第5章

数据库的创建与管理

如果要使用数据库管理系统对数据进行管理，首先需要创建数据库。创建数据库主要包括确定数据库的名称、为数据库设置相关属性信息等。然后以数据库为起点，根据应用需求创建数据表的结构并录入数据，之后才能对表中数据进行查询、更新、统计等操作。在数据库系统中，数据表是数据存储、管理和查询过程中最重要而又最基本的操作对象，是查询操作的数据基础，是视图、索引、存储过程的处理对象。

5.1 数据库的创建

数据库的创建

5.1.1 创建数据库

创建数据库就是确定数据库名称、使用的字符集、校验规则以及相关属性，操作系统为该数据库分配一定的存储空间，用于存储数据库的对象，包括数据表、索引、视图及存储过程等。在 MySQL 中，创建数据库是通过 SQL 语句 CREATE DATABASE(或 CREATE SCHEMA 语句)来实现。语法格式如下：

```
CREATE DATABASE [IF NOT EXISTS] 数据库名称
[CHARSET 字符集 ]
[COLLATE 校验规则];
```

说明如下。

(1) 数据库名称：在 MySQL 系统中，数据库及其包含的对象是以目录或文件的方式进行管理的，因此，数据库名称必须符合操作系统的文件命名规则。

(2) IF NOT EXISTS：可选项，如果使用该选项，则会在创建数据库之前，首先判断该数据库是否存在，若存在则不再创建，也不会报错；如果不存在则会创建该数据库。如果没有使用该选项，则创建数据库时若数据库已经存在，系统就会报错。

(3) CHARSET：可选项，用来设置数据库采用的字符集。字符集是一套字符内容与编码方式的集合。MySQL 使用字符集和校验规则来处理字符数据。省略表示采用默认字符集 utf8mb4。

(4) COLLATE：可选项，用来设置数据库使用的校验规则。校验规则(COLLATION)是指在特定字符集中用于比较字符大小的一套规则，是字符串的排序规则。省略表示采用默认值 utf8mb4_0900_ai_ci。

MySQL 支持多种字符集，每种字符集对应多种校验规则，且都有一种默认的校验规则。每种校验规则只对应一种字符集，与其他的字符集无关。例如，字符集 utf8mb4 的默认校验规则是 utf8mb4_0900_ai_ci，字符集 gbk 的默认校验规则是 gbk_chinese_ci。因此，在设置字符集和校验规则时要注意匹配使用，不能交叉使用。

说明：本书中创建数据库和数据表时，字符集和校验规则都采用系统默认值。

【例 5.1】 创建网络购物系统数据库，数据库名称为 online_sales_system，字符集和校验规则采用默认值。

SQL 语句如下：

```
CREATE DATABASE IF NOT EXISTS online_sales_system;
```

运行结果如图 5.1 所示。

```
mysql> CREATE DATABASE IF NOT EXISTS online_sales_system;
Query OK, 1 row affected (0.00 sec)
```

图 5.1 创建数据库

运行结果中“Query OK，1 row affected(0.00 sec)”表示语句成功运行，有一行受到影响，用时 0.00s。SQL 语句运行速度很快，由于显示精度的原因，不到 0.01s，显示为 0.00s。读者的运行结果中的运行时间可能和上图有所不同，那是因为运行时间的长短与使用的计算机的配置有关。图 5.1 显示结果表明数据库 online_sales_system 已成功被创建。

5.1.2 管理数据库

1. 显示数据库

使用 SHOW DATABASES 语句可以显示当前可用的所有数据库，包括 MySQL 系统提供的数据库和用户自己创建的数据库。语法格式如下：

```
SHOW DATABASES;
```

【例 5.2】 使用 SHOW DATABASES 语句显示当前可用的数据库。

SQL 语句如下：

```
SHOW DATABASES;
```

运行结果如图 5.2 所示。

由运行结果可以看出，SHOW DATABASES 语句正常运行，显示当前目录中共有

5 个不同的数据库，分别是 information_schema、mysql、online_sales_system、performance_schema 和 sys，其中只有 online_sales_system 是用户定义的数据库，其他 4 个是系统自带的数据库。

```
mysql> SHOW DATABASES;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| online_sales_system |
| performance_schema |
| sys                |
+--------------------+
5 rows in set (0.25 sec)
```

图 5.2　显示当前可用的数据库

（1）information_schema 数据库：存储了数据库的元数据（Database Metadata），元数据是关于 MySQL 服务器的信息，例如数据库或表的名称、列的数据类型或访问权限等，有时也称为数据字典（Data Dictionary ）或系统目录（System Catalog）。

（2）mysql 数据库：提供了 MySQL 服务器运行时所需信息的表，包括用于存储数据库对象元数据（Database Object Metadata）的数据字典表，以及诸如日志系统表、时区系统表等一系列系统表。

（3）performance_schema 数据库：用于监控服务器的运行性能，并提供执行时的有用信息的访问。

（4）sys 数据库：此数据库主要是利用 performance_schema 数据库提供的信息，生成一系列的视图、存储过程和存储函数，帮助数据库管理人员和开发者了解系统的性能。

注意：读者的数据库列表可能和这里的不一样，因为显示的数据库与读者自己定义的数据库有关。

2. 打开数据库

连接到 MySQL 时，使用 SHOW DATABASES 语句查看到的都是不同数据库的名称，即不同的数据库是并列存在的。此时没有指定特定数据库。用户使用 CREATE DATABASE 语句创建数据库时，不能将其设置为当前数据库。如果要指定某个数据库为当前数据库，需要使用 USE 命令打开该数据库，使之成为当前数据库。打开数据库的语法格式如下：

```
USE 数据库名称;
```

【例 5.3】　打开网络购物数据库 online_sales_system。

SQL 语句如下：

```
USE online_sales_system;
```

运行结果如图 5.3 所示。

```
mysql> USE online_sales_system;
Database changed
```

图 5.3　打开数据库

执行结果“Database changed”说明 USE 语句执行成功，当前数据库已经切换为 online_sales_system 数据库。之后的命令如果不做特别说明（使用数据库名来限定表名），都是针对该数据库中的对象进行操作。如果需要在不同数据库之间切换，也要使用 USE 语句打开对应的数据库。因此，打开数据库也称为切换数据库。

3. 查看数据库信息

数据库信息主要包括数据库中包含的数据表、视图、存储函数、存储过程等对象,以及数据库使用的字符集和校验规则等,在 MySQL 中提供了相应的语句查看这些信息。

方法一:分别查看数据库中包含的表、使用的字符集和校验规则。

使用 SHOW TABLES 语句查看数据库中包含的表、视图等对象,语法格式如下:

```
SHOW TABLES;
```

使用 SHOW VARIABLES LIKE 语句查看当前数据库使用的字符集,语法格式如下:

```
SHOW VARIABLES LIKE'character_set_database';
```

使用 SHOW VARIABLES LIKE 语句查看当前数据库使用的校验规则,语法格式如下:

```
SHOW VARIABLES LIKE'collation_database';
```

其中,VARIABLES 是指系统变量,在 MySQL 中提供了很多系统变量,用于存储定义数据库时用到的元数据以及系统环境信息。character_set_database 和 collation_database 都属于系统变量。而 LIKE 是一个比较运算符,使用规则为 LIKE <表达式>,用于检索和<表达式>相匹配的变量,其中<表达式>中可以使用通配符(%可以匹配多个字符,"_"可以匹配一个字符)。

【例 5.4】 查看网络购物数据库 online_sales_system 中包含的数据表等对象。

SQL 语句如下:

```
SHOW TABLES;
```

```
mysql> SHOW TABLES;
Empty set (0.09 sec)
```

图 5.4 数据库 online_sales_system 中的数据表

运行结果如图 5.4 所示。

运行结果 Empty set(0.09 sec)的含义是 SHOW TABLES 语句运行成功,但在当前数据库中没有数据库对象,因此显示为 Empty set,(0.09 sec)表示运行该语句用时 0.09s。

【例 5.5】 查看网络购物数据库 online_sales_system 使用的字符集。

SQL 语句如下

```
SHOW VARIABLES LIKE 'character_set_database';
```

运行结果如图 5.5 所示。

由运行结果可以看出,当前数据库 online_sales_system 使用的字符集为 utf8mb4,与

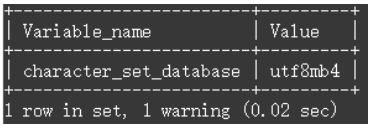

图 5.5　显示数据库 online_sales_system 使用的字符集

创建该数据库时使用的默认字符集一致。

【例 5.6】　查看网络购物数据库 online_sales_system 使用的校验规则。

SQL 语句如下：

```
SHOW VARIABLES LIKE'collation_database';
```

运行结果如图 5.6 所示。

图 5.6　显示数据库 online_sales_system 使用的校验规则

由运行结果可以看出，当前数据库 online_sales_system 使用的校验规则为 utf8mb4_0900_ai_ci，与创建该数据库时使用的默认校验规则一致。

方法二：使用 SHOW CREATE DATABASE 语句查看数据库定义的详细信息，包括创建该数据库的 SQL 语句、数据库使用的字符集和校验规则等。语法格式如下：

```
SHOW CREATE DATABASE <数据库名称>;
```

【例 5.7】　使用 SHOW CREATE DATABASE 语句查看 online_sales_system 数据库定义的详细信息。

SQL 语句如下：

```
SHOW CREATE DATABASE online_sales_system;
```

运行结果如图 5.7 所示。

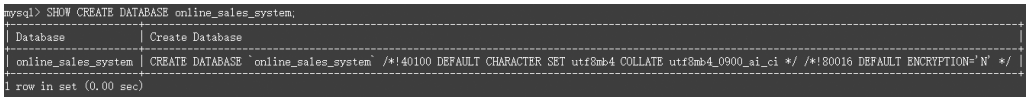

图 5.7　使用 SHOW CREATE DATABASE 显示 online_sales_system 数据库的详细信息

此运行结果中除了显示创建数据库的语句、数据库的字符集和校验规则外，还显示了数据库是否加密的信息，“DEFAULT ENCRYPTION＝'N'”显示加密的值为'N'，表示没加密。如果加密的值是'Y'，表示已加密。

4. 删除数据库

删除数据库是指删除数据库以及数据库中所有的表、索引、视图等对象,并清除操作系统分配给该数据库的存储空间。删除数据库后,所有数据库存储的数据都将被删除,而且删除操作不可恢复。因此,删除数据库时需要特别谨慎,应遵循非必要不删除的原则。即使遇到必须删除的情况,也建议先将数据库进行备份,之后再进行删除。删除数据库的语句的语法规则如下:

```
DROP DATABASE [IF EXISTS]数据库名;
```

【例 5.8】 使用 DROP DATABASE 语句删除刚创建的网络购物数据库。

SQL 语句如下:

```
DROP DATABASE online_sales_system;
```

运行结果如图 5.8 所示。

```
mysql> DROP DATABASE online_sales_system;
Query OK, 0 rows affected (0.00 sec)
```

图 5.8 删除数据库 online_sales_system

【例 5.9】 使用 DROP DATABASE 语句删除数据库时,直接删除和使用 IF EXISTS 可选项再删除一次数据库 online_sales_system。运行结果如图 5.9 所示。

```
mysql> DROP DATABASE online_sales_system;
ERROR 1008 (HY000): Can't drop database 'online_sales_system'; database doesn't exist
```

(a) 直接删除数据库

```
mysql> DROP DATABASE IF EXISTS online_sales_system;
Query OK, 0 rows affected, 1 warning (0.00 sec)
```

(b) 使用IF EXISTS删除数据库

图 5.9 删除数据库

由运行结果可以看出可选项 IF EXISTS 的作用:在执行删除操作之前首先判断数据库是否存在,只有数据库存在才执行删除操作,否则不执行删除操作,也不给出错误信息。而不使用可选项 IF EXISTS,则不做判断直接删除,若被删的数据库不存在,系统就会给出错误提示信息"database doesn't exist"。在实际应用中,建议大家使用 IF EXISTS 选项。


认识数据表

5.2 认识数据表

在 MySQL 中,不同的数据库是以目录的形式进行管理的,使用 USE 语句可以设置当前数据库,也相当于设置一个当前目录。在该目录中包含该

数据库的所有对象，包括数据表、视图、索引、存储过程或存储函数等对象。其中，数据表是存储原始数据的主要对象。它由行和列组成，每列为一个字段，描述实体集的一个属性；每行为一条记录，描述一个具体的实体。在一个数据库中可以包含多个数据表，每个数据表保存一个实体集或一个联系的相关数据。

在 MySQL 中为数据库、数据表、视图、索引、存储过程、存储函数等对象命名的有效字符序列统称为标识符。标识符实际上就是一个名称，但标识符必须符合一定规则或约定，在 MySQL 中标识符的命名规则如下。

(1) 标识符可以由当前字符集中的任何字母、中文、数字字符组成，另外，还可以包括下画线(_)和美元符号($)。但一般不建议使用中文作为标识符，因为中文字符在计算机中存储时涉及字符集的选择，不同的字符集对中文字符的编码不同，如果字符集不匹配会出现乱码。如果使用中文字符作为标识符，那么代码的可移植性就会很差。

(2) 标识符最长为 64 个字符。但标识符的长度受限于所用操作系统限定的长度。

(3) 标识符应尽可能做到“见名知意”。换句话说，通过数据表的名称就可以知道数据表表示的实体集，通过字段名称就可以知道字段表示的属性值的含义，从而增强可读性。通常可以选择能表示数据含义的英文单词或缩写、汉语拼音等作为标识符。如用 customers 作为客户表的名称、用 price 作为销售价格的字段名称。

(4) 用户自己定义的标识符可以直接使用，如果标识符名与系统保留字(关键字)重名，应该用反引号括起来(注意，不是单引号)。但建议在给对象命名时，避免和系统的保留字重名。系统提供的语句命令和语句中的关键字都属于保留字，例如 CREATE、DROP、USE、SELECT、FROM 等。

(5) 操作系统中的文件系统是否大小写敏感，会影响如何命名和引用数据库对象。如果文件系统对大小写敏感(如 UNIX)，名字 tbl_items、TBL_items 和 tbl_ITEMS 是 3 个不同的标识符。如果文件系统对大小写不敏感(如 Windows)，那么这 3 个名字指的是一个标识符，也就是对应一个对象。

注意：各个 DBMS 的约定不完全相同，使用前需要查看相关的系统资料。在 Windows 环境中的 MySQL 系统默认对标识符不区分大小写，在 Linux 环境中默认区分大小写。

1. 表的名称

完整的数据表名称由数据库名和表名两部分构成，其格式如下：

```
数据库名.表名
```

在当前数据库中操作数据表等对象时，数据表名之前的数据库名可以省略，只有在处理属于两个不同数据库中的数据表时，才需要用完整的数据表名。

表名和数据库名的命名需要遵循 MySQL 的标识符命名规则。例如，网络购物数据库的名称为 online_sales_system，包含的 4 个表的名称分别客户表 customers、商品表 items、订单表 orders、订单明细表 order_details，它们均符合标识符的命名规则。

2. 数据表

在 MySQL 中,数据表是一张满足关系模型的二维表,二维表的第一行是各列标题,每列称为一个字段,所有的字段构成一张数据表的表结构,其余每行代表一条记录,记录中每列的值代表该记录对应字段的值。所有的记录构成表的内容。数据表实质上就是行列的集合。

【例 5.10】 显示客户表 customers 的数据,了解表的构成。

SQL 语句如下:

```
SELECT * FROM customers;
```

运行结果如图 5.10 所示。

```
mysql> SELECT * FROM customers;
+-------------+--------------+--------+-----------------+-------------+
| customer_id | name         | gender | registration_date | phone       |
+-------------+--------------+--------+-----------------+-------------+
| 101         | 薛为民       | 男     | 2012-01-09      | 16800001111 |
| 102         | 刘丽梅       | 女     | 2016-01-09      | 16811112222 |
| 103         | Grace_Brown  | 女     | 2016-01-09      | 16822225555 |
| 104         | 赵文博       | 男     | 2017-12-31      | 16811112222 |
| 105         | Adrian_Smith | 男     | 2017-11-10      | 16866667777 |
| 106         | 孙丽娜       | 女     | 2017-11-10      | 16877778888 |
| 107         | 林琳         | 女     | 2020-05-17      | 16888889999 |
+-------------+--------------+--------+-----------------+-------------+
7 rows in set (0.00 sec)
```

图 5.10 显示客户表 customers 的数据

由运行结果可以看出,客户表 customers 包含 5 个字段,分别是 customer_id、name、gender、registration_date 和 phone;customers 包含 7 条记录。第一条记录的含义是 customer_id 的值为字符串'101',name 的值为'薛为民',gender 的值为'男',registration_date 的值为'2012-01-09',phone 的值为'16800001111';第二条记录表示的数据和第一条记录不同,尤其是 customer_id 一定不同,从这里也可以发现,对于不同的记录,字段值是不完全相同的。

3. 表结构

表结构描述的是表的框架,表结构决定数据表拥有哪些字段以及这些字段的特性。字段特性主要包括字段的名称、数据类型、长度、精度、小数位数、是否允许空值(NULL)、是否需要设置默认值、是否为主键、是否为外键、是否有索引等。其中,每个字段的数据类型是必须定义的,数据类型决定该字段的取值范围和可以参与的运算。例如,客户表中的联系电话 phone 字段虽然都是由数字组成,但联系电话不会参与数学运算,有可能会参与字符串连接运算,因此定义为字符类型,又根据目前电话号码最长是 11 位,因此确定 phone 字段最多只能取 11 位字符;注册日期 registration_date 字段只能取合法的日期值,即必须符合现实生活中的日期,月份不能小于 1 或大于 12,日期不能小于 1 或大于 31 等。

【例 5.11】 使用 DESC 语句查看客户表 customers 的表结构。

SQL 语句如下:

```
DESC customers;
```

运行结果如图 5.11 所示。

```
+------------------+-----------------+------+-----+---------+-------+
| Field            | Type            | Null | Key | Default | Extra |
+------------------+-----------------+------+-----+---------+-------+
| customer_id      | char(3)         | NO   | PRI | NULL    |       |
| name             | varchar(20)     | NO   |     | NULL    |       |
| gender           | enum('男','女') | YES  |     | 男      |       |
| registration_date| date            | YES  |     | NULL    |       |
| phone            | char(11)        | YES  |     | NULL    |       |
+------------------+-----------------+------+-----+---------+-------+
5 rows in set (0.14 sec)
```

图 5.11　使用 DESC 查看 customers 的表结构

由运行结果可以看出，客户表 customers 的表结构由 customer_id、name、gender、registration_date 和 phone 一共 5 个字段构成，其中字段 customer_id 的数据类型为 char(3)，且为 customers 表的主键(Key 列的值为 PRI 表示该字段为主键)；字段 name 的数据类型为 varchar(20)；字段 gender 为枚举类型，即只能从“男”和“女”两个值中选择一个，且默认值为“男”；字段 registration_date 的数据类型为 date；字段 phone 的数据类型为 char(11)。

4. 字段名

每个数据表可以拥有多个字段，每个字段分别用来存储不同类型、不同性质的数据。字段名除了必须符合 MySQL 的标识符命名规则之外，还要满足以下要求。

(1) 字段名可由中文、英文字母、数字、下画线(_)、# 符号及 $ 符号组合而成。在实际应用中，不建议使用中文作为字段名。

(2) 同一个数据表中，不能出现重名字段。但不同数据表中的字段名可以重名。

(3) 字段名应尽可能做到“见名知意”。通过字段名称就可以知道字段表示的属性值的含义，从而增强可读性。例如 name 表示姓名、gender 表示性别。

(4) 字段名不能使用 MySQL 语言中的关键字。如 DROP、ALTER、INSERT、CREATE、ONLINE、FROM 等。

5. 表间关系

在关系数据库中，一个数据库系统一般包含多个数据表，数据表之间也会存在关系。表间关系是通过外键(外码)来实现的，也体现了关系之间的参照完整性。如果表 A 外键字段的数据取值要么是 NULL，要么是来自于表 B 主键字段的值，那么将表 A 称为表 B 的从表(子表)，表 B 称为表 A 的主表(父表)。也就是说，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表就是主表(父表)，外键所在的表就是从表(子表)。网络购物数据库中 4 个表之间的关系如图 5.12 所示。

从图 5.12 可以看出，客户表 customers 的主键为客户编号 customer_id，商品表 items 的主键为商品编号 item_id，订单表 orders 的主键为订单编号 order_id。订单表的外键为客户编号 customer_id，与客户表构成从-主表关系，订单明细表 order_details 的主键为订

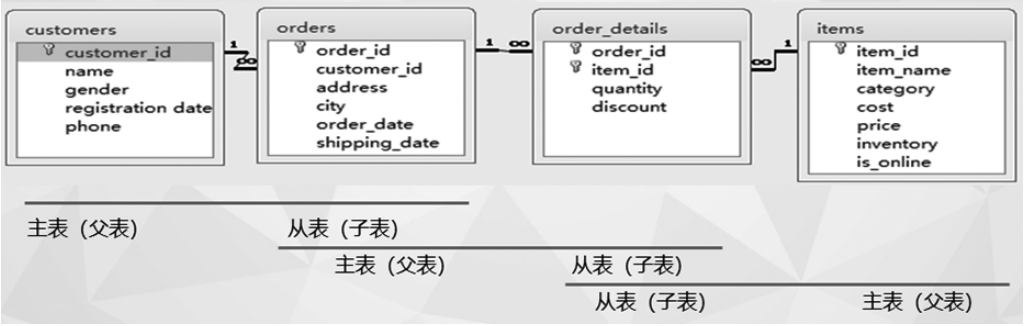

图 5.12 网络购物数据库中的表间关系

单编号 order_id 和商品编号 item_id,一个外键为订单编号 order_id,与订单表构成从-主关系;另一个外键为商品编号 item_id,与商品表 items 构成从-主表关系。

5.3 数据类型

在我们认知的世界中,存在着五花八门的数据,如果要将数据存储到计算机中,就需要按数据类型将这些数据进行分类,不同类型的数据在计算机中占用的字节数、编码方式、取值范围以及能参与的运算各不相同。在创建数据表的结构时,必须首先确定每个字段的数据类型。字段的数据类型是数据完整性的一部分,它限制了字段的取值范围、存储方式和使用方法。MySQL 提供了多种数据类型。在创建表时根据实际需求(字段值的范围、大小、精度、参与的运算等)为每个字段选择合适的数据类型,不但能节省整个数据库占用的存储空间,而且能提高数据库的运行效率。

5.3.1 数值类型

MySQL 支持所有的 ANSI/ISO SQL 92 数值类型(ANSI, American National Standards Institute,美国国家标准局),数值分为整数和小数。其中整数用整数类型表示,小数用浮点数类型或定点数类型表示。

1. 整数类型

整数就是由正负符号和 0～9 构成的不带小数点的数据,正号可以省略,例如 12、56、－128 等都属于整型数据。在 MySQL 中整数类型包括 TINYINT、SMALLINT、MEDIUMINT、INT 和 BIGINT 等。不同类型的整数占用的存储空间不同,取值范围也不同,如表 5.1 所示。

从表 5.1 中可以看出,占用字节数越多的类型存储的数值范围越大。在 MySQL 8.0.17 之前的版本中可以指定整型数据的显示宽度,使用 INT(*M*)进行设置,*M* 表示最大显示宽度。显示宽度是可选项,如果没有指定显示宽度,则按系统默认的宽度显示。例如 INT(5)表示最大有效显示宽度为 5。在 MySQL 8.0.17 之后的版本不推荐设置显示宽度,并将在未来的版本中移除此用法。