每个值独有一个顺序排列的编号，MySQL 中存入的是这个编号，而不是列表中的值。若 ENUM 类型加上 NOT NULL 属性，其默认值为取值列表的第一个元素，若未加上 NOT NULL 属性，其默认值为 NULL。

5. SET

SET 类型又称为集合类型，设置的基本形式如下所示。

```
SET('值 1','值 2',..., '值 n')
```

取值时可以取列表中的一个成员或多个成员的组合。取多个成员时，不同成员之间用逗号隔开。SET 类型的值最多只能是由 64 个成员构成的组合。

5.1.3 日期和时间数据类型

MySQL 提供了如表 5-6 所示的五种日期和时间数据类型。

表 5-6　日期和时间数据类型

数据类型	存　储	值　域	格　式	功　能
DATE	3B	1000-01-01～9999-12-31	YYYY-MM-DD	存储日期值
TIME	3B	－838:59:59～838:59:59	HH:MM:SS	存储时间值
YEAR	1B	1901～2155	YYYY	存储年份值
DATETIME	8B	1000-01-01 00:00:00～9999-12-31 23:59:59	YYYY-MM-DD HH:MM:SS	存储日期时间值
TIMESTAMP	4B	1970-01-01 00:00:00～2037 年某时	YYYYMMDD HHMMSS	存储日期时间值,时间戳

1. 日期和时间数据的零值

每个日期时间类型都有一个有效值范围,当为日期时间变量指定一个不合法的日期时间值时,系统会报错,并将使用“零”值。不同日期与时间类型有不同的零值,如表 5-7 所示。

表 5-7　日期和时间数据的零值

数据类型	DATE	TIME	YEAR	DATETIME	TIMESTAMP
零值	0000-00-00	00:00:00	0000	0000-00-00 00:00:00	0000-00-00 00:00:00

2. 时间类型

类型 TIME 用于存储时间。时间可以是一天中的某个时刻,已经过去的某个时刻或是两个事件之间的时间间隔。所以,TIME 类型变量的存储范围超出 24 小时的时间范围。

时间数据可以在秒的部分使用 6 位小数(精确到毫秒),所以时间数据的表示范围是－838:59:59.000000～838:59:59.000000。

3. 时间戳类型

使用 TIMESTAMP 类型定义表的列,可以实现插入记录或更新记录时自动将该列的值刷新为系统当前日期时间的功能。

(1) 在插入记录或更新记录时都刷新时间戳列的值: TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP。

(2) 在插入记录时更新时间戳的值,而更新记录时不刷新: TIMESTAMP DEFAULT CURRENT_TIMESTAMP。

(3) 在插入记录时将时间戳的值设置为 0,更新记录时刷新时间戳的值: TIMESTAMP ON UPDATE CURRENT_TIMESTAMP。

(4) 在插入记录时将时间戳的值设置为给定值,更新记录时刷新时间戳的值: TIMPSTAMP DEFAULT 'yyyy-mm-dd hh:mm:ss' ON UPDATE CURRENT_TIMESTAMP。

5.1.4　Spatial 数据类型

Spatial 数据即空间数据,又称为几何数据,用来表示物体的位置、形态、大小分布等各方面的信息,是对现实世界中存在的具有定位意义的事物和现象的定量描述。

开放地理空间信息联盟简称为 OGC,发布了空间数据文档。遵循此文档,MySQL 实现了空间扩展。作为几何类型 SQL 环境的子集,该扩展空间实现了空间特性的生成、存储和分析。MySQL 包含的空间数据类型有几何体(GEOMETRY)、点(POINT)、线(LINESTRING)和多

边形(POLYGON),其中几何体可以存储任何类型的几何数据,而其他三种只能存储对应类型的几何数据。

另外,MySQL 还包含其他集合类型的空间数据类型:多点(MULTIPOINT)、多线(MULTILINESTRING)、多边形(MULTIPOLYGON)以及几何集合(GEOMETRYCOLLECTION)。

5.1.5 JSON 数据类型

JSON 数据类型是一种轻量级的数据交换格式。MySQL 中,直至 5.7.8 版本才正式引入 JSON 数据类型。在此之前如果想在表中保存 JSON 格式类型的数据,则需要依靠 VARCHAR 或 TEXT 之类的数据类型。

JSON 数据类型存储时会做格式检验,不满足 JSON 格式会报错,JSON 数据类型默认值不允许为空。在低于 5.7.8 版本的数据库中使用 JSON 类型来建表,显然是不会成功的。

1. JSON 格式数据

JSON 格式数据包括 JSON 数组、JSON 对象、JSON 数组和对象的嵌套。

JSON 数组是包括在方括号"[]"之间,并以逗号","分隔开的值列表,例如:

```
["abc", 10, null, true, false]
```

JSON 对象是包括在大括号"{}"之间,并以逗号","分隔开的"名称/值"对(键值对)的集合,例如:

```
{"id":1,"name":"Tom"}
```

还允许在 JSON 数组元素和 JSON 对象键值内进行嵌套,例如:

```
[99, {"id": "HK500", "cost": 75.99}, ["hot", "cold"]]
{"k1": "value", "k2": [10, 20]}
```

在 MySQL 中,JSON 值是以字符串形式写入的,写入时,MySQL 会对字符串进行解析,以保证写入的格式正确无误。

2. JSON 函数

JSON 类型支持 SQL 函数,当前 MySQL 支持的 JSON 函数如表 5-8 所示。

表 5-8 JSON 函数

名 字	描 述	引 入	弃 用
->	JSON 列路径运算符,即从 JSON 列返回值;相当于 JSON_EXTRACT()		
->>	增强的 JSON 列路径运算符,即从 JSON 列返回值并取消引用;相当于 JSON_UNQUOTE(JSON_EXTRACT())	5.7.13	
JSON_ARRAY()	创建 JSON 数组		

续表

名　字	描　述	引　入	弃　用
JSON_ARRAY_APPEND()	向 JSON 文档附加数据		
JSON_ARRAY_INSERT()	插入到 JSON 数组		
JSON_CONTAINS()	JSON 文档是否包含路径上的特定对象		
JSON_CONTAINS_PATH()	JSON 文档是否包含路径上的任何数据		
JSON_DEPTH()	JSON 文档的最大深度		
JSON_EXTRACT()	从 JSON 文档返回数据		
JSON_INSERT()	向 JSON 文档插入数据		
JSON_KEYS()	返回 JSON 文档的键数组		
JSON_LENGTH()	返回 JSON 文档的元素个数		
JSON_MERGE()	合并 JSON 文档,保留重复的键值。JSON_MERGE_PRESERVE()的弃用同义词		5.7.22
JSON_MERGE_PATCH()	合并 JSON 文档,替换重复的键值	5.7.22	
JSON_MERGE_PRESERVE()	合并 JSON 文档,保留重复的键值	5.7.22	
JSON_OBJECT()	创建 JSON 对象		
JSON_PRETTY()	以可读模式打印 JSON 文档	5.7.22	
JSON_QUOTE()	引用 JSON 文档		
JSON_REMOVE()	从 JSON 文档中删除数据		
JSON_REPLACE()	替换 JSON 文档中的值		
JSON_SEARCH()	JSON 文档中的值路径		
JSON_SET()	向 JSON 文档插入数据		
JSON_STORAGE_SIZE()	用于存储 JSON 文档的二进制表示的空间	5.7.22	
JSON_TYPE()	JSON 值的类型		
JSON_UNQUOTE()	取消引用 JSON 值		
JSON_VALID()	验证 JSON 值是否有效		


视频讲解

5.2　数据表操作

由于数据表是数据库中实际存储数据的对象,所以数据表操作是数据库操作中最基础和最重要的操作。表的质量直接影响着数据库的性能,并且表一旦创建就不应该随意修改,因此在创建表之前必须对表进行设计和评估。

5.2.1　表的概念

图 5-1 是图书销售数据库 booksale 中存放的图书表 books。

1. 表的结构

表的结构也称为“型”(Type),用于描述存储于表中的数据的逻辑结构和属性。定义表就是指定义表的结构,使用数据定义语言来实现。在定义表之前首先需要注意以下几个概念。

(1) 表名:在同一个数据库中,每一个表都应该有一个唯一的名称。表名和数据库的名字一样,都应该满足标识符命名规则。

bookid	title	isbn	author	unitprice	ctgcode
1	Web前端开发基础入门	978-7-3025-7626-6	张颖	65	computer
2	计算机网络（第7版）	978-7-1213-0295-4	谢希仁	49	computer
3	网络实验教程	978-7-1213-9039-5	张举	32	computer
4	Java编程思想	978-7-1112-1382-6	埃克尔	107	computer
5	托福词汇真经	978-7-5213-2173-9	刘洪波	65.9	language
6	好喝的粥	978-7-5184-1973-9	(Null)	60	life
7	环球国家地理百科全书	978-7-5502-7510-2	张越平	80	life
8	托福考试_冲刺试题	978-7-5619-3674-0	(Null)	40	language
9	狼图腾	978-7-535-42730-4	姜戎	32	fiction
10	战争与和平	978-7-5387-6100-9	列夫·托尔斯泰	188	fiction

图 5-1 图书表 books

（2）列名：从图 5-1 中可以看出，每个表由若干列组成，在同一个表中每个列的名字应该是唯一的，列的名字应该符合标识符命名规则。

（3）列的数据类型：表中的每个列都要定义一个数据类型。定义数据类型时需要慎重考虑，如果定义的范围太小，可能会造成无法存放某些数据，如果定义的范围太大可能会造成存储空间的浪费。存储空间的增加将增加系统的 I/O 操作量，从而降低系统的使用效率。

（4）列中是否允许有空值：表中的某些列可能严禁出现空值，例如，若要求每本图书都必须有图书编号，那么“图书编号”列就不允许有空值。某些列，例如“作者”列中可能会存在空值，也就是说某些图书没有明确作者或作者未知，这时这些列就应该定义成允许空值。

2. 表中的数据

表中的数据也称为“值”（Value），是“型”的具体赋值。操纵表中的数据通过数据操纵语言实现。

（1）数据行：一个数据行也被称为一个元组或一条记录，是现实世界中一个物理或逻辑实体的数据描述形式。

（2）数据列：一个数据列也被称为一个属性或一个字段，是同一类型的所有实体在某个属性上的全部值的集合。列是表定义的基本对象，定义一个表的主要任务就是定义这个表中的各个列。

（3）主键：表的主键是表中的某个列或某几个列的组合，其值可以唯一标识表中的每个行。一个表只能定义一个主键，而且通常都应该定义一个主键。主键的值不能为空值，也不能重复。如果存在多个列或列组合同时满足作为主键的条件，则应该选择运算效率高的列或列组合作为表的主键。通常数值型的列比字符型的列运算效率高；如果同为字符型，则取值范围小的列的运算效率通常更高。

（4）自增列：又称标识列，可以将表中具有整数性质的某个列定义为自增列来唯一标识表中的每一行，定义的关键词为 AUTO_INCREMENT。一个表中最多只能有一个列被定义为自增列。自增列不允许为空值，也不允许重复，自增列必须是主键或主键的一部分。默认情况下自增列中的第一个值是 1，后续值自动加 1。如果用户设置了一个非 1 的初始值，后续值将在该值基础上自动加 1。

提示：系统数据库 information_schema 中的数据表为系统数据表，如：SCHEMATA 表（提供了当前 MySQL 实例中所有数据库的信息，SHOW DATABASES 的结果取自此表）、

TABLES 表(提供了关于数据库中的表的信息,详细表述了某个表属于哪个 schema、表类型、表引擎、创建时间等信息,SHOW TABLES FROM schemaname 的结果取自此表)、COLUMNS 表(提供了表中的列信息,详细表述了某张表的所有列以及每个列的信息,SHOW COLUMNS FROM schemaname. table_name 的结果取自此表)等。

5.2.2 表的创建

1. 创建表

创建表就是在数据库中建立新表。创建表的基本语法格式如下所示。

```
CREATE TABLE [IF NOT EXISTS] table_name(
    column1 DATETYPE [PRIMARY KEY] [AUTO_INCREMENT]
    [,] column2 DATETYPE  [NULL | NOT NULL]
    ......
    [,] columnn DATETYPE  [NULL | NOT NULL]
    [,] [PRIMARY KEY (column1 [, column2] [, ......])
);
```

语法说明如下。

- table_name 是要定义的数据表的表名,可以是字母、数字和下画线组成的任意字符串。在同一数据库中数据表名是唯一的,不可与已经存在的数据表重名。
- IF NOT EXISTS 是可选选项。添加该选项,表示指定的数据表不存在时执行创建数据表操作,否则忽略此操作。
- column 是列的名字; DATATYPE 是该列的数据类型; NOT NULL 表示该列中不允许有空值,NULL 表示该列中允许有空值,为默认选项。
- PRIMARY KEY 用于定义主键。如果是某个列作为主键,则可以直接在该列上定义主键约束; 如果由多个列组成主键,则必须定义表级主键约束,其形式为"PRIMARY KEY (column1 [, column2] [, …])"。
- AUTO_INCREMENT 表示将列定义为自增列。

【例 5-2】 在图书销售数据库 booksale 中创建图书表 books 用于存放图书的信息。

```
USE booksale;
CREATE TABLE books(
    bookid INT NOT NULL,
    title VARCHAR(50) NOT NULL,
    isbn CHAR(17) NOT NULL,
    author VARCHAR(50),
    unitprice DECIMAL(6,2),
    ctgcode VARCHAR(20));
```

定义列时使用 NOT NULL 表示这个列在存储数据时不允许出现空值,否则使用默认的属性 NULL,表示这个列在存储数据时允许出现空值。

如果数据表 books 已经存在,再运行上面的命令,系统会提示错误信息"Table 'books' already exists",为了防止这种错误发生,在创建数据表时可以在"数据表名称"前添加 IF NOT EXISTS,这样命令执行后,只是返回一条警告信息"Query OK, 0 rows affected, 1

warning (0.01 sec)"而已。

【例 5-3】 在图书销售数据库 booksale 中创建顾客表 customers 用于存放顾客的信息。

```
CREATE TABLE customers(
    cstid INT PRIMARY KEY,
    cstname VARCHAR(20) NOT NULL,
    telephone CHAR(11) NOT NULL,
    postcode CHAR(6),
    address VARCHAR(50) NOT NULL,
    emailaddress VARCHAR(50) NOT NULL,
    password VARCHAR(50) NOT NULL);
```

定义列 cstid 时使用 PRIMARY KEY 表示将该列定义为表的主键。定义主键时系统自动将该列定义为 NOT NULL,即不允许空。

【例 5-4】 在图书销售数据库 booksale 中创建订单表 orders 用于存放订单的信息。

```
CREATE TABLE orders(
    orderid INT PRIMARY KEY AUTO_INCREMENT,
    orderdate TIMESTAMP DEFAULT current_timestamp,
    shipdate DATETIME,
    cstid INT NOT NULL);
```

定义列 orderid 时使用 AUTO_INCREMENT 表示将该列定义为自增列,系统会自动在该列中生成不重复的整数序列值。定义列的 AUTO_INCREMENT 属性时必须将该列定义为主键或主键的一部分。

定义列 orderdate 时使用数据类型 TIMESTAMP,并且将默认值设置为 current_timestamp,表示插入记录时系统会自动将系统当前日期时间存入该列中。默认值约束的设置见 6.3.4 节。

【例 5-5】 在图书销售数据库 booksale 中创建订单项目表 orderitems 用于存放订单项目的信息。

```
CREATE TABLE IF NOT EXISTS orderitems(
    orderid INT NOT NULL,
    bookid INT NOT NULL,
    quantity INT NOT NULL,
    price DECIMAL(6,2),
    PRIMARY KEY (orderid, bookid));
```

该表的主键由两列组成,所以这里需要使用表级主键。因为主键所在列都不允许出现空值,所以即使定义主键所在列时没有使用 NOT NULL,系统也会自动为该列增加非空属性。

添加 IF NOT EXISTS 参数,表示要创建的 orderitems 表只有在不存在时,才执行该创建表命令。

2. 创建带 JSON 类型的表

新的数据类型 JSON 的引用可以将复杂数据存储在一个数据列中,易于存储。

【例 5-6】 在图书销售数据库 booksale 中创建带有 JSON 类型的表 t_json 用于存放售货员信息,然后查看数据库中已经存在的数据表。

```
CREATE TABLE t_json(
    id INT NOT NULL AUTO_INCREMENT,
    json_col JSON,
    PRIMARY KEY (id));
SHOW TABLES FROM booksale;
```

该表的主键由一列组成,可以采用列级主键,也可以采用表级主键,这里使用的表级主键。因为主键所在列都不允许出现空值,所以无论该列是否定义 NOT NULL,系统都会自动为该列增加非空属性。

3. 表的复制

使用上述的 CREATE TABLE 命令可以根据实际需要创建表,是实际开发中较常用的方式。而 CREATE TABLE LIKE 命令则可以对源表的模式进行复制,从现有的数据表中精确地复制表的定义(不复制其数据),其创建的表除了表名和源表不一样外,其余所有的细节都是一样的。复制表的基本语法格式如下所示。

```
CREATE [TEMPORARY] TABLE [IF NOT EXISTS] table_name
    LIKE old_table_name | (LIKE old_table_name);
```

语法说明如下。

- table_name 是生成的新表名。
- TEMPORARY 是可选选项,用于创建临时表。临时表仅在当前会话中可见,并在会话关闭时自动丢弃。
- IF NOT EXISTS 是可选选项。添加该选项,表示指定的数据表不存在时执行数据表复制操作,否则忽略此操作。
- LIKE old_table_name 是基于表 old_table_name 的定义创建空表 table_name,包括原始表中定义的任何列属性和索引。该子句可加括号也可不加括号。

【例 5-7】 在图书销售数据库 booksale 中创建和图书表 books 一样结构的临时表图书备份表 booksbak。

```
CREATE TEMPORARY TABLE booksbak LIKE books;
```

booksbak 表和 books 表的结构一模一样。当退出 MySQL 再次登录后,该临时表将不再存在。SHOW TABLES 命令,不能看到临时表。

4. 查看表结构

查看表结构是指查看数据库中已存在的表的定义。查看表结构的语句包括 DESCRIBE 语句和 SHOW CREATE TABLE 语句,通过这两个语句,可以查看表的数据列名、数据列的数据类型和完整性约束条件等。

1) DESCRIBE 语句查看表定义

可以使用 DESCRIBE(可以缩写为 DESC)命令查看表的基本定义,包括数据列的列名、

数据类型、是否为空、是否为主键、默认值、自增列等，其基本语法格式如下所示。

```
DESCRIBE | DESC table_name;
```

【例 5-8】　查看 orders 表的结构。

```
DESC orders;
```

执行结果如图 5-2 所示。

```
mysql> USE booksale;
Database changed
mysql> DESC orders;
+-----------+-----------+------+-----+-------------------+-------------------+
| Field     | Type      | Null | Key | Default           | Extra             |
+-----------+-----------+------+-----+-------------------+-------------------+
| orderid   | int       | NO   | PRI | NULL              | auto_increment    |
| orderdate | timestamp | NO   |     | CURRENT_TIMESTAMP | DEFAULT_GENERATED |
| shipdate  | datetime  | YES  |     | NULL              |                   |
| cstid     | int       | NO   | MUL | NULL              |                   |
+-----------+-----------+------+-----+-------------------+-------------------+
4 rows in set (0.01 sec)
```

图 5-2　查看 Orders 表的结构

2）SHOW CREATE TABLE 语句查看表详细定义

可以使用 SHOW CREATE 命令查看定义表的 SQL 语句，从而得到表的详细结构，包括列的名称、数据类型、是否为空、默认值、表的存储引擎、字符编码等，比使用 DESC 命令显示的信息要全面。SHOW CREATE TABLE 命令的基本语法格式如下所示。

```
SHOW CREATE TABLE table_name;
```

【例 5-9】　查看 books 表的结构。

```
SHOW CREATE TABLE books;
```

执行结果如图 5-3 所示。

```
命令提示符 - mysql -uroot -proot123456
mysql> SHOW CREATE TABLE books;
+-------+--------------
| Table | Create Table
+-------+--------------
| books | CREATE TABLE `books` (
  `bookid` int NOT NULL,
  `title` varchar(50) NOT NULL,
  `isbn` char(17) NOT NULL,
  `author` varchar(50) DEFAULT NULL,
  `unitprice` decimal(6,2) DEFAULT NULL,
  `ctgcode` varchar(20) DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci |
+-------+--------------
1 row in set (0.01 sec)
```

图 5-3　查看 books 表的结构

提示：图 5-3 是在命令提示符下显示的结果，在显示内容较长的情况下，使用“\G”选项可以更好地显示结果。如果在客户端工具 Navicat 中，由于显示列宽度有限，可以将其复制出来查看。

5.2.3 表的修改

修改表是指修改数据库中已存在的表的定义。表创建好以后，可以根据需要使用 ALTER TABLE 语句修改表的结构，包括在表中增加新列、修改列的属性以及删除列等。

1. 增加列

增加新列的基本语法格式如下所示。

```
ALTER TABLE table_name
    ADD [COLUMN] columndefinition [FIRST | AFTER columnname];
```

语法说明如下。

- table_name 是要修改的数据表的表名，该表必须是数据库中已经存在的表。
- ADD COLUMN 是增加新列的命令关键字，其中 COLUMN 关键字可以省略。
- columndefinition 是对新增加列的完整定义。
- FIRST 表示新增加的列作为表的第一列；也可以使用 AFTER columnname 的形式将新增加的列指定到 columnname 所表示的列之后；默认情况下，新增加的列是表的最后一列。

【例 5-10】 在图书表 books 中新增一个新列 press，用于存放出版社名称。该列数据类型为 VARCHAR(50)，允许空值。

```
ALTER TABLE books ADD press VARCHAR(50) NULL;
```

关键词 NULL 表示该列允许空值，由于 NULL 是默认设置，所以该关键词可以省略。也可以通过以下两条语句完成增加列操作。

```
ALTER TABLE books ADD press VARCHAR(50) FIRST;
```

多加了一个关键字 FIRST，表示 press 列在表中第一的位置。

```
ALTER TABLE books ADD press VARCHAR(50) AFTER author;
```

多加了一个关键字 AFTER，表示 press 列在 author 列的后面。

这三条命令添加的列名相同，实操操作完一个命令后，应先删除该列，再继续下一个命令。

提示：如果表中已经有数据，那么在表中增加一个新列时，新列中是没有数据的，所以如果将增加的新列设置成不允许有空值，必然产生错误。可以有两种方法解决这个问题，一种是首先将新列定义成允许有空值，然后向新列中输入数据后再将这个列修改为不允许有空值；另一种是在添加新列时为该列定义一个默认值。

2. 修改列

修改列的基本语法格式如下所示。

```
#语法 1
ALTER TABLE table_name
    MODIFY [COLUMN] columndefinition [ FIRST | AFTER columnname ];
```

语法说明如下。

- table_name 是要修改的数据表的表名，该表必须是数据库中已经存在的表。
- MODIFY COLUMN 是修改列的命令关键字，其中 COLUMN 关键字可以省略。
- columndefinition 是对修改列的完整定义。
- FIRST 表示将修改的列调整为表的第一列；也可以使用 AFTER columnname 的形式将修改的列指定到 columnname 所表示的列之后。

```
#语法 2
ALTER TABLE table_name
    CHANGE [COLUMN] oldcolumnname columndefinition [ FIRST | AFTER columnname ];
```

语法说明如下。

- oldcolumnname 是要修改列的列名。
- columndefinition 是对修改列的完整定义，该定义中列名可以重新命名。

提示：通过该语句不仅可以修改列的属性，也可以修改列的名称。

【例 5-11】 修改图书表 books 中的出版社列 press，将数据类型修改为 VARCHAR(20)，不允许空值，并将位置修改为位于作者列 author 之后。

```
ALTER TABLE books MODIFY press VARCHAR(20) NOT NULL AFTER author;
```

【例 5-12】 修改图书表 books，将图书编号列 bookid 修改为自增、主键列。

```
ALTER TABLE books MODIFY bookid INT PRIMARY KEY AUTO_INCREMENT;
```

【例 5-13】 修改订单表 orders，删除订单编号列 orderid 的自增属性。

```
ALTER TABLE orders MODIFY orderid INT;
```

订单编号列 orderid 的为空性属性和主键属性不变。

【例 5-14】 将图书表 books 中的出版社列 press 的名称改为 publisher，其他属性不变。

```
ALTER TABLE books CHANGE press publisher VARCHAR(20) NOT NULL;
```

3. 删除列

删除列的基本语法格式如下所示。

```
ALTER TABLE table_name
    DROP [COLUMN] columnname;
```

语法说明如下。

- table_name 是要修改的数据表的表名,该表必须是数据库中已经存在的表。
- DROP COLUMN 是删除列的命令关键字,其中 COLUMN 关键字可以省略。
- columnname 是要删除列的列名。

【例 5-15】 删除图书表 books 中的出版社列 publisher。

```
ALTER TABLE books DROP publisher;
```

4. 重命名表

数据库系统通过表名来区分不同的表,表名在同一个数据库中唯一标识一张表。重命名表的基本语法格式如下所示。

```
ALTER TABLE table_name
    RENAME [TO] new_table_name;
```

语法说明如下。

- table_name 是要修改的数据表的表名,该表必须是数据库中已经存在的表。
- RENAME[TO]是重命名表的命令关键字,其中 TO 关键字可以省略。
- new_table_name 是数据表修改后的新表名,该表名在数据库中不能存在。

【例 5-16】 将顾客表 customers 的名称重命名为 users。

```
ALTER TABLE customers RENAME users;
```

数据库 booksale 中 customers 表已经不存在了,取而代之的是 users 表。

5.2.4 表的删除

删除表是指删除数据库中已存在的表。删除表将同时删除表中的数据。因此,删除表操作要想好了再做。创建表时可能存在外键约束,被关联的父表删除比较复杂。这里只讲没有关联的普通表的删除,关联表的删除在讲解外键约束时再讲解。

删除表的基本语法格式如下所示。

```
DROP TABEL [IF EXISTS] table_name[,table_name...]
    [RESTRICT | CASCADE];
```

语法说明如下。

- table_name 是要删除的数据表的表名,可以一次性删除多个数据表。
- IF EXISTS 是可选选项。添加该选项,表示指定的数据表存在时执行删除数据表操作,否则忽略此操作。
- RESTRICT | CASCADE 是可选选项。RESTRICT 是确保只有不存在相关视图和完整性约束的表才能删除。CASCADE 是任何相关视图和完整性约束一并被删除。

【例 5-17】 删除顾客表 users。

```
DROP TABLE users;
```

数据库 booksale 中 users 表必须存在，否则命令执行将提示错误信息“ERROR 1051 (42S02): Unknown table 'booksale. ***'”。

5.3 数据操作


视频讲解

创建表只是创建表的结构，其中并不包含数据，所以表是空的。可以通过 INSERT 语句向表中插入数据，如果数据有错误，还可以使用 UPDATE 语句更新数据，如果某些行是多余的，可以通过 DELETE 语句将其删除。

5.3.1 数据插入

数据插入是常见的数据操作，可以向表中增加新的数据记录。在 MySQL 中使用 INSERT 语句向表中插入行。INSERT 语句的基本语法格式如下所示。

```
INSERT [INTO] table_name [(columnlist)]
VALUES ({expression | NULL} [,...n] ) [,...n];
```

语法说明如下。

- table_name 是要插入数据的数据表的表名。
- columnlist 是表中列的列表，用来指定要向其中插入数据的列，列与列之间用逗号分开。如果向表中所有列插入值，则可以省略列的列表。
- VALUES 用于指出要插入的数据。NULL 表示列中保持空(前提是该列允许空)；expression 是数据表达式，表示将表达式的结果插入到列中，多个数据项之间用逗号分开。可以同时向表中插入多行数据。

向表中插入数据时需要注意以下几点。

- 数据表达式的数据值应该与列表 columnlist 中的列一一对应，并且数据类型也应该兼容。
- 必须为表中所有定义为 NOT NULL 的列提供值，定义为 NULL 的列可以提供值，也可以不提供值。
- 如果表中存在自增列，则系统可以自动维护自增列中的值，不需要向自增列中插入值。
- 主键所在列的值不允许重复，也不允许出现空值。

1. 插入完整数据记录

插入数据记录时可以一次插入一条完整的数据记录。

【例 5-18】 向订单表 orders 中插入第 1 条订单记录。

```
INSERT INTO orders (orderid,orderdate,shipdate,cstid)
    VALUES (1,'2021-4-14','2021-4-17',3);
```

提示：字符型数据和日期时间型数据需要使用单引号(')括起来。

记录插入后，可输入查询语句"SELECT * FROM orders;"进行验证。

【例 5-19】 向订单表 orders 中插入第 2 条订单记录。

```
INSERT INTO orders VALUES (2,'2021-4-15','2021-4-18',1);
```

由于每个列都提供了值，所以可以省略列的列表。

2. 插入多条完整数据记录

插入数据记录时除了可以一次插入一条数据记录外，还可以一次插入多条数据记录。

【例 5-20】 向订单表 orders 中插入第 3～6 条记录。

```
INSERT INTO orders VALUES
    (3,'2021-4-21','2021-4-22',1),(4,'2021-5-13','2021-5-14',2),
    (5,'2021-5-15',NULL,3),(6,DEFAULT,NULL,1);
```

由于第 5 和第 6 张订单还未发货，因此没有发货日期。在插入数据时，需要为该列提供 NULL 表示空。第 6 张订单的订购日期列的数据类型为时间戳且设置了默认值，需要为该列提供 DEFAULT 表示默认值，则在插入记录时，时间戳列的值自动存入。

【例 5-21】 向图书表 books 中插入前两条记录。

```
INSERT INTO books VALUES
    (NULL,'Web前端开发基础入门', '978-7-3025-7626-6', '张颖', 65.00, 'computer'),
    (NULL,'计算机网络(第7版)', '978-7-1213-0295-4', '谢希仁', 49.00, 'computer');
```

由于书号 bookid 是自增列，所以即使在插入语句中没有为该列提供值，系统也自动定义第一本书书号为 1，第二本书书号为 2。若书号为 2 的记录删除后再添加记录，自增列的编号为 3，书号 2 不再出现。

3. 插入部分数据记录

插入数据记录时除了可以插入完整数据记录外，还可以插入指定列的部分数据记录。

【例 5-22】 向订单表 orders 中插入第 7 条订单记录。

```
INSERT INTO orders (orderid,orderdate,cstid) VALUES (7,'2021-5-16',4);
```

该订单还未发货，因此没有发货日期，且该列可以为空，录入时可选择不录入该列，该列系统自动存入 NULL 值。

提示：未指定的列必须是可以为空的列或非空列但设置了默认值的列。若省略的列中包含非空列，且没有设置默认值，则系统将提示错误信息“ERROR 1136 (21S01): Column count doesn't match value count at row 1”。

4. 插入多条部分数据记录

插入数据记录时还可以一次插入多条指定列的部分数据记录。

【例 5-23】 向图书表 books 中插入两条记录。

```
INSERT INTO books(title, isbn) VALUES
    ('德语词汇联想与速记', '978-7-5213-2183-8'),('奇妙博物馆', '978-7-5596-5308-6');
```

books 数据表中只有图书编号 bookid、书名 title 和图书国际标准书号 isbn 三个列是非空列，插入记录时必须输入。其中 bookid 是自增列，系统会自动为其赋值，因此也可以不输入。未输入的列系统自动存入 NULL 值。

5. 插入 JSON 结构的数据记录

【例 5-24】 向表 t_json 插入一条记录。

```
INSERT INTO t_json(json_col) VALUES ('{"name":"王平","gender":"女","regular":true}');
```

JSON 值是以字符串形式写入的，写入时 MySQL 会对字符串进行解析，如果不符合 JSON 格式，将无法写入。在 Navicat 中无法查看 JSON 类型列的值，可输入以下查询语句进行验证。

```
SELECT * FROM t_json;
```

执行结果如图 5-4 所示。

```
+----+-----------------------------------------------+
| id | json_col                                      |
+----+-----------------------------------------------+
|  1 | {"name": "王平", "gender": "女", "regular": true} |
+----+-----------------------------------------------+
1 row in set (0.00 sec)
```

图 5-4 向 t_json 表插入记录

5.3.2 数据更新

数据更新是常见的数据操作，可以更新表中已经存在数据记录中的值。在 MySQL 中使用 UPDATE 语句更新表中的数据。UPDATE 语句的基本语法格式如下所示。

```
UPDATE table_name
    SET columnname = {expression | NULL} [ ,...n]
    [WHERE searchcondition];
```

语法说明如下。

- table_name 是要更新数据的数据表的表名。
- SET 子句中的 columnname 是要更新数据列的列名。expression 是指将表达式的值更新到该列中；NULL 是指将该列设置为空值，即删除该列中的值。另外，在一个 UPDATE 语句中，可以同时更新多个列中的值。
- WHERE 子句是可选选项，其中的 searchcondition 是一个对行进行筛选的表达式，指定要更新哪些行中的数据。如果省略 WHERE 子句，将更新表中所有的行。

1. 更新特定数据记录

根据筛选条件可以将特定的数据记录的列值进行更新。

【例 5-25】 更新图书表 books 中编号为 1 的图书的作者和类别编号。

```
UPDATE books SET author = '张影',ctgcode = 'information'  WHERE bookid = 1;
```

2. 更新所有数据记录

不加筛选条件时可以将所有数据记录的列值进行更新。

【例 5-26】 将所有图书的单价调低为 9 折以后的价格。

```
UPDATE books SET unitprice = unitprice * 0.9;
```

books 表中的 unitprice 列中的值均修改为原有值×0.9 后的值。

3. 更新 JSON 结构的数据记录

可通过 JSON 函数更新 JSON 数据记录,常用的函数有 JSON_ARRAY_APPEND、JSON_ARRAY_INSERT、JSON_INSERT、JSON_MERGE、JSON_MERGE_PATCH、JSON_MERGE_PRESERVE、JSON_REMOVE、JSON_REPLACE、JSON_SET 等。

【例 5-27】 将表 t_json 中 id 为 1 的记录性别修改为“男”。

```
UPDATE t_json SET json_col = JSON_REPLACE(json_col, '$.gender', '男') WHERE id = 1;
```

5.3.3 数据删除

数据删除可以删除表中已经存在的数据记录,可一次删除一行或多行数据。但在删除记录前应进行数据备份,以避免数据的丢失。在 MySQL 中可以用 DELETE 语句删除表中的特定记录或所有记录,也可以用 TRUNCATE TABLE 删除表中的所有记录。

1. 删除特定数据记录

DELETE 语句的基本语法格式如下所示。

```
DELETE FROM table_name
    [WHERE searchcondition];
```

语法说明如下。

- table_name 是要删除数据的数据表的表名。需要特别注意的是,DELETE 语句删除的对象以行为单位,即一次至少删除一行数据。
- WHERE 子句是可选选项,其中的 searchcondition 是一个对行进行筛选的表达式,指定要删除哪些行。如果省略 WHERE 子句,将删除表中所有的行。

【例 5-28】 删除图书表中编号为 1 的图书记录。

```
DELETE FROM books WHERE bookid = 1;
```

2. 删除所有数据记录

TRUNCATE TABLE 语句的基本语法格式如下所示。

```
TRUNCATE TABLE table_name;
```

📖**提示**：TRUNCATE TABLE 和 DELETE 的区别是：DELETE 是一条一条删除表中的数据。TRUNCATE TABLE 是删除原来的表再重新创建一个表，而不是逐行删除表中的数据。其删除速度比 DELETE 快，但该语句不能被撤销。

在实际开发中很少使用 DELETE 语句。删除有物理删除和逻辑删除，其中逻辑删除可以通过给表添加一列(isDel)，若值为 1，代表删除；若值为 0，代表没有删除。此时，对数据的删除操作就变成了 UPDATE 操作。这样可以更好地保护数据安全。

【例 5-29】 删除所有的订单记录。

```
TRUNCATE TABLE orders;
-- 等价于
DELETE FROM orders;
```

5.4 可视化操作指导

📖**注意**：删除数据库 booksale 并重新创建该数据库，然后打开 Navicat for MySQL，连接到数据库服务器。表的结构请参考附录 A。

1. 表的定义

1) 定义图书表 books

(1) 打开 Navicat for MySQL，连接到数据库服务器，右击 booksale 数据库，在弹出的快捷菜单中选择“打开数据库”命令，右击“表”节点，在弹出的快捷菜单中选择“新建表”命令，打开新建表工作页。

(2) 在新打开的工作页中首先给出表中各个列的名称、数据类型以及是否允许空值。

(3) 单击图书编号列 bookid 前面的选择按钮选择该列，单击“主键”按钮 主键，将该列设置为表的主键；单击选中“自动递增”复选按钮，使该列的值可以自增，如图 5-5 所示。

图 5-5 创建图书表 books

(4) 单击“保存”按钮,在弹出的“表名”对话框中将表命名为 books,单击“确定”按钮保存该表。

(5) 关闭创建图书表 books 的工作页。

提示:如果要修改表的结构,在左侧窗口中右击要修改的表,在弹出的快捷菜单中选择“设计表”命令,打开定义表结构工作页,在该工作页中直接修改表结构即可。修改表结构以后单击“保存”按钮保存对表的修改,然后关闭该工作页。

如果要删除表,在左侧窗口中右击要删除的表,在弹出的快捷菜单中选择“删除表”命令,弹出“确认删除”对话框,在该对话框中单击“删除”按钮,该表被删除。

2) 定义订单项目表 orderitems

(1) 在 booksale 数据库中右击“表”节点,在弹出的快捷菜单中选择“新建表”命令,打开新建表工作页。

(2) 在新打开的工作页中首先给出表中各个列的名称、数据类型以及是否允许空值。该表的主键由订单编号 orderid 和图书编号 bookid 共同组成,所以分别选中两列后单击“主键”按钮设置主键,如图 5-6 所示。

栏位 | 索引 | 外键 | 触发器 | 选项 | 注释 | SQL 预览

名	类型	长度	小数点	不是 null	
orderid	int	11	0	☑	1
bookid	int	11	0	☑	2
quantity	int	11	0	☑	
price	decimal	6	2	☐	

图 5-6 创建订单项目表 orderitems

(3) 单击“保存”按钮,在弹出的“表名”对话框中将表命名为 orderitems,单击“确定”按钮保存该表。

(4) 关闭创建订单项目表 orderitems 的工作页。

2. 数据操作

(1) 展开 booksale 数据库中的“表”节点,在 books 表上右击选择“打开表”命令,打开编辑表数据工作页。

(2) 表中第一列 bookid 是标识列,不需要输入数据,所以将光标置于 title 列,输入“Web 前端开发基础入门”,按 Tab 键或单击鼠标移动到下一个列,在 isbn 列中输入“978-7-3025-7626-6”,在 author 列中输入“张颖”,在 unitprice 列中输入“65.00”,在 ctgcode 列中输入“computer”,按 Tab 键转换到第二行时,第一行记录的输入完成,此时系统自动为第一本书添加书号“1”。

(3) 继续录入其他记录,直到完成所有记录的添加。

(4) 关闭编辑图书表 books 中数据的工作页,在弹出的“确认”对话框中单击“保存”按钮,保存对数据的编辑。

提示:在输入数据的过程中,如果想要删除当前行中某个列的数据,可以单击 Esc 键,该列中显示 NULL,表示该列的数据已经被删除。如果在显示 NULL 的列中再次单击 Esc 键则会取消整条记录的输入。

若打开的数据表为空(没有记录)可直接进行记录输入;若打开的数据表已经存在部分

记录，此时需要单击编辑表数据工作页左下角的“+”(新建记录)来人工添加一条记录后，完成相应记录的输入。

更新表中的数据非常简单，只需要直接修改列中的数据就可以了。如果要删除一条记录，首先单击该记录前面的按钮以选中整条记录，在该行上右击，在弹出的快捷菜单中选择“删除记录”命令(或单击要删除的记录，再单击编辑数据表数据工作页左下角的“-”删除记录)，在弹出的警告“确认删除”对话框中单击“删除一条记录”按钮将删除相应的记录。

5.5 实践练习

1. 定义表

注意：运行脚本文件 Ex-Chapter5-Database.sql 创建数据库 teachingsys，并在该数据库下完成练习。

(1) 创建系部表 departments，该表的结构见附录 B 中表 B-7(此处不需要定义唯一性约束)。

(2) 修改系部表 departments，增加一列 dptlocation 表示系部地址，数据类型为 VARCHAR(50)，允许空值。

(3) 创建班级表 classes，该表的结构见附录 B 中表 B-8(此处不需要定义唯一性约束和外键约束)。

(4) 修改班级表 classes，将班级名称列 classname 的数据类型修改为 VARCHAR(20)，允许空值。

(5) 创建学生表 students，该表的结构见附录 B 中表 B-9(此处不需要定义默认值约束和外键约束)。

(6) 修改学生表 students，将学生出生日期列 dob 的名称修改为 birthday，并且移动到性别列 gender 之后。

(7) 创建教师表 teachers，该表的结构见附录 B 中表 B-10(此处不需要定义外键约束)。

(8) 修改教师表 teachers，删除表示教师职称的列 protitle。

(9) 创建课程表 courses，该表的结构见附录 B 中表 B-11(此处不需要定义唯一性约束)。

(10) 创建选修表 studying，该表的结构见附录 B 中表 B-12(此处不需要定义外键约束)。

2. 操作表中数据

注意：运行脚本文件 Ex-Chapter5-Table.sql 重新创建数据库 teachingsys 及相关数据表，并在该数据库下完成练习。

(1) 使用一个 INSERT 语句向班级表 classes 中插入 4 条记录，数据见附录 B 中表 B-2。

(2) 向学生表 students 中插入最后 1 条记录，数据见附录 B 中表 B-3。

(3) 将教师表 teachers 中教师编号 tchid 为 2 的教师的职称 protitle 修改为“教授”。

(4) 将课程表 courses 中编号 crsid 为 1 的课程的名称 crsname 修改为“mysql 数据库实现与维护”，并将学分 credit 修改为 4。

(5) 将选修表 studying 中所有没有成绩的选修成绩 mark 修改为 0 分。

(6) 将教师表 teachers 中教师编号 tchid 为 4 的教师记录删除。

(7) 删除选修表 studying 中的所有记录。