CHAPTER 第3章 MySQL 数据库和表

可以将数据库看作一个存储数据对象的容器，这些对象包括表、视图、触发器、存储过程等。其中，表是最基本的数据对象，是存放数据的实体。实际应用中，必须首先创建数据库，然后才能建立表及其他数据对象。

3.1 MySQL 数据库

3.1.1 创建数据库

使用 CREATE DATABASE 或 CREATE SCHEMA 命令可以创建数据库，其语法格式为：

```
CREATE {DATABASE|SCHEMA} [IF NOT EXISTS] 数据库名
[ 选项  ...]
```

选项：

```
[DEFAULT] CHARACTER SET 字符集
|[DEFAULT] COLLATE 校对规则名
```

说明：

- 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 DATABASE 操作。用此选项可以避免出现数据库已经存在而再新建的错误。
- DEFAULT——指定默认值。
- 字符集——指定数据库采用的字符集。
- COLLATE——指定字符集的校对规则。字符集和校对规则的概念参考附录 D。

【例 3.1】 创建学生成绩数据库 xscj。

```
mysql>create database xscj
```

如果已经创建了数据库，如 mytest，重复创建时系统会提示数据库已经存在，不能再创建。系统显示错误信息，如图 3.1 所示。使用 IF NOT EXISTS 选项从句可不显示错误信息。

```
mysql> create database mytest;
ERROR 1007 (HY000): Can't create database 'mytest'; database exists
mysql> create database if not exists mytest;
Query OK, 1 row affected, 1 warning (0.00 sec)
```

图 3.1 错误信息提示

创建了数据库之后使用 USE 命令可指定当前数据库,其语法格式为:

```
USE  数据库名;
```

例如,指定当前数据库为学生成绩数据库(xscj):

```
mysql>use xscj
```

说明:这个语句也可以用来从一个数据库"跳转"到另一个数据库,在用 CREATE DATABASE 语句创建了数据库之后,该数据库不会自动成为当前数据库,需要用这条 USE 语句来指定。

注意:在 MySQL 中,每一条 SQL 语句都以分号";"作为结束标志。

3.1.2 修改数据库

数据库创建后,如果需要修改数据库的参数,可以使用 ALTER DATABASE 命令,其语法格式为:

```
ALTER {DATABASE |SCHEMA} [数据库名]
    选项  ...
```

选项:

```
[DEFAULT] CHARACTER SET 字符集名
    | [DEFAULT] COLLATE 校对规则名
```

说明:ALTER DATABASE 用于更改数据库的全局特性,这些特性存储在数据库目录中的 db.opt 文件中。用户必须有对数据库进行修改的权限,才可以使用 ALTER DATABASE。修改数据库的选项与创建数据库相同,功能不再重复说明。如果语句中数据库名称忽略,则修改当前(默认)数据库。

【例 3.2】 修改学生成绩数据库(xscj)默认字符集和校对规则。

输入命令如图 3.2 所示。

```
mysql> alter database xscj
    -> default character set gb2312
    -> default collate gb2312_chinese_ci;
Query OK, 1 row affected (0.00 sec)
```

图 3.2 执行命令

3.1.3 删除数据库

已经创建的数据库需要删除,可使用 DROP DATABASE 命令,其语法格式为:

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

还可以使用 IF EXISTS 子句,避免删除不存在的数据库时出现 MySQL 错误信息。

注意:这个命令必须小心使用,因为它将删除指定的整个数据库,该数据库的所有表(包

括其中的数据)也将永久删除。

3.2　MySQL 表

在数据库创建后,就应该创建表,因为表是数据库存放数据的对象实体。没有表,数据库中其他数据对象就都没有意义。要查看数据库中有哪些表,可以使用 SHOW TABLES 命令。

3.2.1　创建表

1. 全新创建

从头创建一个全新的表,可使用 CREATE TABLE 命令,其语法格式为:

```
CREATE [TEMPORARY] TABLE [IF NOT EXISTS] 表名
    [([列定义]  ...|[表索引定义])]
    [表选项] [select 语句];
```

说明:

- TEMPORARY——表示用 CREATE 命令新建的表为临时表。不加该关键字创建的表称为持久表。在数据库中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。此后可能要重复地使用这个结果,但该结果又不需永久保存,这时可使用临时表。用户可像操作持久表一样操作临时表。只不过临时表的生命周期较短,而且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。
- IF NOT EXISTS——在创建表前加上一个判断,只有该表目前尚不存在时才执行 CREATE TABLE 操作。用此选项可避免出现表已经存在无法再新建的错误。
- 列定义——包括列名、数据类型,可能还有一个空值声明和一个完整性约束。
- 表索引项定义——主要定义表的索引、主键、外键等,具体定义参见第 5 章。
- select 语句——用于在一个已有表的基础上创建表。

(1)“列定义”格式为:

```
列名  type  [NOT NULL|NULL] [DEFAULT 默认值]
    [AUTO_INCREMENT] [UNIQUE [KEY]|[PRIMARY] KEY]
    [COMMENT 'string'] [参照定义]
```

其中:

- 列名——必须符合标识符规则,长度不能超过 64 个字符,而且在表中要唯一。如果为 MySQL 保留字,必须用单引号括起来。
- type——列的数据类型,有的数据类型需要指明长度 n,并用括号括起来,MySQL 支持的数据类型见附录 C。
- AUTO_INCREMENT——设置自增属性,只有整型列才能设置此属性。当插入 NULL 值或 0 到一个 AUTO_INCREMENT 列中时,列被设置为 value+1,这里 value 是此前表中该列的最大值。AUTO_INCREMENT 顺序从 1 开始。每个表只

能有一个 AUTO_INCREMENT 列,并且它必须被索引。

- NOT NULL|NULL——指定该列是否允许为空。如果不指定,则默认为 NULL。
- DEFAULT 默认值——为列指定默认值,默认值必须为一个常数。其中,BLOB 和 TEXT 列不能被赋予默认值。如果没有为列指定默认值,MySQL 自动地分配一个。如果列可以取 NULL 值,默认值就是 NULL。如果列被声明为 NOT NULL,默认值取决于列类型:
  - 对于没有声明 AUTO_INCREMENT 属性的数字类型,默认值是 0。对于一个 AUTO_INCREMENT 列,默认值是在顺序中的下一个值。
  - 对于除 TIMESTAMP 以外的日期和时间类型,默认值是该类型适当的“零”值。对于表中第一个 TIMESTAMP 列,默认值是当前的日期和时间。
  - 对于除 ENUM 的字符串类型,默认值是空字符串。对于 ENUM,默认值是第一个枚举值。
- UNIQUE KEY|PRIMARY KEY:PRIMARY KEY 和 UNIQUE KEY 都表示字段中的值是唯一的。PRIMARY KEY 表示设置为主键,一个表只能定义一个主键,主键一定要为 NOT NULL。
- COMMENT 'string':对于列的描述,string 是描述的内容。
- 参照定义:指定参照的表和列,具体定义将在 5.3 节中介绍。

type 定义如下:

```
TINYINT[(length)][UNSIGNED][ZEROFILL]
|SMALLINT[(length)][UNSIGNED][ZEROFILL]
|MEDIUMINT[(length)][UNSIGNED][ZEROFILL]
|INT[(length)][UNSIGNED][ZEROFILL]
|INTEGER[(length)][UNSIGNED][ZEROFILL]
|BIGINT[(length)][UNSIGNED][ZEROFILL]
|REAL[(length,decimals)][UNSIGNED][ZEROFILL]
|DOUBLE[(length,decimals)][UNSIGNED][ZEROFILL]
|FLOAT[(length,decimals)][UNSIGNED][ZEROFILL]
|DECIMAL(length,decimals) [UNSIGNED][ZEROFILL]
|NUMERIC(length,decimals) [UNSIGNED][ZEROFILL]
|BIT[M]
|DATE
|TIME
|TIMESTAMP
|DATETIME
|CHAR(length) [BINARY|ASCII|UNICODE]
|VARCHAR(length) [BINARY]
|TINYBLOB
|BLOB
|MEDIUMBLOB
|LONGBLOB
```

```
|TINYTEXT [BINARY]
|TEXT [BINARY]
|MEDIUMTEXT [BINARY]
|LONGTEXT [BINARY]
|ENUM(value1,value2,value3,...)
|SET(value1,value2,value3,...)
|spatial_type
```

说明：在字符数据类型和数值数据类型之后，MySQL 允许指定一个数据类型选项来改变数据类型的属性和功能。

对于字符数据类型，MySQL 支持两种数据类型选项：CHARACTER SET 和 COLLATE。如果要区分字符的大小写，可以在字符类型后面加上 BINGARY。

对于除 BIT 以外的数值数据类型，MySQL 允许添加一个或多个数据类型选项。UNSIGNED：不允许负值。ZEROFILL：当插入的值长度小于字段设定的长度时，剩余部分用 0 填补。

spatial_type 是空间类型数据，本书不讨论。

(2)“表选项”定义如下：

```
{ENGINE|TYPE}=engine_name                                  /*存储引擎*/
|AUTO_INCREMENT=value                                      /*初始值*/
|AVG_ROW_LENGTH=value                                      /*表的平均行长度*/
|[DEFAULT] CHARACTER SET 字符集名 [COLLATE 校对规则名]      /*默认字符集和校对*/
|CHECKSUM={0|1}                                            /*设置 1 表示求校验和*/
|COMMENT='string'                                          /*注释*/
|CONNECTION='connect_string'                               /*连接字符串*/
|MAX_ROWS=value                                            /*行的最大数*/
|MIN_ROWS=value                                            /*列的最小数*/
|PACK_KEYS={0|1|DEFAULT}
|PASSWORD='string'                                         /*对.frm 文件加密*/
|DELAY_KEY_WRITE={0|1}                                     /*对关键字的更新*/
|ROW_FORMAT={DEFAULT|DYNAMIC|FIXED|COMPRESSED|REDUNDANT|COMPACT}
                                                           /*定义各行应如何储存*/
|UNION=(表名[,表名]...)                                     /*表示哪个表应该合并*/
|INSERT_METHOD={NO|FIRST|LAST}                             /*是否执行 INSERT 语句*/
|DATA DIRECTORY='absolute path to directory'               /*数据文件的路径*/
|INDEX DIRECTORY='absolute path to directory'               /*索引的路径*/
```

说明：表中大多数的选项涉及的是表数据如何存储及存储在何处。多数情况下，不必指定表选项。ENGINE 选项是定义表的存储引擎。

【例 3.3】　在学生成绩数据库(xscj)中也创建一个学生情况表，表名为 xs。

输入以下命令：

```
USE xscj
CREATE TABLE xs
(
    学号        char(6)        not null  primary key,
    姓名        char(8)        not null,
    专业名      char(10)       null,
    性别        tinyint(1)     not null  default 1,
    出生日期    date           not null,
    总学分      tinyint(1)     null,
    照片        blob           null,
    备注        text           null
) engine=innodb;
```

在上面的例子里,每个字段都包含附加约束或修饰符,这些可用来增加对所输入数据的约束。primary key 表示将“学号”字段定义为主键。default 1 表示“性别”的默认值为 1。engine=innodb 表示采用的存储引擎是 InnoDB,InnoDB 是 MySQL 在 Windows 平台默认的存储引擎,所以 engine=innodb 也可以省略。

然后,用 show tables 命令显示 xscj 数据库中产生了学生(xs)表,用 describe xs 命令可以显示 xs 表的结构,如图 3.3 所示。

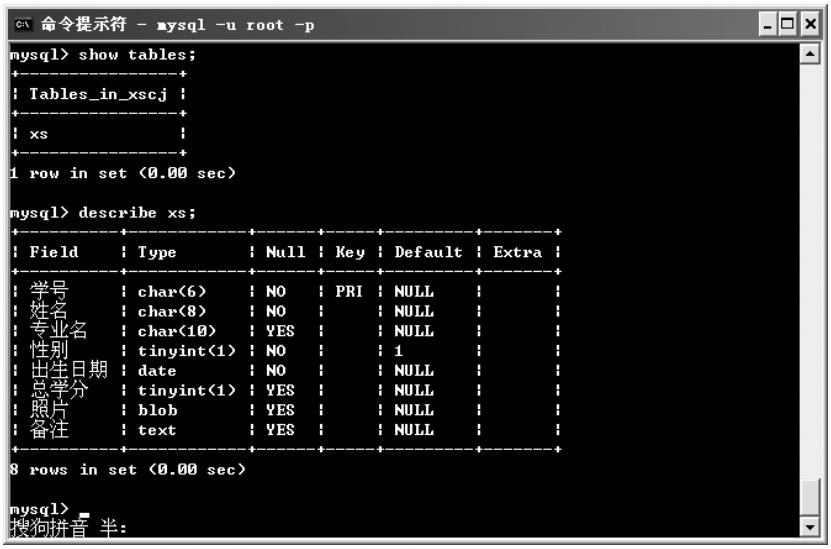

图 3.3 学生(xs)表结构

2. 复制现成的表

除了全新创建外,用户也可直接复制数据库中已有表的结构和数据,用这种方式构建一个表,十分方便、快捷,其语法格式为:

```
CREATE [TEMPORARY] TABLE [IF NOT EXISTS] 表名
    [ ( ) LIKE 已有表名 [ ] ]
    |[AS ( 表达式 )];
```

说明：

- 使用 LIKE 关键字创建一个与已有表名相同结构的新表，列名、数据类型、空指定和索引也将复制，但是表的内容不会复制，因此创建的新表是一个空表。
- 使用 AS 关键字可以复制表的内容，但索引和完整性约束是不会复制的。表达式是复制内容部分，例如可以是一条 SELECT 语句。

【例 3.4】 在 mytest 数据库中，用复制的方式创建一个名为 user_copy1 的表，表结构直接取自 user 表；再创建一个名为 user_copy2 的表，其结构和内容（数据）都取自 user 表。

（1）创建 user_copy1 表：

```
USE mytest
CREATE TABLE user_copy1 LIKE user;
```

（2）创建 user_copy2 表：

```
CREATE TABLE user_copy2 AS (select * from user);
```

执行过程及结果，如图 3.4 所示。

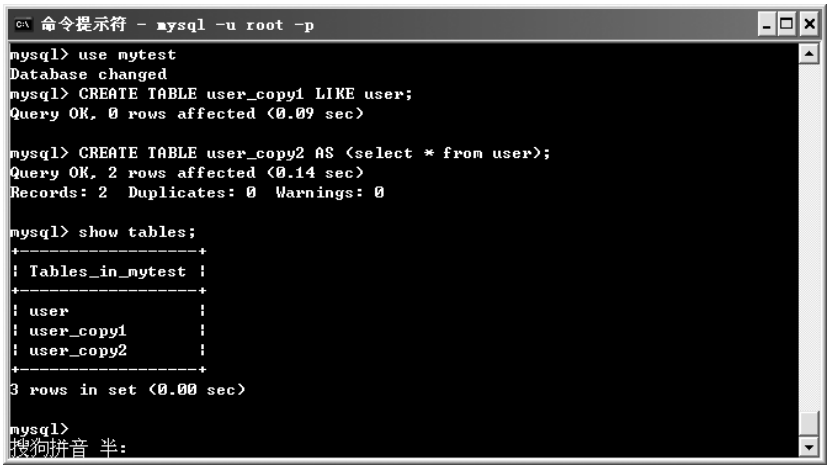

图 3.4　用复制的方式创建表

查询 user_copy1 表中没有记录，而 user_copy2 表中包含 user 表中所有记录，如图 3.5 所示。

3.2.2　修改表

1. 修改表结构

ALTER TABLE 用于更改原有表的结构。例如，可以增加（删减）列、创建（取消）索引、更改原有列的类型、重新命名列或表，还可以更改表的评注和表的类型。其语法格式为：

```
ALTER [IGNORE] TABLE 表名
    选项 ...
```

选项：

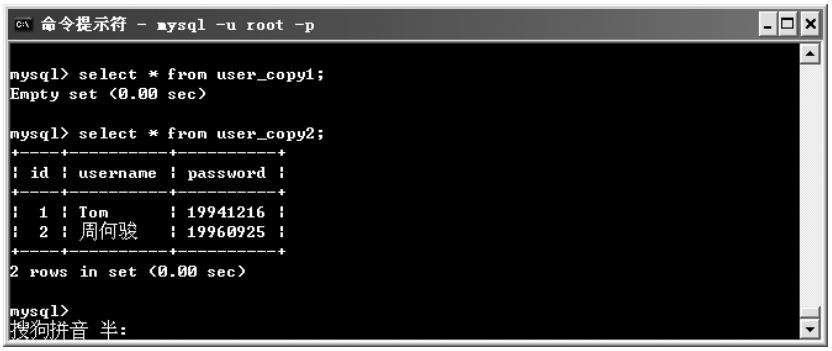

图 3.5 查询表记录

```
ADD [COLUMN] 列定义 [FIRST|AFTER 列名 ]                          /* 添加列 */
|ALTER [COLUMN] 列名{SET DEFAULT literal|DROP DEFAULT}           /* 修改默认值 */
|CHANGE [COLUMN] 列名  原列名 [FIRST|AFTER 列名]                  /* 列名重定义 */
|MODIFY [COLUMN] 列定义 [FIRST|AFTER 列名]                        /* 修改列数据类型 */
|DROP [COLUMN] 列名                                               /* 删除列 */
|RENAME [TO] 新表名                                               /* 重命名该表 */
|ORDER BY 列名                                                    /* 排序 */
|CONVERT TO CHARACTER SET 字符集名 [COLLATE 校对规则名]           /* 将字符集转换为二进制 */
|[DEFAULT] CHARACTER SET 字符集名 [COLLATE 校对规则名]            /* 修改默认字符集 */
|表选项
|列或表中索引项的增、删、改(详细见第 5 章)
```

说明:

- IGNORE——是 MySQL 相对于标准 SQL 的扩展。若在修改后的新表中存在重复关键字,如果没有指定 IGNORE,当重复关键字错误发生时操作失败。如果指定了 IGNORE,则对于有重复关键字的行只使用第一行,其他有冲突的行被删除。
- ADD[COLUMN]子句——向表中增加新列。例如,在表 user 中增加新的一列 a:

```
user mytest
alter table user add column a tinyint null;
```

- FIRST|AFTER 列名——表示在某列的前或后添加,不指定则添加到最后。
- ALTER [COLUMN]子句——修改表中指定列的默认值。
- CHANGE [COLUMN]子句——修改列的名称。重命名时,需给定旧的列名和新的列名称和数据类型。例如,要把一个 INTEGER 列的名称从 a 变更到 b:

```
alter table user change a b integer;
```

- MODIFY [COLUMN]子句——修改指定列的数据类型。例如,要把一个列的数据类型改为 BIGINT:

```
alter table user modify b bigint not null;
```

注意：若表中该列所存数据的数据类型与将要修改的列的类型冲突，则发生错误。例如，原来 CHAR 类型的列要修改成 INT 类型，而原来列值中有字符型数据，则无法修改。

- DROP 子句——从表中删除列或约束。
- RENAME 子句——修改该表的表名。例如，将表 user_copy1 改名为 use1：

```
alter table user_copy1 rename to usera;
```

- ORDER BY 子句——用于在创建新表时，让各行按一定的顺序排列。注意，在插入和删除后，表不会仍保持此顺序。在对表做了大的改动后，通过使用此选项可提高查询效率。在有些情况下，如果表按列排序，对于 MySQL 来说，排序可能会更简单。
- 表选项——修改表选项，具体定义与 CREATE TABLE 语句中一样。

可以在一个 ALTER TABLE 语句里写入多个 ADD、ALTER、DROP 和 CHANGE 子句，中间用逗号分开。这是 MySQL 相对于标准 SQL 的扩展，在标准 SQL 中，每个 ALTER TABLE 语句中的每个子句只允许使用一次。

【例 3.5】 在 xscj 数据库的 xs 表中，增加“奖学金等级”一列，并将表中的“姓名”列删除。

```
user xscj
alter table xs
    add 奖学金等级 tinyint null,
    drop column 姓名;
```

执行后，xs 表的结构如图 3.6 所示。

```
mysql> alter table xs
    ->     add 奖学金等级 tinyint null,
    ->     drop column 姓名;
Query OK, 0 rows affected (0.67 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> describe xs;
+------------+------------+------+-----+---------+-------+
| Field      | Type       | Null | Key | Default | Extra |
+------------+------------+------+-----+---------+-------+
| 学号       | char(6)    | NO   | PRI | NULL    |       |
| 专业名     | char(10)   | YES  |     | NULL    |       |
| 性别       | tinyint(1) | NO   |     | 1       |       |
| 出生日期   | date       | NO   |     | NULL    |       |
| 总学分     | tinyint(1) | YES  |     | NULL    |       |
| 照片       | blob       | YES  |     | NULL    |       |
| 备注       | text       | YES  |     | NULL    |       |
| 奖学金等级 | tinyint(4) | YES  |     | NULL    |       |
+------------+------------+------+-----+---------+-------+
8 rows in set (0.01 sec)
```

图 3.6　修改后的 xs 表结构

为了在后面演示表记录操作的方便，此处完成后要及时地将 xs 表改回原样，语句如下：

```
alter table xs
    add 姓名 char(8) not null after 学号,
    drop column 奖学金等级;
```

这样，xs 表结构就又恢复原状了。

2. 更改表名

除了上面的 ALTER TABLE 命令外,还可以直接用 RENAME TABLE 语句来更改表的名字,其语法格式为:

```
RENAME TABLE 老表名 TO 新表名 ...
```

【例 3.6】 将 mytest 数据库 usera 表重命名为 user1,user_copy2 表重命名为 user2。

```
rename table usera to user1,user_copy2 to user2;
```

3.2.3 删除表

需要删除一个表时可以使用 DROP TABLE 语句,其语法格式为:

```
DROP [TEMPORARY] TABLE [IF EXISTS] 表名 ...
```

说明:这个命令将表的描述、表的完整性约束、索引及和表相关的权限等一并删除。

【例 3.7】 删除表 uesra。

```
drop table if exists usera;
```

3.2.4 表结构特点

在使用工具或 SQL 语句创建表之前,先要确定表的名字、所包含的列名、列的数据类型及长度、是否可为空值、默认值情况、是否要使用及何时使用约束、默认设置或规则及所需索引、哪些列是主键、哪些列是外键等,这些构成表的结构。

1. 空值概念

空值通常表示未知、不可用或将在以后添加的数据。若一个列允许为空值,则向表中输入记录值时可不为该列给出具体值;而一个列若不允许为空值,则在输入时必须给出该列的具体值。

注意:表的关键字不允许为空值。空值不能与数值数据 0 或字符类型的空字符混为一谈。任意两个空值都不相等。

2. 列的标志属性

对任何表都可创建包含系统所生成序号值的一个标志列,该序号值唯一标志表中的一列,可以作为键值。每个表只能有一个列设置为标志属性,该列只能是 decimal、int、numeric、smallint、bigint 或 tinyint 数据类型。定义标志属性时,可指定其种子(即起始)值、增量值,二者的默认值均为 1。系统自动更新标志列值,标志列值不允许空值。

3. 隐含地改变列类型

在下列情况下,MySQL 隐含地改变在一个 CREATE TABLE 语句中给出的一个列类型(这也可能在 ALTER TABLE 语句上出现)。

(1) 长度小于 4 的 varchar 被改变为 char。

(2) 如果在一个表中的任何列有可变长度,结果使整个行是变长的。因此,如果一张表

包含任何变长的列(varchar、text 或 Blob),所有大于三个字符的 char 列被改变为 varchar 列。这在任何方面都不影响用户如何使用列。在 MySQL 中这种改变可以节省空间并且使表操作更加快捷。

(3) timestamp 的显示尺寸必须是偶数且在 2～14 内。如果指定 0 显示尺寸或比 14 大,尺寸被强制为 14。从 1～13 的奇数值尺寸被强制为下一个更大的偶数。

(4) 不能在一个 timestamp 列中存储一个 NULL,将它设为 NULL 默认为当前的日期和时间。

如果想要知道 MySQL 是否使用了除指定的以外的一种列类型,在创建表之后,使用一个 DESCRIBE 语句即可。DESCRIBE 语句将在 3.4 节详细介绍。

3.3　MySQL 表记录操作

创建数据库和表后,需要对表中的数据(记录)进行操作,包括插入、修改和删除操作,可以通过 SQL 语句操作表记录,也可以用第 2 章介绍的各种 MySQL 界面工具来操作。与界面操作相比,通过 SQL 语句操作更为灵活,功能更强大。

3.3.1　插入记录

一旦创建了数据库和表,下一步就是向表里插入数据记录。通过 INSERT 或 REPLACE 语句可以向表中插入一行或多行记录。

1. 插入新记录

向表中插入全新的记录用 INSERT 语句,其语法格式为:

```
INSERT [LOW_PRIORITY|DELAYED|HIGH_PRIORITY] [IGNORE]
    [INTO] 表名 [(列名,...)]
    VALUES ({expr|DEFAULT},...),(...),...
    |SET 列名={expr|DEFAULT}, ...
    [ ON DUPLICATE KEY UPDATE 列名=expr, ... ]
```

说明:

- 列名——需要插入数据的列名。如果要给全部列插入数据,列名可以省略。如果只给表的部分列插入数据,需要指定这些列。对于没有指出的列,它们的值根据列默认值或有关属性来确定,MySQL 处理的原则是:
  ① 具有 IDENTITY 属性的列,系统生成序号值来唯一标志列。
  ② 具有默认值的列,其值为默认值。
  ③ 没有默认值的列,若允许为空值,则其值为空值;若不允许为空值,则出错。
  ④ 类型为 timestamp 的列,系统自动赋值。
- VALUES 子句——包含各列需要插入的数据清单,数据的顺序要与列的顺序相对应。若表名后不给出列名,则在 VALUES 子句中要给出每一列(除 IDENTITY 和 timestamp 类型的列)的值,如果列值为空,则值必须置为 NULL,否则会出错。VALUES 子句中的值:

① expr——可以是一个常量、变量或一个表达式,也可以是空值 NULL,其值的数据类型要与列的数据类型一致。例如,列的数据类型为 int,插入的数据是'aaa'就会出错。当数据为字符型时要加单引号。

② DEFAULT——指定为该列的默认值。前提是该列原先已经指定了默认值。

如果列清单和 VALUES 清单都为空,则 INSERT 会创建一行,每个列都设置成默认值。INSERT 语句支持下列修饰符:

- LOW_PRIORITY——可以使用在 INSERT、DELETE 和 UPDATE 等操作中,当原有客户端正在读取数据时,延迟操作的执行,直到没有其他客户端从表中读取为止。
- DELAYED——若使用此关键字,则服务器会把待插入的行放到一个缓冲器中,而发送 INSERT DELAYED 语句的客户端会继续运行。如果表正在被使用,则服务器会保留这些行。当表空闲时,服务器开始插入行,并定期检查是否有新的读取请求(仅适用于 MyISAM、MEMORY 和 ARCHIVE 表)。
- HIGH_PRIORITY——可以使用在 SELECT 和 INSERT 操作中,使操作优先执行。
- IGNORE——使用此关键字,在执行语句时出现的错误就会被当作警告处理。
- ON DUPLICATE KEY UPDATE——使用此选项插入行后,若导致 UNIQUE KEY 或 PRIMARY KEY 出现重复值,则根据 UPDATE 后的语句修改旧行(使用此选项时 DELAYED 被忽略)。
- SET 子句——用于给列指定值,使用 SET 子句时表名的后面省略列名。要插入数据的列名在 SET 子句中指定,列名为指定列名,等号后面为指定数据,未指定的列,列值为默认值。

从 INSERT 的语法格式可以看到,使用 INSERT 语句可以向表中插入一行记录,也可以插入多行记录,插入的行可以给出每列的值,也可只给出部分列的值,还可以向表中插入其他表的数据。

【例 3.8】 向学生成绩数据库(xscj)的表 xs(表中列包括学号、姓名、专业名、性别、出生日期、总学分、照片、备注)中插入如下一行:

```
081101,王林,计算机,1,1994-02-10,50,NULL,NULL
```

使用下列语句:

```
use xscj
insert into xs
    values('081101', '王林', '计算机', 1, '1994-02-10', 50, null, null);
```

若表 xs 中专业的默认值为“计算机”,照片、备注默认值为 NULL,插入数据也可以使用如下命令:

```
insert into xs (学号, 姓名, 性别, 出生日期, 总学分)
    values('081101', '王林', 1, '1994-02-10', 50);
```

与下面这个命令的效果相同:

```
insert into xs
    values('081101', '王林', default, 1, '1994-02-10', 50, null, null);
```

当然，也可以使用 SET 子句来实现：

```
insert into xs
    set 学号='081101', 姓名='王林', 专业=default, 性别=1, 出生日期='1994-02-10',
    总学分=50;
```

执行结果如图 3.7 所示。

```
mysql> select * from xs;
+--------+------+--------+------+------------+--------+------+------+
| 学号   | 姓名 | 专业名 | 性别 | 出生日期   | 总学分 | 照片 | 备注 |
+--------+------+--------+------+------------+--------+------+------+
| 081101 | 王林 | 计算机 |    1 | 1994-02-10 |     50 | NULL | NULL |
+--------+------+--------+------+------------+--------+------+------+
1 row in set (0.00 sec)
```

图 3.7　修改后的 xs 表记录

注意：若原有行中存在 PRIMARY KEY 或 UNIQUE KEY，而插入的数据行中含有与原有行中 PRIMARY KEY 或 UNIQUE KEY 相同的列值，则 INSERT 语句无法插入此行。要插入这行数据需要使用 REPLACE 语句，它的用法与 INSERT 语句基本相同。

2. 从已有表中插入新记录

使用 INSERT INTO…SELECT…，可以快速地从一个或多个已有的表记录向表中插入多个行，其语法格式为：

```
INSERT [LOW_PRIORITY|HIGH_PRIORITY] [IGNORE]
    [INTO] 表名 [(列名,...)]
    SELECT 语句
    [ON DUPLICATE KEY UPDATE 列名=expr, ...]
```

说明：SELECT 语句中返回的是一个查询到的结果集，INSERT 语句将这个结果集插入到指定表中，但结果集中每行数据的字段数、字段的数据类型要与被操作的表完全一致。有关 SELECT 语句会在第 7 章具体介绍。

【例 3.9】　将 mytest 数据库 user 表记录插入到 user1 表中。

```
user mytest
insert into user1 select * from user;
```

命令执行前后的效果如图 3.8 所示。

3. 替换旧记录

REPLACE 语句可以在插入数据之前将与新记录冲突的旧记录删除，从而使新记录能够替换旧记录，正常插入。REPLACE 语句格式与 INSERT 相同。

【例 3.10】　若上例中的记录行已经插入，其中学号为主键(PRIMARY KEY)，现在想再插入下列一行记录：

```
mysql> select * from user;
+----+----------+----------+
| id | username | password |
+----+----------+----------+
|  1 | Tom      | 19941216 |
|  2 | 周何骏   | 19960925 |
+----+----------+----------+
2 rows in set (0.00 sec)

mysql> select * from user1;
Empty set (0.00 sec)
```

```
mysql> insert into user1 select * from user;
Query OK, 2 rows affected (0.02 sec)
Records: 2  Duplicates: 0  Warnings: 0

mysql> select * from user1;
+----+----------+----------+
| id | username | password |
+----+----------+----------+
|  1 | Tom      | 19941216 |
|  2 | 周何骏   | 19960925 |
+----+----------+----------+
2 rows in set (0.00 sec)
```

图 3.8　插入前后 user1 表记录

```
081101,刘华,通信工程,1,1995-03-08,48,NULL,NULL
```

若直接使用 INSERT 语句,会产生如图 3.9 所示的错误。

```
mysql> insert into xs
    ->     values('081101', '刘华', '通信工程', 1, '1995-03-08', 48, null, null)
;
ERROR 1062 (23000): Duplicate entry '081101' for key 'PRIMARY'
```

图 3.9　错误提示

使用 REPLACE 语句,则可以成功插入,如图 3.10 所示。

```
mysql> replace into xs
    ->     values('081101', '刘华', '通信工程', 1, '1995-03-08', 48, null, null)
;
Query OK, 2 rows affected (0.03 sec)
```

图 3.10　成功插入

4. 插入图片

MySQL 还支持图片的插入,图片一般可以以路径的形式来存储,即插入图片可以采用插入图片的存储路径的方式。当然也可以直接插入图片本身,只要用 LOAD_FILE 函数即可。

【例 3.11】　向 xs 表中插入一行记录:

```
081102,程明,计算机,1,1995-02-01,50,picture.jpg,NULL
```

设照片路径为 D:\IMAGE\picture.jpg。使用如下语句:

```
insert into xs
    values('081102', '程明', '计算机', 1, '1995-02-01', 50, ' D:\IMAGE\ picture.
    jpg', null);
```

也可使用以下语句直接存储图片本身:

```
insert into xs
    values('081102', '程明', '计算机', 1, '1995-02-01', 50, load_file('D:\ IMAGE\
    picture.jpg'), null);
```

执行结果如图 3.11 所示。

```
+--------+------+----------+------+------------+--------+--------------------+-
----+
| 学号   | 姓名 | 专业名   | 性别 | 出生日期   | 总学分 | 照片               |
备注 |
+--------+------+----------+------+------------+--------+--------------------+-
----+
| 081101 | 刘华 | 通信工程 |    1 | 1995-03-08 |     48 | NULL               |
NULL |
| 081102 | 程明 | 计算机   |    1 | 1995-02-01 |     50 |  D:IMAGEpicture.jpg |
NULL |
+--------+------+----------+------+------------+--------+--------------------+-
```

图 3.11　例 3.11 执行结果

3.3.2　修改记录

要修改表中的一行记录，使用 UPDATE 语句，UPDATE 可用来修改一个表，也可以修改多个表。

1. 修改单个表

使用 UPDATE 修改单个表的语法格式为：

```
UPDATE [LOW_PRIORITY] [IGNORE] 表名
    SET 列名 1=expr1 [, 列名 2=expr2 ...]
    [WHERE 条件]
    [ORDER BY ...]
    [LIMIT row_count]
```

说明：

- 若语句中不设定 WHERE 子句，则更新所有行。列名 1、列名 2……为要修改列，列值为 expr，expr 可以是常量、变量、列名或表达式。可以同时修改所在数据行的多个列值，中间用逗号隔开。
- WHERE 子句——指定的删除记录条件。如果省略 WHERE 子句则删除该表的所有行。
- ORDER BY 子句——各行按照子句中指定的顺序进行删除，此子句只在与 LIMIT 联用时才起作用。子句 ORDER BY 和 LIMIT 的具体定义将在第 7 章中介绍。
- LIMIT 子句——用于告知服务器，在控制命令被返回到客户端前，被删除的行的最大值。

【例 3.12】　将学生成绩数据库(xscj)的学生(xs)表中的所有学生的总学分都增加 10。将姓名为“刘华”的学生的备注填写为“辅修计算机专业”，学号改为 081250。

```
update xs
    set 总学分=总学分+10;
update xs
    set 学号='081250' , 备注='辅修计算机专业'
    where 姓名='刘华';
select 学号, 姓名, 总学分, 备注 from xs;
```

执行结果如图 3.12 所示。

这样,可以发现表中所有学生的总学分已经都增加了 10,姓名为“刘华”的学生的备注填写为“辅修计算机专业”,学号也改成了 081250。

学号	姓名	总学分	备注
081102	程明	60	NULL
081250	刘华	58	辅修计算机专业

图 3.12 例 3.12 执行结果

2. 修改多个表

使用 UPDATE 修改多个表的语法格式为:

```
UPDATE [LOW_PRIORITY] [IGNORE] 表名,表名...
    SET 列名 1=expr1 [, 列名 2=expr2 ...]
    [WHERE 条件]
```

【例 3.13】 mytest 数据库表 user 和表 user2 中都有两个字段:id int(11)、password varchar(10),其中 id 为主键。当表 user 中 id 值与 user2 中 id 值相同时,将表 user 中对应的 password 值修改为 11111111,将表 user2 中对应的 password 值改为 22222222。

```
user mytest
update user, user2
    set user.password='11111111', user2.password='22222222'
    where user.id=user2.id;
```

修改后的结果如图 3.13 所示。

mysql> select * from user;

id	username	password
1	Tom	11111111
2	周何骏	11111111

(a) 修改后user表

mysql> select * from user2;

id	username	password
1	Tom	22222222
2	周何骏	22222222

(b) 修改后user2表

图 3.13 同时修改两个表

3.3.3 删除记录

DELETE 语句或 TRUNCATE TABLE 语句可以用于删除表中的一行或多行记录。

1. 删除满足条件的行

使用 DELETE 语句删除表中满足条件的记录行。

从单个表中删除的语法格式为:

```
DELETE [LOW_PRIORITY] [QUICK] [IGNORE] FROM 表名
    [WHERE 条件]
    [ORDER BY ...]
    [LIMIT row_count]
```

说明:

- QUICK 修饰符——可以加快部分种类的删除操作的速度。
- FROM 子句——用于说明从何处删除数据,表名为要删除数据的表名。

- WHERE 子句——指定的删除记录条件。如果省略 WHERE 子句则删除该表的所有行。
- ORDER BY 子句——各行按照子句中指定的顺序进行删除，此子句只在与 LIMIT 联用时才起作用。ORDER BY 子句和 LIMIT 子句的具体定义，将在第 4 章 MySQL 数据库查询语句中介绍。
- LIMIT 子句——用于告知服务器，在控制命令被返回到客户端前，被删除的行的最大值。

【例 3.14】 删除 mytest 数据库中 user2 表“周何骏”记录。

```
use mytest
delete from person
    where username='周何骏';
```

或

```
delete from xs
    where id=2;
```

2. 从多个表中删除行

删除操作若要在多个表中进行，其语法格式为：

```
DELETE [LOW_PRIORITY] [QUICK] [IGNORE] 表名[.*] [, 表名[.*] ...]
    FROM table_references
    [WHERE where_definition]
```

或

```
DELETE [LOW_PRIORITY] [QUICK] [IGNORE]
    FROM 表名[.*] [, 表名[.*] ...]
    USING table_references
    [WHERE where_definition]
```

说明：对于第一种语法格式，只删除列于 FROM 子句之前的表中对应的行；对于第二种语法格式，只删除列于 FROM 子句之中（在 USING 子句之前）的表中对应的行。作用是，可以同时删除多个表中的行，并使用其他的表进行搜索。

【例 3.15】 删除 user1 中 id 值等于 user 的 id 值的所有行以及 user2 中 id 值等于 user 的 id 值的所有行。使用如下语句：

```
DELETE    user1, user2
    FROM  user1, user2, user
    WHERE user1.id=user.id AND user2.id=user.id;
```

执行结果如图 3.14 所示。

3. 清除表数据

使用 TRUNCATE TABLE 语句将删除指定表中的所有数据，因此也称其为清除表数据语句，其语法格式为：

```
mysql> DELETE   user1, user2
    ->      FROM  user1, user2, user
    ->      WHERE  user1.id=user.id  AND  user2.id=user.id;
Query OK, 4 rows affected (0.02 sec)

mysql> select * from user;
+----+----------+----------+
| id | username | password |
+----+----------+----------+
|  1 | Tom      | 19941216 |
|  2 | 周何骏   | 19960925 |
+----+----------+----------+
2 rows in set (0.00 sec)

mysql> select * from user1;
Empty set (0.00 sec)

mysql> select * from user2;
Empty set (0.00 sec)
```

图 3.14　例 3.15 执行结果

```
TRUNCATE TABLE 表名
```

说明：由于 TRUNCATE TABLE 语句将删除表中的所有数据，且无法恢复，因此使用时必须十分小心！

TRUNCATE TABLE 在功能上与不带 WHERE 子句的 DELETE 语句(如 DELETE FROM XS)相同，二者均删除表中的全部行。但 TRUNCATE TABLE 比 DELETE 速度快，且使用的系统和事务日志资源少。DELETE 语句每次删除一行，并在事务日志中为所删除的每行记录一项。而 TRUNCATE TABLE 通过释放存储表数据所用的数据页来删除数据，并且只在事务日志中记录页的释放。使用 TRUNCATE TABLE，AUTO_INCREMENT 计数器被重新设置为该列的初始值。

对于参与了索引和视图的表，不能使用 TRUNCATE TABLE 删除数据，而应使用 DELETE 语句。

3.4　MySQL 数据库信息显示

在使用 MySQL 时，经常需要查看数据库本身的一些信息，如系统中已有哪些数据库、某数据库中已建立了哪几张表、某个表的结构等，通常使用 SHOW 和 DESCRIBE 这两个语句来显示这些常用信息，在以后的学习中也会经常用到它们。

1. SHOW 语句

SHOW tables 或 SHOW tables from database_name：显示当前数据库中所有表的名称。

SHOW databases：显示 MySQL 中所有数据库的名称。

SHOW columns from table_name from database_name 或 SHOW columns from database_name.table_name：显示表中列的名称。

SHOW grants for user_name：显示一个用户的权限，显示结果类似于 grant 命令。

SHOW index from table_name：显示表的索引。

SHOW status：显示一些系统特定资源的信息，如正在运行的线程数量。

SHOW variables：显示系统变量的名称和值。

SHOW processlist：显示系统中正在运行的所有进程，即当前正在执行的查询。大多数用户可查看自己的进程，如果拥有 process 权限，还可查看所有人的进程，包括密码。

SHOW table status：显示当前使用或者指定的 database 中的每个表的信息。信息包括表类型和表的最新更新时间。

SHOW privileges：显示服务器所支持的不同权限。

SHOW create database database_name：显示创建某一个数据库的 CREATE DATABASE 语句。

SHOW create table table_name：显示创建一个表的 CREATE TABLE 语句。

SHOW events：显示所有事件的列表。

SHOW innodb status：显示 innoDB 存储引擎的状态。

SHOW logs：显示 BDB 存储引擎的日志。

SHOW warnings：显示最后一个执行的语句所产生的错误、警告和通知。

SHOW errors：只显示最后一个执行语句所产生的错误。

SHOW [storage] engines：显示安装后的可用存储引擎和默认引擎。

SHOW procedure status：显示数据库中所有存储过程基本信息，包括所属数据库、存储过程名称、创建时间等。

SHOW create procedure sp_name：显示某一个存储过程的详细信息。

2. DESCRIBE 语句

DESCRIBE 语句用于显示表中各列的信息，结果与 SHOW columns…from…语句相同。其语法格式为：

```
{DESCRIBE|DESC} 表名 [列名|wild ]
```

说明：

- DESC 是 DESCRIBE 的简写，二者用法相同。
- 列名可以是一个列名称，或一个包含%和_通配符的字符串，用于获得对于带有与字符串相匹配的名称的各列的输出。没有必要在引号中包含字符串，除非其中包含空格或其他特殊字符。

【例 3.16】 用 DESCRIBE 语句查看 xscj 数据库 xs 表的列的信息。

```
use xscj
describe xs;
```

执行结果如图 3.15 所示。

Field	Type	Null	Key	Default	Extra
学号	char(6)	NO	PRI	NULL	
姓名	char(8)	NO		NULL	
专业名	char(10)	YES		NULL	
性别	tinyint(1)	NO		1	
出生日期	date	NO		NULL	
总学分	tinyint(1)	YES		NULL	
照片	blob	YES		NULL	
备注	text	YES		NULL	

图 3.15　例 3.16 执行结果

【例 3.17】 查看 xs 表学号列的信息。

```
use xscj
desc xs 学号;
```

执行结果如图 3.16 所示。

Field	Type	Null	Key	Default	Extra
学号	char(6)	NO	PRI	NULL	

图 3.16 例 3.17 执行结果

习题 3

1. 写出创建产品销售数据库 cpxs 及其中表的语句,库中所包含的表如下。

产品表:产品编号,产品名称,价格,库存量。

销售商表:客户编号,客户名称,地区,负责人,电话。

产品销售表:销售日期,产品编号,客户编号,数量,销售额。

要求:全部使用本章所讲的命令行方式创建,不要借助界面工具。

2. 简要说明空值的概念及其作用。

3. 写出命令行语句,对 cpxs 数据库的产品表进行如下操作。

(1) 插入如下记录:

```
0001    空调    3000    200
0203    冰箱    2500    100
0301    彩电    2800    50
0421    微波炉  1500    50
```

(2) 将产品表中每种产品的价格打 8 折。

(3) 将产品表中价格打 8 折后低于 50 元的产品记录删除。