

第3章 添加、更新与删除数据

学习目标

- ◆ 学会为数据表的字段添加数据
- ◆ 学会更新数据表中的数据
- ◆ 学会删除数据表中的数据

通过上一章的学习，相信大家对数据库和数据表的基本操作有了一定了解，但要想操作数据库中的数据，必须得通过 MySQL 提供的数据库操作语言实现，包括插入数据的 INSERT 语句，更新数据的 UPDATE 语句以及删除数据的 DELETE 语句，本章将针对这些操作进行详细地讲解。

3.1 添加数据

要想操作数据表中的数据，首先得保证数据表中存在数据。MySQL 使用 INSERT 语句向数据表中添加数据，并且根据添加方式的不同分为三种，分别是为表的所有字段添加数据、为表的指定字段添加数据、同时添加多条记录。本节将针对这三种添加数据的方式进行详细地讲解。

3.1.1 为表中所有字段添加数据

通常情况下，向数据表中添加的新记录应该包含表所有字段，即为该表中的所有字段添加数据，为表中所有字段添加数据的 INSERT 语句有两种，具体如下：

1、INSERT 语句中指定所有字段名

向表中添加新记录时，可以在 INSERT 语句中列出表的所有字段名，其语法格式如下所示：

```
INSERT INTO 表名 (字段名 1, 字段名 2, .....)  
VALUES (值 1, 值 2, .....);
```

在上述语法格式中，“字段名 1，字段名 2，.....”表示数据表中的字段名称，此处必须列出表所有字段的名称；“值 1，值 2，.....”表示每个字段的值，每个值的顺序、类型必须与对应的字段相匹配。

【例 3-1】向 student 表中添加一条新记录，记录中 id 字段的值为 1，name 字段的值为 ‘zhangsan’，grade 字段的值为 98.5。

在添加新记录之前需要先创建一个数据库 chapter03，创建数据库的 SQL 语句如下所示：

```
CREATE DATABASE chapter03;
```

选择使用数据库 chapter03，SQL 语句如下：

```
USE chapter03;
```

在数据库中创建一个表 student 用于存储学生信息，创建 student 表的 SQL 语句如下所示：

```
CREATE TABLE student(  
    id    INT(4),
```

```

    name    VARCHAR(20) NOT NULL,
    grade    FLOAT
);

```

使用 **INSERT** 语句向 **student** 表中插入一条数据，SQL 语句如下所示：

```

INSERT INTO student(id,name,grade)
VALUES (1,'zhangsan',98.5);

```

当上述 **SQL** 语句执行成功后，会在表 **student** 中添加一条数据。为了验证数据是否添加成功，使用 **SELECT** 语句查看 **student** 表中的数据，查询结果如下：

```

mysql> SELECT * FROM student;
+-----+-----+-----+
| id  | name    | grade |
+-----+-----+-----+
| 1  | zhangsan | 98.5  |
+-----+-----+-----+
1 row in set (0.00 sec)

```

从查询结果可以看出，**student** 表中成功地添加了一条记录，“1 row in set”表示查询出了一条记录。关于 **SELECT** 查询语句的相关知识，将在第4章进行详细讲解，这里大家有个大致印象即可。需要注意的是，使用 **INSERT** 语句添加记录时，表名后的字段顺序可以与其在表中定义的顺序不一致，它们只需要与 **VALUES** 中值的顺序一致即可。

【例 3-2】 向 **student** 表中添加一条新记录，记录中 **id** 字段的值为 2，**name** 字段的值为 ‘lisi’，**grade** 字段的值为 95，SQL 语句如下所示：

```

INSERT INTO student(name,grade,id)
VALUES ('lisi',95,2);

```

执行结果如下所示：

```

mysql> INSERT INTO student(name,grade,id)
-> VALUES('lisi',95,2);
Query OK, 1 row affected (0.02 sec)

```

从执行结果可以看到，三个字段 **id**，**name** 和 **grade** 的顺序进行了调换，同时 **VALUES** 后面值的顺序也做了相应地调换，**INSERT** 语句同样执行成功，接下来通过查询语句查看数据是否成功添加，执行结果如下所示：

```

mysql> select * from student;
+-----+-----+-----+
| id  | name    | grade |
+-----+-----+-----+
| 1  | zhangsan | 98.5  |
| 2  | lisi     | 95    |
+-----+-----+-----+
2 rows in set (0.00 sec)

```

从查询结果可以看出，**student** 表中同样成功地添加了一条记录。

2、INSERT 语句中不指定字段名

在 **MySQL** 中，可以通过不指定字段名的方式添加记录，其基本的语法格式如下所示：

```

INSERT INTO 表名 VALUES(值 1, 值 2, .....);

```

在上述格式中，“值 1，值 2，……”用于指定要添加的数据。需要注意的是，由于 **INSERT** 语句中没

有指定字段名，添加的值的顺序必须和字段在表中定义的顺序相同。

【例 3-3】 向 student 表中添加一条新记录，记录中 id 字段的值为 3，name 字段的值为 ‘wangwu’，grade 字段的值为 61.5，INSERT 语句如下所示：

```
INSERT INTO student
VALUES (3, 'lisi', 61.5);
```

SQL 语句执行成功后，同样会在 student 表中添加一条新的记录。为了验证数据是否添加成功，使用 SELECT 语句查看 student 表中的数据，查询结果如下所示：

```
mysql> select * from student;
+-----+-----+-----+
| id    | name    | grade |
+-----+-----+-----+
| 1     | zhangsan | 98.5  |
| 2     | lisi     | 95    |
| 3     | wangwu   | 61.5  |
+-----+-----+-----+
3 rows in set (0.00 sec)
```

从上述结果可以看出，student 表中成功添加了一条记录。由此可见，INSERT 语句中不指定字段名同样成功添加数据。

3.1.2 为表的指定字段添加数据

为表的指定字段添加数据，就是在 INSERT 语句中只向部分字段中添加值，而其他字段的值为表定义时的默认值。为表的指定字段添加数据的基本语法格式如下所示：

```
INSERT INTO 表名 (字段 1, 字段 2, ...)
VALUES (值 1, 值 2, ...)
```

在上述语法格式中，“字段 1，字段 2，...”表示数据表中的字段名称，此次只指定表中部分字段的名称。“值 1，值 2，.....”表示指定字段的值，每个值的顺序、类型必须与对应的字段相匹配。

【例 3-4】 向 student 表中添加一条新记录，记录中 id 字段的值为 4，name 字段的值为“zhaoliu”，grade 字段不指定值，SQL 语句如下所示：

```
INSERT INTO student(id,name)
VALUES (4, 'zhaoliu');
```

上述 SQL 语句执行成功后，会向 student 表中添加一条新的数据。为了验证数据是否添加成功，使用 SELECT 语句查看 student 表，结果如下所示：

```
mysql> select * from student;
+-----+-----+-----+
| id    | name    | grade |
+-----+-----+-----+
| 1     | zhangsan | 98.5  |
| 2     | lisi     | 95    |
| 3     | wangwu   | 61.5  |
| 4     | zhaoliu  | NULL  |
+-----+-----+-----+
4 rows in set (0.00 sec)
```

从查询结果可以看出，新记录添加成功，但是 `grade` 字段的值为 `NULL`。这是因为在添加新记录时，如果没有为某个字段赋值，系统会自动为该字段添加默认值。通过 SQL 语句“`SHOW CREATE TABLE student\G`”可以查看 `student` 表的详细结构，SQL 执行结果如下所示：

```
mysql> SHOW CREATE TABLE student\G
***** 1. row *****
      Table: student
Create Table: CREATE TABLE 'student' (
  'id' int(4) DEFAULT NULL,
  'name' varchar(20) NOT NULL,
  'grade' float DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8
1 row in set (0.00 sec)
```

从表的详细结构中可以看出，`grade` 字段的默认值为 `NULL`。本例中没有为 `grade` 字段赋值，系统会自动为其添加默认值 `NULL`。

需要注意的是，如果某个字段在定义时添加了非空约束，但没有添加 `default` 约束，那么插入新记录时必须为该字段赋值，否则数据库系统会提示错误。

【例 3-5】向 `student` 表中添加一条新记录，记录中 `id` 字段的值为 5，`grade` 字段的值为 97，`name` 字段不指定值，SQL 语句如下所示：

```
INSERT INTO student(id,grade)
VALUES (5,97);
```

执行结果如下所示：

```
mysql> INSERT INTO student(id,grade)
-> VALUES (5,97);
ERROR 1364 (HY000): Field 'name' doesn't have a default value
```

从执行结果可以看出，执行 `INSERT` 语句时发生了错误，发生错误的原因是 `name` 字段没有指定默认值，且添加了非 `NULL` 约束。接下来，通过查询语句查看数据是否成功添加，执行结果如下所示：

```
mysql> SELECT * FROM student;
+-----+-----+-----+
| id   | name   | grade |
+-----+-----+-----+
| 1   | zhangsan | 98.5 |
| 2   | lisi    | 95   |
| 3   | wangwu  | 61.5 |
| 4   | zhaoliu | NULL  |
+-----+-----+-----+
4 rows in set (0.00 sec)
```

通过查询结果可以看到，`student` 表中仍然只有 4 条记录，新记录没有添加成功。

为指定字段添加数据时，指定字段也无需与其在表中定义的顺序一致，它们只要与 `VALUES` 中值的顺序一致即可。

【例 3-6】向 `student` 表中添加一条新记录，记录中 `name` 字段的值为 ‘sunbin’，`grade` 字段的值为 55，`id` 字段不指定值，SQL 语句如下所示：

```
INSERT INTO student(grade,name)
VALUES (55,'sunbin');
```

执行 `INSERT` 语句向 `student` 表中添加数据，然后通过查询语句查看数据是否成功添加，执行结果如下

所示:

```
mysql> SELECT * FROM student;
+-----+-----+-----+
| id    | name    | grade |
+-----+-----+-----+
| 1     | zhangsan | 98.5  |
| 2     | lisi     | 95    |
| 3     | wangwu   | 61.5  |
| 4     | zhaoliu  | NULL  |
| NULL  | sunbin   | 55    |
+-----+-----+-----+
5 rows in set (0.00 sec)
```

从查询结果可以看出，新记录添加成功。



多学一招：INSERT 语句其他写法

INSERT 语句还有一种语法格式，可以为表中指定的字段或者全部字段添加数据，其格式如下所示：

```
INSERT INTO 表名
```

```
SET 字段名 1=值 1[, 字段名 2=值 2, .....]
```

在上面的语法格式中，“字段名 1”、“字段名 2”是指需要添加数据的字段名称，“值 1”、“值 2”表示添加的数据。如果在 SET 关键字后面指定了多个“字段名=值”对，每对之间使用逗号分隔，最后一个“字段名=值”对之后不需要逗号。接下来通过一个案例来演示使用这种语法格式向 student 表中添加记录。

【例 3-7】向 student 表中添加一条新记录，该条记录中 id 字段的值为 5，name 字段的值为 ‘boya’，grade 字段的值为 99，，INSERT 语句如下所示：

```
INSERT INTO student
```

```
SET id=5,name='boya',grade=99;
```

执行结果如下所示：

```
mysql> INSERT INTO student
-> SET id=5,name='boya',grade=99;
Query OK, 1 row affected (0.00 sec)
```

从执行结果可以看到 INSERT 语句成功执行，接下来通过查询语句查看数据是否成功添加，执行结果如下所示：

```
mysql> SELECT * FROM student;
+-----+-----+-----+
| id    | name    | grade |
+-----+-----+-----+
| 1     | zhangsan | 98.5  |
| 2     | lisi     | 95    |
| 3     | wangwu   | 61.5  |
| 4     | zhaoliu  | NULL  |
| NULL  | sunbin   | 55    |
| 5     | boya     | 99    |
+-----+-----+-----+
6 rows in set (0.00 sec)
```

从查询结果可以看出，student 表中新记录添加成功。

3.1.3 同时添加多条记录

有时候，需要一次向表中添加多条记录，当然，我们可以使用上面学习的两种方式将记录逐条添加，但是这样做需要书写多条 INSERT 语句，比较麻烦。其实，在 MySQL 中提供了使用一条 INSERT 语句同时添加多条记录的功能，其语法格式如下所示：

```
INSERT INTO 表名[(字段名 1, 字段名 2, .....)]
VALUES(值 1, 值 2, .....), (值 1, 值 2, .....),
... ..
(值 1, 值 2, .....);
```

在上述语法格式中，“(字段名 1, 字段名 2,)”是可选的，用于指定插入的字段名。“(值 1, 值 2,)”表示要插入的记录，该记录可以有多条，并且每条记录之间用逗号隔开。

【例 3-8】向 student 表中添加三条新记录，INSERT 语句如下所示：

```
INSERT INTO student VALUES
(6, 'lilei', 99),
(7, 'hanmeimei', 100),
(8, 'poly', 40.5);
```

执行结果如下所示：

```
mysql> INSERT INTO student VALUES
-> (6, 'lilei', 99),
-> (7, 'hanmeimei', 100),
-> (8, 'poly', 40.5);
Query OK, 3 rows affected (0.00 sec)
Records: 3 Duplicates: 0 Warnings: 0
```

从执行结果可以看出，INSERT 语句成功执行。其中“Records: 3”表示添加三条记录，“Duplicates: 0”表示添加的三条记录没有重复，“Warning: 0”表示添加记录时没有警告。在添加多条记录时，可以不指定字段列表，只需要保证 VALUES 后面跟随的值列表依照字段在表中定义的顺序即可。接下来通过查询语句查看数据是否成功添加，执行结果如下所示：

```
mysql> SELECT * FROM student;
+-----+-----+-----+
| id    | name      | grade |
+-----+-----+-----+
| 1     | zhangsan  | 98.5  |
| 2     | lisi      | 95    |
| 3     | wangwu    | 61.5  |
| 4     | zhaoliu   | NULL  |
| NULL  | sunbin    | 55    |
| 5     | boya      | 99    |
| 6     | lilei     | 99    |
| 7     | hanmeimei | 100   |
| 8     | poly      | 40.5  |
+-----+-----+-----+
```

```
8 rows in set (0.00 sec)
```

从查询结果可以看到，student 表中添加了三条新的记录。

和添加单条记录一样，如果不指定字段名，必须为每个字段添加数据，如果指定了字段名，就只需要为指定的字段添加数据。

【例 3-9】向 student 表中添加三条新记录，记录中只为 id 和 name 字段添加值，INSERT 语句如下所示：

```
INSERT INTO student(id,name) VALUES
(9,'liubei'),(10,'guanyu'),(11,'zhangfei');
```

执行 INSERT 语句向 student 表中添加数据，然后通过查询语句查看数据是否成功添加，执行结果如下所示：

```
mysql> SELECT * FROM student
-> WHERE id>8;

+-----+-----+-----+
| id    | name    | grade |
+-----+-----+-----+
| 9     | liubei  | NULL  |
| 10    | guanyu  | NULL  |
| 11    | zhangfei| NULL  |
+-----+-----+-----+
3 rows in set (0.01 sec)
```

通过查询结果可以看出，student 表中添加了 3 条新的记录，由于 INSERT 语句中没有为 grade 字段添加值，系统自动为其添加默认值 NULL。需要注意的是，由于 student 表中存在多条记录，都查询出来不便于观察，因此在查询语句中使用了 WHERE 子句来指定查询条件，WHERE id>8 限定了只查询 student 表中 id 值大于 8 的记录。

3.2 更新数据

更新数据是指对表中存在的记录进行修改，这是数据库常见的操作，比如某个学生改了名字，就需要对其记录信息中的 name 字段值进行修改。MySQL 中使用 UPDATE 语句来更新表中的记录，其基本的语法格式如下所示：

```
UPDATE 表名
SET 字段名 1 = 值 1[, 字段名 2 = 值 2, .....]
[WHERE 条件表达式]
```

在上述语法格式中，“字段名 1”，“字段名 2”用于指定要更新的字段名称，“值 1”，“值 2”用于表示字段更新的新数据。“WHERE 条件表达式”是可选的，用于指定更新数据需要满足的条件。UPDATE 语句可以更新表中的部分数据和全部数据，下面就对这两种情况进行讲解：

1、UPDATE 更新部分数据

更新部分数据是指根据指定条件更新表中的某一条或者某几条记录，需要使用 WHERE 子句来指定更新记录的条件。

【例 3-10】更新 student 表中 id 字段值为 1 的记录，将记录中的 name 字段的值更改为‘caocao’，grade 字段的值更改为 50。在更新数据之前，首先使用查询语句查看 id 字段值为 1 的记录，执行结果如下所示：

```
mysql> SELECT * FROM student
```

```
-> WHERE id=1;

+-----+-----+-----+
| id    | name      | grade |
+-----+-----+-----+
| 1     | zhangsan  | 98.5  |
+-----+-----+-----+

1 row in set (0.00 sec)
```

从查询结果可以看到，id 字段值为 1 的记录只有一条，记录中 name 字段的值为 ‘zhangsan’，grade 字段的值为 98.5。下面使用 UPDATE 语句更新这条记录，SQL 语句如下所示：

```
UPDATE student
set name='caocao',grade=50
WHERE id=1;
```

上述 SQL 语句执行成功后，会将 id 为 1 的数据进行更新。为了验证数据是否更新成功，使用 SELECT 语句查看数据库 student 中 id 为 1 的记录，查询结果如下所示：

```
mysql> SELECT * FROM student
      -> WHERE id=1;

+-----+-----+-----+
| id    | name      | grade |
+-----+-----+-----+
| 1     | caocao    | 50     |
+-----+-----+-----+

1 row in set (0.00 sec)
```

从查询结果可以看到，id 字段值为 1 的记录发生了更新，记录中 name 字段的值变为 ‘caocao’，grade 字段的值变为 50。

如果表中有多条记录满足 WHERE 子句中的条件表达式，则满足条件的记录都会发生更新。

【例 3-11】更新 student 表中 id 字段值小于 4 的记录，将这些记录的 grade 字段值都更新为 100。在更新数据前，首先使用查询语句查看 id 字段值小 4 的记录，执行结果如下所示：

```
mysql> SELECT * FROM student
      -> WHERE id<4;

+-----+-----+-----+
| id    | name      | grade |
+-----+-----+-----+
| 1     | caocao    | 50     |
| 2     | lisi      | 95     |
| 3     | wangwu    | 61.5   |
+-----+-----+-----+

3 rows in set (0.00 sec)
```

从查看结果可以看到，id 字段值小于 4 的记录一共有 3 条，它们的 grade 字段值各不相同。下面使用 UPDATE 语句更新这 3 条记录，UPDATE 语句如下所示：

```
UPDATE student
SET grade=100
WHERE id<4;
```

执行 UPDATE 语句更新 student 表中的数据，然后通过查询语句查看更新后的数据，执行结果如下所示：


```
mysql> SELECT * FROM student
      -> WHERE id<4;

+-----+-----+-----+
| id    | name  | grade |
+-----+-----+-----+
| 1     | caocao | 100   |
| 2     | lisi   | 100   |
| 3     | wangwu | 100   |
+-----+-----+-----+
3 rows in set (0.00 sec)
```

从查询结果可以看出，id 字段值为 1、2、3 的记录其 grade 字段值都变为 100，这说明满足 WHERE 子句中条件表达式的记录都更新成功。

2、UPDATE 更新全部数据

在 UPDATE 语句中如果没有使用 WHERE 子句，则会将表中所有记录的指定字段都进行更新。

【例 3-12】更新 student 表中全部 11 条记录，将 grade 字段值都更新为 80，UPDATE 语句如下所示：

```
UPDATE student
SET grade=80;
```

执行 UPDATE 语句更新 student 表中的数据，接下来通过查询语句查看更新后的记录，SQL 语句如下所示：

```
mysql> select * from student;

+-----+-----+-----+
| id    | name      | grade |
+-----+-----+-----+
| 1     | caocao    | 80    |
| 2     | lisi      | 80    |
| 3     | wangwu    | 80    |
| 4     | zhaoliu   | 80    |
| NULL  | sunbin     | 80    |
| 5     | boya      | 80    |
| 6     | lilei     | 80    |
| 7     | hanmeimei | 80    |
| 8     | poly      | 80    |
| 9     | liubei    | 80    |
| 10    | guanyu    | 80    |
| 11    | zhangfei  | 80    |
+-----+-----+-----+
11 rows in set (0.00 sec)
```

从查询结果可以看出，student 表中所有记录的 grade 字段都变为 80，数据更新成功。

3.3 删除数据

删除数据是指对表中存在的记录进行删除，这是数据库的常见操作，比如一个学生转学了，就需要在

`student` 表中将其信息记录删除。MySQL 中使用 `DELETE` 语句来删除表中的记录，其语法格式如下所示：

```
DELETE FROM 表名 [WHERE 条件表达式]
```

在上面的语法格式中，“表名”指定要执行删除操作的表，`WHERE` 子句为可选参数，用于指定删除的条件，满足条件的记录会被删除。`DELETE` 语句可以删除表中的部分数据和全部数据，下面就对这两种情况进行讲解：

1、DELETE 删除部分数据

删除部分数据是指根据指定条件删除表中的某一条或者某几条记录，需要使用 `WHERE` 子句来指定删除记录的条件。

【例 3-13】在 `student` 表中，删除 `id` 字段值为 11 的记录，在删除数据之前，首先使用查询语句查看 `id` 字段值为 11 的记录，执行结果如下所示：

```
mysql> SELECT * FROM student
-> WHERE id=11;

+-----+-----+-----+
| id    | name    | grade |
+-----+-----+-----+
| 11    | zhangfei | 80    |
+-----+-----+-----+

1 row in set (0.02 sec)
```

从查询结果可以看到，`student` 表中有一条 `id` 字段值为 11 的记录，下面使用 `DELETE` 语句删除这条记录，`DELETE` 语句如下所示：

```
DELETE FROM student
WHERE id=11;
```

执行结果如下所示：

```
mysql> DELETE FROM student
-> WHERE id=11;

Query OK, 1 row affected (0.00 sec)
```

从执行结果可以看出，`DELETE` 语句成功执行，接下来再次通过查询语句查看 `id` 字段为 11 的记录，执行结果如下所示：

```
mysql> SELECT * FROM student
-> WHERE id=11;

Empty set (0.00 sec)
```

从查询结果可以看到记录为空，说明 `id` 字段为 11 的记录被成功删除。

在执行删除操作的表中，如果有多条记录满足 `WHERE` 子句中的条件表达式，则满足条件的记录都会被删除。

【例 3-14】在 `student` 表中，删除 `id` 字段值大于 5 的所有记录，在删除数据之前，首先使用查询语句查看 `id` 字段值大于 5 的所有记录，执行结果如下所示：

```
mysql> SELECT * FROM student
-> WHERE id>5;

+-----+-----+-----+
| id    | name    | grade |
+-----+-----+-----+
| 6     | lilei    | 80    |
| 7     | hanmeimei | 80    |
+-----+-----+-----+
```

```

| 8 | poly | 80 |
| 9 | liubei | 80 |
| 10 | guanyu | 80 |
+-----+-----+-----+
5 rows in set (0.00 sec)

```

从查询结果可以看到，student 表中 id 字段值大于 5 的记录有 5 条，下面使用 DELETE 语句删除满足条件的这 5 条记录，DELETE 语句如下所示：

```

DELETE FROM student
WHERE id>5;

```

执行 DELETE 语句删除 student 表中的数据，然后再次通过查询语句查看 id 字段值大于 5 的记录，执行结果如下所示：

```

mysql> SELECT * FROM student
      -> WHERE id>5;
Empty set (0.00 sec)

```

从查询结果可以看到记录为空，说明 id 字段置大于 5 的记录被成功删除了。

2、DELETE 删除全部数据

在 DELETE 语句中如果没有使用 WHERE 子句，则会将表中的所有记录都删除。

【例 3-15】删除 student 表中的所有记录，在删除数据之前首先使用查询语句查看 student 表中的所有记录，执行结果如下所示：

```

mysql> SELECT * FROM student;
+-----+-----+-----+
| id  | name  | grade |
+-----+-----+-----+
| 1  | caocao | 80  |
| 2  | lisi  | 80  |
| 3  | wangwu | 80  |
| 4  | zhaoliu | 80  |
| NULL | sunbin | 80  |
| 5  | boya  | 80  |
+-----+-----+-----+
6 rows in set (0.00 sec)

```

从查询结果可以看出，student 表中还有 6 条记录，下面使用 DELETE 语句将这 6 条记录全部删除，DELETE 语句如下所示：

```

DELETE FROM student;

```

执行 DELETE 语句删除 student 表中的数据，然后再次通过查询语句查看 student 表中的记录，执行结果如下所示：

```

mysql> SELECT * FROM student;
Empty set (0.00 sec)

```

从查询结果可以看到记录为空，说明表中所有的记录成功删除。



多学一招：使用关键字 TRUNCATE 删除表中数据

在 MySQL 数据库中，还有一种方式可以用来删除表中所有的记录，这种方式需要用到一个关键字 TRUNCATE，其语法格式如下：

TRUNCATE [TABLE] 表名

TRUNCATE 的语法格式很简单，只需要通过“表名”指定要执行删除操作的表即可。下面通过一个案例来演示 TRUNCATE 的用法。

【例 3-16】在数据库 chapter03 中创建一张表 tab_truncate，创建 tab_truncate 表的 SQL 语句如下所示：

```
CREATE TABLE tab_truncate(
    id INT(3) PRIMARY KEY AUTO_INCREMENT,
    name VARCHAR(4)
);
```

在创建的 tab_truncate 表中，id 字段值设置了 AUTO_INCREMENT，在每次添加记录时系统会为该字段自动添加值，id 字段的默认初始值是 1，每添加一条记录，该字段值会自动加 1。接下来向 tab_truncate 表中添加 5 条记录，且只添加 name 字段的值，SQL 语句如下所示：

```
INSERT INTO tab_truncate(name)
VALUES('A'),('B'),('C'),('D'),('E');
```

执行 INSERT 语句向 tab_truncate 表中添加 5 条记录，接下来通过查询语句查看数据是否成功添加，执行结果如下所示：

```
mysql> SELECT * FROM tab_truncate;
+----+-----+
| id | name |
+----+-----+
| 1  | A    |
| 2  | B    |
| 3  | C    |
| 4  | D    |
| 5  | E    |
+----+-----+
5 rows in set (0.00 sec)
```

从查询结果可以看出，tab_truncate 表中添加了 5 条记录，且系统自动为每条记录的 id 字段添加了值。接下来使用 TRUNCATE 语句删除 tab_truncate 表中的所有记录，TRUNCATE 语句如下所示：

```
TRUNCATE TABLE tab_truncate;
```

执行结果如下所示：

```
mysql> TRUNCATE TABLE tab_truncate;
Query OK, 0 rows affected (0.02 sec)
```

从执行结果可以看到 TRUNCATE 语句成功执行，接下来通过查询语句查看 tab_truncate 表中的记录是否删除成功，执行语句如下所示：

```
mysql> SELECT * FROM tab_truncate;
Empty set (0.00 sec)
```

通过查询结果可以看到记录为空，说明 tab_表中的记录被全部删除了。

TRUNCATE 语句和 DELETE 语句都能实现删除表中的所有数据的功能，但两者也有一定的区别，下面就针对两者的区别进行说明：

- 1、DELETE 语句是 DML 语句，TRUNCATE 语句通常被认为是 DDL 语句。
- 2、DELETE 语句后面可以跟 WHERE 子句，通过指定 WHERE 子句中的条件表达式只删除满足条件的部分记录，而 TRUNCATE 语句只能用于删除表中的所有记录
- 3、使用 TRUNCATE 语句删除表中的数据后，再次向表中添加记录时，自动增加字段的默认初始值重新由 1 开始，而使用 DELETE 语句删除表中所有记录后，再次向表中添加记录时，自动增加字段的值为删除

除时该字段的最大值加 1。

【例 3-17】在空表 tab_truncate 中，重新添加 5 条记录，SQL 语句如下：

```
INSERT INTO tab_truncate(name)
VALUES('F'),('G'),('H'),('I'),('J');
```

执行 INSERT 语句向 tab_truncate 表中添加 5 条记录，使用查询语句查看表中的记录，执行结果如下所示：

```
mysql> SELECT * FROM tab_truncate;
+----+-----+
| id | name |
+----+-----+
| 1  | F    |
| 2  | G    |
| 3  | H    |
| 4  | I    |
| 5  | J    |
+----+-----+
5 rows in set (0.00 sec)
```

从查询结果可以看出，系统为 tab_truncate 表中 id 字段默认添加了值，初始值从 1 开始。接下来使用 DELETE 语句删除 tab_truncate 表中的所有记录，DELETE 语句如下所示：

```
DELETE FROM tab_truncate;
```

执行 DELETE 语句删除 tab_truncate 表中全部记录，然后向表中添加一条新记录，SQL 语句如下所示：

```
INSERT INTO tab_truncate(name) VALUES('K');
```

执行 INSERT 语句向 tab_truncate 表中添加一条记录，再次使用查询语句查看表中的记录，SQL 语句如下所示：

```
mysql> SELECT * FROM tab_truncate;
+----+-----+
| id | name |
+----+-----+
| 6  | K    |
+----+-----+
1 row in set (0.00 sec)
```

从查询结果可以看到，新添加记录的 id 字段为 6，这是因为在使用 DELETE 语句删除的 5 条记录中，id 字段的最大值为 5，因此再次添加记录时，新记录的 id 字段值就为 5+1。

4、使用 DELETE 语句时，每删除一条记录都会在日志中记录，而使用 TRUNCATE 语句时，不会在日志中记录删除的内容，因此 TRUNCATE 语句的执行效率比 DELETE 语句高。

3.4 本章小结

本章主要讲解了添加、更新和删除表中数据的基本操作，这些内容都是本章的重点，也是数据库开发最基础的操作。大家在学习时一定要多加练习，在实际操作中掌握本章的内容，为以后的数据库操作学习和数据库开发奠定坚实的基础。