【互联网 + 教育】新形态一体化系列教材

MySQL 数据库技术应用与实战

主　编　张　景　李世华　王　欢　张　勇

副主编　姚学峰　刘文林　邓　茜　谭告成

龙　滔　杨　倩

参　编　王天翼

上海交通大学出版社

SHANGHAI JIAO TONG UNIVERSITY PRESS

内容提要

本书深入浅出地介绍了MySQL数据库技术的发展和应用，涉及MySQL基础知识、MySQL数据库操作、MySQL数据库优化等几个部分。每章的内容充实、循序渐进。在讲解过程中，穿插了大量的实例和任务，帮助读者理解和掌握各个知识；每章结尾都配有相关习题，有助于读者自测和练习。

本书共12章，包括数据库设计概述、MySQL环境配置和创建数据库、MySQL表结构的管理、表记录的更新操作、表记录的检索、MySQL运算符和系统函数、视图和触发器、存储过程和存储函数、MySQL分区、MySQL查询优化、索引优化和数据库优化以及用户信息管理系统。

本书可以作为各类院校计算机科学与技术、网络工程、电子信息等相关专业“MySQL数据库设计”课程的教材，也可作为从事Web程序设计相关工作的技术人员自学参考用书。

图书在版编目（CIP）数据

MySQL数据库技术应用与实战 / 张景等主编. — 上海 : 上海交通大学出版社, 2022.8（2025.2 重印）
ISBN 978-7-313-26998-0

Ⅰ. ①M… Ⅱ. ①张… Ⅲ. ①SQL语言—数据库管理系统 Ⅳ. ①TP311.132.3

中国版本图书馆CIP数据核字（2022）第109093号

MySQL数据库技术应用与实战
MYSQL SHUJUKU JISHU YINGYONG YU SHIZHAN

主　　编：张　景　李世华　王　欢　张　勇
出版发行：上海交通大学出版社　　地　　址：上海市番禺路951号
邮政编码：200030　　电　　话：021-64071208
印　　刷：三河市宏图印务有限公司　　经　　销：全国新华书店
开　　本：787mm × 1092mm　1/16　　印　　张：19.25
字　　数：492千字
版　　次：2022年8月第1版　　印　　次：2025年2月第2次印刷
书　　号：ISBN 978-7-313-26998-0
定　　价：58.00元

前　言

党的二十大报告指出：“推动战略性新兴产业融合集群发展，构建新一代信息技术、人工智能、生物技术、新能源、新材料、高端装备、绿色环保等一批新的增长引擎。”随着计算机的不断发展，需要计算机处理的数据越来越多，能够高效存储和处理的数据库技术应运而生。数据库技术已经成为IT领域不可缺少的一部分。数据库技术的设计和开发已经成为各类应用和网站中主要的组成部分，其需求也相应增多。因此数据库设计和维护技术作为一项知识技能受到大家越来越多的重视。在众多数据库中，MySQL数据库作为访问数据库常用的SQL语言，具备体积小、速度快、总体拥有成本低、开放源码等特点。所以，MySQL成为一般中小型网站开发的首选数据库。

本书以MySQL数据库为基础，围绕最新的MySQL数据库技术展开深入讲解，以清晰的思路、典型的实例使读者快速入门，并逐步掌握网络编程的知识。本书注重基础理论与实用开发相结合，突出数据库设计思想与数据库操作优化方法的介绍，所选实例都具有较强的概括性和实际应用价值。

本书是编者根据多年从事数据库程序设计工作和讲授计算机专业相关课程的教学实践，在已编多部讲义和教材的基础上编写而成的；内容充实，循序渐进，选材上注重系统性、先进性和实用性；注重实践性，精选大量例题，增加了代码注释且所有例题已在MySQL数据库上调试通过，可直接引用，读者也可按照书中提示步骤自己动手完成。

由于编者水平有限，加之编写时间仓促，书中难免存在错误和疏漏之处，希望广大读者批评指正。

编　者

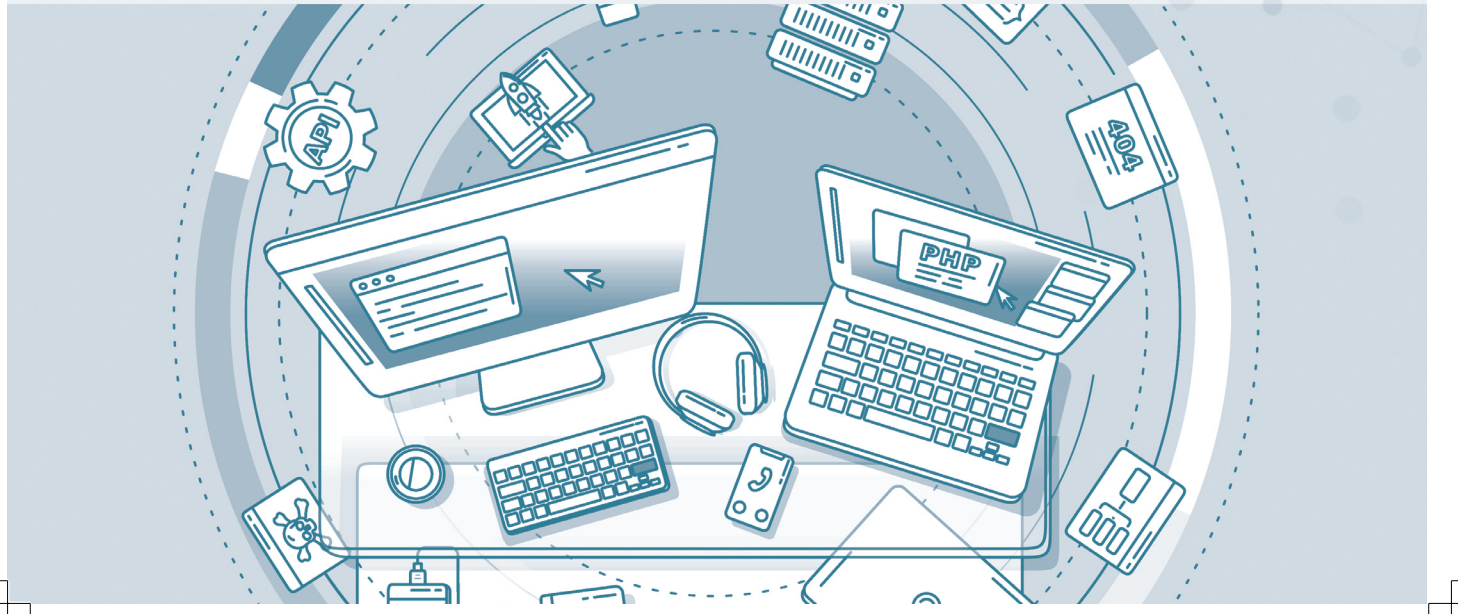
目 录

：表示所在页码包含微课视频。

第 4 章

表记录的更新操作

表记录的更新操作主要包括对数据的插入、更新与删除 3 种方式，它们分别需要使用不同的语句来实现。使用 INSERT 语句实现插入数据，使用 UPDATE 语句实现更新数据，使用 DELETE 语句删除数据。



1）更新数据的意义

更新数据包括对数据表中数据的查找、添加、删除以及替换操作。更新数据的意义是为了让数据库的信息时刻保持最新，从而为用户管理产品提供更加准确的信息依据。

表记录的更新操作

2）更新数据的原理

更新数据分为两步：第一步是使用选择语句查找并选中对应的数据内容；第二步是对数据进行插入（INSERT）、更新（UPDATE）和删除（DELETE）等操作。

3）数据插入规则

使用INSERT语句向MySQL数据表中插入数据记录规则如下所示。

（1）当没有指定需要插入数据的列或字段时，MySQL默认会向所有列或字段中插入数据，要求插入数据的值必须对应数据表中所有的列或字段一一对应。

（2）当指定插入数据的列或字段时，必须指定所有列或字段插入的对应数据。

（3）当未指定插入数据的列或字段时，插入值列表的顺序、个数、类型必须和数据表中定义字段相符。

（4）当指定插入数据的列或字段时，只需要保证插入的值列表的顺序与列或字段的列表顺序一致即可。

（5）当指定插入数据的列或字段时，既可以向数据表中插入完整的行记录，也可以只向部分字段中插入数据，省略的部分由字段的默认值填充。

4）删除数据

删除指定数据时一定要确认对应数据不再使用，并且要确认删除的数据的准确性，避免因为操作失误导致数据删除错误，从而丢失数据。

4.1 插入数据

数据插入

插入数据是指将新的数据行插入现有表中。插入数据的方式包括插入完整行记录、向指定字段插入数据、一次性插入多条数据以及将查询结果插入另一个表中。

4.1.1 插入完整的行记录

向MySQL数据表中插入完整的行记录时，可以选择是否指定插入数据的列或字段。不指

定插入的数据列或字段的语法格式如下所示。

```
INSERT INTO table_name
VALUES
(value1 [, value2, ... , valuen])
```

指定插入数据列或字段的语法格式如下所示。

```
INSERT INTO table_name
(column1 [, column2, ..., columnn])
VALUES
(value1 [,value2, ..., valuen])
```

其中，各属性如下所示。

（1）table_name：表示数据表的名称。

（2）column1 [, column2, ..., column]：表示列或字段列表，该项为可选项。

（3）value1 [, value2, ... , valuen] 数据值列表。

实例 4-1 向 shop1 数据表插入数据。

（1）查看 t_goods 数据表中的数据，效果如下所示。

```
mysql> SELECT * FROM shop1;
Empty set (0.00 sec)
```

结果显示数据为空。

（2）向 shop1 表中插入完整数据，效果如下所示。

```
mysql> INSERT INTO t_goods VALUES (1,'张三',18);
Query OK, 1 row affected (0.00 sec)
```

（3）再次查看数据表中的数据，效果如下所示。

```
mysql> SELECT * FROM shop1;
+----+--------+-----+
| id | name   | age |
+----+--------+-----+
|  1 | 张三   |  18 |
+----+--------+-----+
1 row in set (0.00 sec)
```

可以看到张三的信息被插入表中。

（4）以指定数据列的形式再次插入一条数据，效果如下所示。

```
mysql> INSERT INTO shop1 (id,name,age)   VALUES (2,'李四',19);
Query OK, 1 row affected (0.00 sec)
```

（5）查看 shop1 数据表中的数据，如下所示。

```
mysql> SELECT * FROM shop1;
+----+---------+-------+
| id | name    | age   |
+----+---------+-------+
|  1 | 张三    |  18   |
|  2 | 李四    |  19   |
+----+---------+-------+
2 rows in set (0.00 sec)
```

可以看到，shop1 数据表中成功插入了两条数据。

> 注意：指定需要插入数据的字段或列时，只需要保证值列表的顺序与字段列表的顺序一致即可。

4.1.2 向指定字段插入数据

指定字段插入数据

向指定字段插入数据是指只插入某个字段对应的数据，其他字段自动填充为默认值，其语法形式如下所示。

```
INSERT INTO table_name
(column1 [, column2, ..., columnn])
VALUES
(value1 [,value2, ..., valuen])
```

其中，column 为要指定的字段，value 为对应的数据。

实例 4-2 向数据表插入指定字段和数据。

（1）对数据表 shop1 插入一行数据，数据只包含 name 字段对应值王五，效果如下所示。

```
mysql> INSERT INTO shop1 (name) VALUES ('王五');
Query OK, 1 row affected (0.01 sec)
```

（2）插入成功后，查看数据表 shop1，效果如下所示。

```
mysql> SELECT * FROM shop1;
+------+------------+--------+
| id   | name       | age    |
+------+------------+--------+
|    1 | 张三       |   18   |
|    2 | 李四       |   19   |
|    0 | 王五       |    0   |
+------+------------+--------+
3 rows in set (0.00 sec)
```

从运行结果可以看出，最后一行数据中 name 字段的值为王五，其他字段的值为默认值 0。


一次插入多条数据记录

4.1.3 一次插入多条数据记录

如果需要将多条数据一次性插入指定表，则可以使用INSERT语句来实现，其语法形式如下所示。

```
INSERT INTO table_name
VALUES
(value1 [,value2, ..., valuen]),
(value1 [,value2, ..., valuen]),
......
(value1 [,value2, ..., valuen])
```

或

```
INSERT INTO table_name
(column1 [, column2, ..., columnn])
VALUES
(value1 [,value2, ..., valuen]),
(value1 [,value2, ..., valuen]),
......
(value1 [,value2, ..., valuen])
```

其中，一次插入多条记录时，可以只指定需要插入数据，省略字段内容，但是插入的内容的类型要符合对应的字段类型。

实例 4-3 向指定的数据表插入多行数据。

（1）向数据表shop1插入多条数据记录，效果如下所示。

```
mysql> INSERT INTO shop1 VALUES
-> (3,'周琦',19),
-> (4,'刘柳',20);
Query OK, 2 rows affected (0.01 sec)
Records: 2  Duplicates: 0  Warnings: 0
```

（2）查看shop1数据表中的数据，效果如下所示。

```
mysql> SELECT * FROM shop1;
+----+----------+--------+
| id | name     | age    |
+----+----------+--------+
|  0 | 王五     |    0   |
|  1 | 张三     |   18   |
|  2 | 李四     |   19   |
|  3 | 周琦     |   19   |
|  4 | 刘柳     |   20   |
+----+----------+--------+
5 rows in set (0.00 sec)
```

从运行结果可以看出，shop1 表的最下方添加了两条新的数据。如果需要指定字段并插入多条数据，此时要注意数据顺序要与指定字段的数据相同，并且数据的类型要与字段类型相符。

（3）以指定字段的方式插入多条数据的效果如下所示。

```
mysql> INSERT INTO shop1  (id,name,age)  VALUES (5,'周
武',19),(6,'郑旺',20);
Query OK, 2 rows affected (0.01 sec)
Records: 2  Duplicates: 0  Warnings: 0
```

（4）查看 shop1 数据表，效果如下所示。

```
mysql> SELECT * FROM shop1;
+----+----------+--------+
| id | name     | age    |
+----+----------+--------+
|  0 | 王五     |    0   |
|  1 | 张三     |   18   |
|  2 | 李四     |   19   |
|  3 | 周琦     |   19   |
|  4 | 刘柳     |   20   |
|  5 | 周武     |   19   |
|  6 | 郑旺     |   20   |
+----+----------+--------+
7 rows in set (0.00 sec)
```

从运行结果可以看到，数据插入成功。

4.1.4 将查询结果插入另一个表中

将查询结果插入另一个表中

INSERT 语句配合查询语句可以将查询的结果批量插入指定的数据表中，其语法形式如下所示。

```
INSERT INTO target_table
(tar_column1 [, tar_column2, ..., tar_columnn])
SELECT
(src_column1 [, src_column2, ..., src_columnn])
FROM source_table
[WHERE condition]
```

其中，各属性功能如下所示。

（1）target_table：需要插入数据的目标表。

（2）tar_column1 [, tar_column2, ..., tar_columnn]：需要插入数据的目标表中的字段列表。

（3）source_table：使用 SELECT 语句要查询的数据表，也就是源数据表。

（4）src_column1 [, src_column2, ..., src_columnn]：指定被查询源数据表中的字段。

（5）condition：使用SELECT语句查询数据的条件限制，该项为可选项。

实例 4-4 查询指定表的数据并将查询结果插入另外的数据表中。

（1）查看要插入数据的数据表shop2 的数据内容，效果如下所示。

```
mysql> SELECT * FROM shop2;
Empty set (0.00 sec)
```

可以看出，数据表shop2 中没有数据。

（2）通过数据表shop1 中id值为 3 的数据，并插入数据表shop2 中，效果如下所示。

```
mysql> INSERT INTO shop2
    -> (id,name,age)
    -> SELECT
    -> id, name,age
    -> FROM shop1
    -> WHERE id = 3;
Query OK, 1 row affected (0.01 sec)
Records: 1  Duplicates: 0  Warnings: 0
```

可以看出，查询并插入数据成功。

（3）查询数据表shop2 中的数据，效果如下所示。

```
mysql> SELECT * FROM shop2;
+----+----------+-------+
| id | name     | age   |
+----+----------+-------+
|  3 | 周琦     |  19   |
+----+----------+-------+
1 row in set (0.00 sec)
```

从运行结果可以看出，id值为 3 的数据被成功插入表中。由于数据表shop1 和数据表shop2 的字段数量和类型相同，所以在查询并插入数据时可以省略指定字段的语句，使用星号（*）代替，表示查询所有表中所有的数据。

（4）当不指定字段名时将id值为 5 的数据插入数据表shop2 中，效果如下所示。

```
mysql> INSERT INTO shop2
    -> SELECT *
    -> FROM shop1
    -> WHERE id = 5;
Query OK, 1 row affected (0.01 sec)
Records: 1  Duplicates: 0  Warnings: 0
```

（5）查看数据表shop2 的数据，效果如下所示。

```
mysql> SELECT * FROM shop2;
+----+--------+-----+
| id | name   | age |
```

```
+----+--------+-----+
|  3 | 周琦   |  19 |
|  5 | 周武   |  19 |
+----+--------+-----+
2 rows in set (0.00 sec)
```

从运行结果可以看出，id为5的数据行被插入数据表shop2中。

任务4-1 将“老师信息表”中的所有数据插入“员工信息表”中


将“老师信息表”中所有数据插入“员工信息表”中

任务描述

（1）为“老师信息表”添加数据。

（2）将“老师信息表”中的所有数据插入“员工信息表”中。

任务实施

1. 查询“老师信息表”的现有数据

首先选择bookshop数据库，然后查询“老师信息表”的数据内容，效果如下所示。

```
mysql> USE bookshop;
Database changed
mysql> SELECT * FROM 老师信息表;
Empty set (0.00 sec)
```

从运行结果可以看出，“老师信息表”的数据为空。

2. 为“老师信息表”添加数据

向“老师信息表”中添加多条数据内容，效果如下所示。

```
mysql> INSERT INTO 老师信息表 VALUES
    -> (1,'章小红',25),
    -> (2,'李时光',43),
    -> (3,'王胜利',40);
Query OK, 3 rows affected (0.01 sec)
Records: 3  Duplicates: 0  Warnings: 0
```

3. 再次查询“老师信息表”的数据

通过查询语句再次查询“老师信息表”中的数据，确认插入数据成功，效果如下所示。

```
mysql> SELECT * FROM 老师信息表;
+------------------+----------------+----------+
| 老师编号         | 姓名           | 年龄     |
+------------------+----------------+----------+
|              1   | 章小红         |      25  |
|              2   | 李时光         |      43  |
|              3   | 王胜利         |      40  |
+------------------+----------------+----------+
3 rows in set (0.00 sec)
```

从运行结果可以看出，成功插入了 3 条数据。

4. 查询“员工信息表”的现有数据

通过查询语句查看“员工信息表”的数据，效果如下所示。

```
mysql> SELECT * FROM 员工信息表;
Empty set (0.00 sec)
```

5. 将“老师信息表”中的数据插入“员工信息表”中

通过查询和添加语句将“老师信息表”的所有数据添加到“员工信息表”中，效果如下所示。

```
mysql> INSERT INTO 员工信息表
    -> SELECT *
    -> FROM 老师信息表;
Query OK, 3 rows affected (0.00 sec)
Records: 3  Duplicates: 0  Warnings: 0
```

再次查看“员工信息表”中的数据，效果如下所示。

```
mysql> SELECT * FROM 员工信息表;
+------------------+---------------+-----------+
| 员工编号         | 姓名          | 年龄      |
+------------------+---------------+-----------+
|            1     | 章小红        |     25    |
|            2     | 李时光        |     43    |
|            3     | 王胜利        |     40    |
+------------------+---------------+-----------+
3 rows in set (0.00 sec)
```

从运行结果可以看出，“老师信息表”中的数据被成功插入“员工信息表”中。

数据更新

4.2 更新数据

当现实的数据发生改变后，需要及时对数据库中的数据进行更新。MySQL 使用 UPDATE 语句可以实现对数据表中的数据进行更新操作。更新时既可以更新全部数据，也可以根据条件更新指定数据。更新数据的语法形式如下所示。

```
UPDATE table_name
SET column1=value1, column2=value2, ... , columnn=valuen
[WHERE condition]
```

其中，各属性功能如下所示。

（1）table_name：需要更新数据的表名称。

（2）column1，column2, ... ,columnn：需要更新的字段的名称。

（3）value1, value2, ... , valuen：字段的更新值。

（4）condition：更新的记录需要满足的条件限制，该项为可选项。

4.2.1 更新数据表中的所有数据

更新数据表中的所有数据需要将更新语法中的WHERE条件进行省略，然后指定要更新的字段，以及更新的值，就可以将所有数据行对应字段的值更新为指定值。

实例 4-5 将指定字段的所有数据进行更新。

（1）将shop1数据表中的age字段的所有数据更新为18，效果如下所示。

```
mysql> UPDATE shop1 SET age = 18;
Query OK, 6 rows affected (0.01 sec)
Rows matched: 7  Changed: 6  Warnings: 0
```

（2）从运行结果可以看出更新数据成功，然后通过查询语句查看数据表shop1中的数据，效果如下所示。

```
mysql> SELECT * FROM shop1;
+----+-----------+-------+
| id | name      | age   |
+----+-----------+-------+
|  0 | 王五      |    18 |
|  1 | 张三      |    18 |
|  2 | 李四      |    18 |
|  3 | 周琦      |    18 |
|  4 | 刘柳      |    18 |
|  5 | 周武      |    18 |
|  6 | 郑旺      |    18 |
+----+-----------+-------+
7 rows in set (0.00 sec)
```

从运行结果可以看出，字段age中的所有数据都变为了18。

4.2.2 更新表中特定的数据行

更新表中特定的数据行

如果只想更新数据表中的某一行数据，就可以使用WHERE条件语句指定条件，找出对应的数据行，然后进行更新数据的操作。

实例 4-6 通过条件语句更新指定行的数据。

（1）将数据表shop1的name值为周琦的age值修改为30，效果如下所示。

```
mysql> UPDATE shop1 SET age = 30 WHERE name = '周琦';
Query OK, 1 row affected (0.01 sec)
Rows matched: 1  Changed: 1  Warnings: 0
```

（2）从运行结果可以看出修改了一个值。使用查询语句查看更新后的数据表shop1的数据，效果如下所示。

```
mysql> SELECT * FROM shop1;
+----+-----------+--------+
| id | name      | age    |
+----+-----------+--------+
|  0 | 王五      |   18   |
|  1 | 张三      |   18   |
|  2 | 李四      |   18   |
|  3 | 周琦      |   30   |
|  4 | 刘柳      |   18   |
|  5 | 周武      |   18   |
|  6 | 郑旺      |   18   |
+----+-----------+--------+
7 rows in set (0.00 sec)
```

从运行结果可以看出，name值为周琦的数据行中，age的值更新为30。


更新某个范围内的数据

4.2.3 更新某个范围内的数据

如果要一次性更新多条数据，可以使用BETWEEN ... AND语句、比较运算符、LIKE、IN以及NOT IN等语句实现。

1. BETWEEN ... AND语句

BETWEEN ... AND语句为WHERE查询语句的子语句，它的作用是指定字段对应数据的范围，其语法形式如下所示。

```
BETWEEN  起始值  AND  结束值
```

其中，起始值为查询范围的最小值，结束值为查询范围的最大值。选择的数据包括与起始值和结束值相等的数据行。

实例 4-7 更新指定范围内的数据。

（1）将数据表shop1中id值为3到6的数据行的age字段的值更新为40，效果如下所示。

```
mysql> UPDATE shop1 SET age = 40 WHERE id BETWEEN 3 AND 6;
Query OK, 4 rows affected (0.01 sec)
Rows matched: 4  Changed: 4  Warnings: 0
```

（2）从运行结果可以看出更新了4条数据。使用查询语句确定更新是否正确，效果如下所示。

```
mysql> SELECT * FROM shop1;
+----+----------+--------+
| id | name     | age    |
+----+----------+--------+
|  0 | 王五     |   18   |
|  1 | 张三     |   18   |
|  2 | 李四     |   18   |
```

```
|  3 | 周琦       |   40   |
|  4 | 刘柳       |   40   |
|  5 | 周武       |   40   |
|  6 | 郑旺       |   40   |
+----+----------+--------+
7 rows in set (0.00 sec)
```

2. 比较运算符


比较运算符

比较运算符包括大于号(>)、大于等于号(>=)、小于号(<)、小于等于号(<=)、不等于号(!=)等，比较运算符使用时可搭配AND或OR关键字实现多个比较运算符的使用，在WHERE语句的字句中使用，其语法形式如下所示。

```
字段名 比较运算符 数据 AND/OR 字段名 比较运算符 数据
```

其中，如果只使用一个比较运算符查找数据，不需要添加AND或OR关键字及其后面的内容。

实例 4-8 使用比较运算符选择并更新多行内容的数据。

(1)将数据表shop1中id值为2到4的数据行的age字段的值更新为30，效果如下所示。

```
mysql> UPDATE shop1 SET age = 30 WHERE id >= 2 AND id <= 4;
Query OK, 3 rows affected (0.01 sec)
Rows matched: 3  Changed: 3  Warnings: 0
```

(2)从运行结果可以看出修改了3行数据，然后使用查询语句看出数据表shop1的数据，效果如下所示。

```
mysql> SELECT * FROM shop1;
+----+----------+--------+
| id | name     | age    |
+----+----------+--------+
|  0 | 王五       |   18   |
|  1 | 张三       |   18   |
|  2 | 李四       |   30   |
|  3 | 周琦       |   30   |
|  4 | 刘柳       |   30   |
|  5 | 周武       |   40   |
|  6 | 郑旺       |   40   |
+----+----------+--------+
7 rows in set (0.00 sec)
```

从运行结果可以看出，id值为2，3，4的数据行的age值更新为30。

3. LIKE语句


LIKE语句

LIKE语句是WHERE语句的子句，其功能是选择与指定条件相同的内容。它可以与百分号(%)配合使用。例如，LIKE %王，表示查找以王字为结尾的数据。

实例 4-9 使用LIKE语句更新多行数据。

（1）将数据表shop1中name值以“五”结尾的数据行的age字段的值更新为50，效果如下所示。

```
mysql> UPDATE shop1 SET age=50 WHERE name LIKE '%五';
Query OK, 2 rows affected (0.00 sec)
Rows matched: 2  Changed: 2  Warnings: 0;
```

（2）从运行结果可以看出修改了两行数据。使用查询语句查询数据表shop1，效果如下所示。

```
mysql> SELECT * FROM shop1;
+----+----------+--------+
| id | name     | age    |
+----+----------+--------+
|  0 | 王五     |   50   |
|  1 | 张三     |   18   |
|  2 | 李四     |   30   |
|  3 | 周五     |   50   |
|  4 | 刘柳     |   30   |
|  5 | 周武     |   40   |
|  6 | 郑旺     |   40   |
+----+----------+--------+
7 rows in set (0.00 sec)
```

从运行结果可以看出，name字段中以“五”结尾的数据行的age字段的值更新为50。


IN和NOTIN语句

4. IN与ONT IN语句

IN语句是SELECT语句的子句，其功能是选择和指定数值中相同的内容。ONT IN语句也是SELECT语句的子句，其功能是选择和指定数值中不相同的内容。

实例 4-10 使用IN语句查询并更新数据。

（1）将数据表shop1中id值为3和5的数据行的age字段的值改为99，效果如下所示。

```
mysql> UPDATE shop1 SET age=99 WHERE id IN(3,5);
Query OK, 2 rows affected (0.01 sec)
Rows matched: 2  Changed: 2  Warnings: 0
```

（2）从运行结果可以看出修改了2行数据。使用查询语句查询数据表shop1，效果如下所示。

```
mysql> SELECT * FROM shop1;
+----+----------+--------+
| id | name     | age    |
+----+----------+--------+
|  0 | 王五     |   50   |
|  1 | 张三     |   18   |
```

```
|  2 | 李四       |   30   |
|  3 | 周五       |   99   |
|  4 | 刘柳       |   30   |
|  5 | 周武       |   99   |
|  6 | 郑旺       |   40   |
+----+----------+--------+
7 rows in set (0.00 sec)
```

从运行结果可以看出，id值为3和5的数据行的age属性的是数据更新为99。

实例4-11 使用NOT IN语句查询并更新数据。

（1）将数据表shop1中id值为0的数据行的age字段的值改为99，效果如下所示。

```
mysql> UPDATE shop1 SET age=99 WHERE id NOT IN(1,2,3,4,5,6);
Query OK, 1 row affected (0.01 sec)
Rows matched: 1  Changed: 1  Warnings: 0
```

（2）从运行结果可以看出修改了1行数据。使用查询语句查询数据表shop1，效果如下所示。

```
mysql> SELECT * FROM shop1;
+----+----------+--------+
| id | name     | age    |
+----+----------+--------+
|  0 | 王五       |   99   |
|  1 | 张三       |   18   |
|  2 | 李四       |   30   |
|  3 | 周五       |   99   |
|  4 | 刘柳       |   30   |
|  5 | 周武       |   99   |
|  6 | 郑旺       |   40   |
+----+----------+--------+
7 rows in set (0.00 sec)
```

从运行结果可以看出，id值为1的数据行的age属性的数据更新为99。

4.2.4 更新符合正则表达式的数据


更新符合正则表达式的数据

通过正则表达式更新数据，需要使用MySQL数据库中的关键字REGEXP来实现。在使用时需要在关键字REGEXP后面紧跟正则表达式规则。

实例4-12 使用正则表达式实现数据更新。

（1）将数据表shop1中age值结尾0的数据行的age字段的值改为33，效果如下所示。

```
mysql> UPDATE shop1 SET age=33 WHERE age REGEXP '0$';
Query OK, 3 rows affected (0.02 sec)
Rows matched: 3  Changed: 3  Warnings: 0
```

（2）从运行结果可以看出修改了 3 行数据。使用查询语句查询数据表 shop1，效果如下所示。

```
mysql> SELECT * FROM shop1;
+----+----------+--------+
| id | name     | age    |
+----+----------+--------+
|  0 | 王五     |   99   |
|  1 | 张三     |   18   |
|  2 | 李四     |   33   |
|  3 | 周五     |   99   |
|  4 | 刘柳     |   33   |
|  5 | 周武     |   99   |
|  6 | 郑旺     |   33   |
+----+----------+--------+
7 rows in set (0.00 sec)
```

从运行结果可以看出，age 值以 0 结尾的数据行对的 age 值更新为 33。

任务 4-2

更新“学生信息表”中学生的年龄信息

更新“学生信息表”中学生的年龄信息

任务描述

（1）为“学生信息表”添加数据。

（2）更新“学生信息表”的年龄字段。

任务实施

1. 查询“学生信息表”的表结构与现有数据

查询“学生信息表”的表结构，效果如下所示。

```
mysql> DESC 学生信息表;
+--------+--------------+-------+-------+----------+--------+
| Field  | Type         | Null  | Key   | Default  | Extra  |
+--------+--------------+-------+-------+----------+--------+
| 学号   | int          | NO    | PRI   | NULL     |        |
| 姓名   | varchar(30)  | NO    |       | NULL     |        |
| 年龄   | int          | NO    |       | NULL     |        |
+--------+--------------+-------+-------+----------+--------+
3 rows in set (0.00 sec)
```

从运行结果可以看出，“学生信息表”的每个字段及其对应的数据类型。查询“学生信息表”中的现有数据，效果如下所示。

```
mysql> SELECT * FROM 学生信息表;
Empty set (0.00 sec)
```

从运行结果可以看出，“学生信息表”中的数据为空。

2. 为“学生信息表”添加数据

向“学生信息表”中添加多条数据内容，效果如下所示。

```
mysql> INSERT INTO 学生信息表 VALUES
    ->(1,'小明',25),
    ->(2,'小红',43),
    ->(3,'小花',40);
Query OK, 3 rows affected (0.01 sec)
Records: 3  Duplicates: 0  Warnings: 0
```

从运行效果可以看出添加了 3 条信息。

3. 再次查询“学生信息表”的数据

通过查询语句再次查询“学生信息表”中的数据，确认插入数据成功，效果如下所示。

```
mysql> SELECT * FROM 学生信息表;
+----------+------------+----------+
| 学号      | 姓名        | 年龄      |
+----------+------------+----------+
|      1   | 小明        |     25   |
|      2   | 小红        |     43   |
|      3   | 小花        |     40   |
+----------+------------+----------+
3 rows in set (0.00 sec)
```

从运行结果可以看出成功插入 3 条数据，但是年龄明显在插入的数据时出现了错误。

4. 更新学生的年龄信息

通过更新语句更新所有学生的年龄为 8，效果如下所示。

```
mysql> UPDATE 学生信息表 SET 年龄 = 8;
Query OK, 3 rows affected (0.01 sec)
Rows matched: 3  Changed: 3  Warnings: 0
```

5. 再次查看“学生信息表”

再次查询“学生信息表”，确认数据更新成功，效果如下所示。

```
mysql> SELECT * FROM 学生信息表;
+----------+------------+------------+
| 学号      | 姓名        | 年龄        |
+----------+------------+------------+
|      1   | 小明        |      8     |
|      2   | 小红        |      8     |
|      3   | 小花        |      8     |
+----------+------------+------------+
3 rows in set (0.00 sec)
```

从运行结果可以看出，“学生信息表”中的“年龄”字段全部更新为 8。


数据删除

4.3 删除数据

当数据表或数据表中的某些数据被废弃时可以对其进行删除处理。在MySQL中，删除数据需要使用使用DELETE语句来实现，同时可以配合使用WHERE字句指定删除数据的限制条件，其语法形式如下所示。

```
DELETE FROM table_name [WHERE condition]
```

其中，各属性功能如下所示。

（1）table_name：需要删除数据的表名称。

（2）condition：WHERE子句的条件，该项可以省略，省略后会删除整个数据表。

4.3.1 删除数据表中特定的数据

通过WHERE字句可以实现删除数据表中的某一行数据。

实例 4-13 删除数据表中的指定行数据。

（1）删除shop1数据表中id值为0的数据行内容，效果如下所示。

```
mysql> DELETE FROM shop1 WHERE id=0;
Query OK, 1 row affected (0.41 sec)
```

（2）从运行结果可以看出删除了1条内容，通过查询语句查看数据表shop1，确认删除是否成功，效果如下所示。

```
mysql> SELECT * FROM shop1;
+----+----------+--------+
| id | name     | age    |
+----+----------+--------+
|  1 | 张三     |   18   |
|  2 | 李四     |   33   |
|  3 | 周五     |   99   |
|  4 | 刘柳     |   33   |
|  5 | 周武     |   99   |
|  6 | 郑旺     |   33   |
+----+----------+--------+
6 rows in set (0.00 sec)
```

从运行结果可以看出，id值为0的数据行已被成功删除。


删除某个范围内的数据

4.3.2 删除某个范围内的数据

如果要一次性删除多条数据，则可以配合使用BETWEEN ... AND语句、比较运算符、LIKE、IN，以及NOT IN等语句实现。

1. 使用BETWEEN...AND语句删除数据

实例 4-14 使用BETWEEN...AND语句删除多行内容。

（1）将shop1数据中表中id值在5和6之间的数据行内容进行删除，效果如下所示。

```
mysql> DELETE FROM shop1 WHERE id BETWEEN 5 AND 6;
Query OK, 2 rows affected (0.01 sec)
```

（2）从运行结果可以看出删除了2条内容，通过查询语句查看数据表shop1，确认删除是否成功，效果如下所示。

```
mysql> SELECT * FROM shop1;
+----+--------+--------+
| id | name   | age    |
+----+--------+--------+
|  1 | 张三   |   18   |
|  2 | 李四   |   33   |
|  3 | 周五   |   99   |
|  4 | 刘柳   |   33   |
+----+--------+--------+
4 rows in set (0.00 sec)
```

从运行结果可以看出，id值为5和6的数据行已被成功删除。

2. 使用运算符删除数据

实例 4-15 使用运算符删除多行内容。

（1）将shop1数据中表中id值小于3的数据行内容进行删除，效果如下所示。

```
mysql> DELETE FROM shop1 WHERE id <3;
Query OK, 2 rows affected (0.01 sec)
```

（2）从运行结果可以看出删除了2条内容，通过查询语句查看数据表shop1，确认删除是否成功，效果如下所示。

```
mysql> SELECT * FROM shop1;
+----+--------+--------+
| id | name   | age    |
+----+--------+--------+
|  3 | 周五   |   99   |
|  4 | 刘柳   |   33   |
+----+--------+--------+
2 rows in set (0.00 sec)
```

从运行结果可以看出，id值为1和2的数据行已被成功删除。

3. LIKE语句删除数据

实例 4-16 使用LIKE语句删除内容。

（1）将shop1数据中表中name字段值包含“五”的对应数据行进行删除，效果如下所示。

```
mysql> DELETE FROM shop1 WHERE name LIKE '%五%';
Query OK, 1 row affected (0.01 sec)
```

（2）从运行结果可以看出删除了 1 条内容，通过查询语句查看数据表 shop1，确认删除是否成功，效果如下所示。

```
mysql> SELECT * FROM shop1;
+----+--------+--------+
| id | name   | age    |
+----+--------+--------+
|  4 | 刘柳   |   33   |
+----+--------+--------+
1 row in set (0.12 sec)
```

从运行结果可以看出，name 值包含“五”的数据行已被成功删除。

4. IN 语句删除数据

实例 4-17 使用 IN 语句删除指定数据行。

（1）将 shop1 数据中表中 id 字段值包含 4 的对应数据行进行删除，效果如下所示。

```
mysql> DELETE FROM shop1 WHERE id IN (4);
Query OK, 1 row affected (0.01 sec)
```

（2）从运行结果可以看出删除了 1 条内容，通过查询语句查看数据表 shop1，确认删除是否成功，效果如下所示。

```
mysql> SELECT * FROM shop1;
Empty set (0.00 sec)
```

从运行结果可以看出，id 值为 4 的数据被删除，shop1 表成为一个张空表。


删除符合正则表达式的数据

4.3.3 删除符合正则表达式的数据

在删除语句中可以使用 WHERE 子句以正则表达式作为删除限制条件使用。同样，使用正则表达式时需要使用 REGEXP 关键字。

实例 4-18 使用正则表达式删除指定行数据。

（1）删除数据表中 shop1 中 name 属性以“张”开头的数据行内容。首先查看 shop1 数据表中的现有数据，效果如下所示。

```
mysql> SELECT * FROM shop1;
+----+---------+---------+
| id | name    | age     |
+----+---------+---------+
|  1 | 张琦    |   19    |
|  2 | 郑旺    |   20    |
|  3 | 张东琦  |   19    |
|  4 | 王猛    |   20    |
```

```
|  5 | 张三     |   19    |
|  6 | 刘柳     |   20    |
|  7 | 张利     |   19    |
|  8 | 刘风     |   20    |
+----+---------+---------+
8 rows in set (0.00 sec)
```

（2）使用正则表达式删除语句删除以“张”开头的数据，效果如下所示。

```
mysql> DELETE FROM shop1 WHERE name REGEXP '^张';
Query OK, 4 rows affected (0.01 sec)
```

从运行结果可以看出删除了4条内容，通过查询语句查看数据表shop1，确认删除是否成功，效果如下所示。

```
mysql> SELECT * FROM shop1;
+----+--------+-------+
| id | name   | age   |
+----+--------+-------+
|  2 | 郑旺   |   20  |
|  4 | 王猛   |   20  |
|  6 | 刘柳   |   20  |
|  8 | 刘风   |   20  |
+----+--------+-------+
4 rows in set (0.00 sec)
```

从运行结果可以看出，name值中包含“张”字的数据行已被成功删除。

4.3.4 删除数据表中的所有数据


删除数据表中的所有数据

如果要删除整个数据表，只需要将删除语法中的WHERE字句进行省略即可。

实例4-19 删除数据表中的所有数据。

（1）删除数据表shop1中的所有数据内容，效果如下所示。

```
mysql> DELETE FROM shop1;
Query OK, 4 rows affected (0.01 sec)
```

（2）从运行结果可以看出删除了4条内容，通过查询语句查看数据表shop1，确认删除是否成功，效果如下所示。

```
mysql> SELECT * FROM shop1;
Empty set (0.00 sec)
```

从运行结果可以看出，shop1表成为一个空表。

任务 4-3

删除“学生信息表”中“小红”的信息

删除“学生信息表”中“小红”的信息

任务描述

（1）查询“学生信息表”中的信息。

（2）删除“小红”的数据。

任务实施

1. 查询现有数据

查询“学生信息表”的现有数据，效果如下所示。

```
mysql> SELECT * FROM 学生信息表;
+---------+---------+---------+
| 学号    | 姓名    | 年龄    |
+---------+---------+---------+
|       1 | 小明    |       8 |
|       2 | 小红    |       8 |
|       3 | 小花    |       8 |
+---------+---------+---------+
3 rows in set (0.00 sec)
```

从运行结果可以看出，“学生信息表”中有“小红”的对应信息。

2. 删除数据

删除“小红”的相关数据行，效果如下所示。

```
mysql> DELETE FROM 学生信息表 WHERE 姓名='小红';
Query OK, 1 row affected (0.01 sec)
```

从运行效果可以看出删除了 1 条信息。

3. 再次查询数据表

通过查询语句再次查询“学生信息表”的数据，确认删除数据成功，效果如下所示。

```
mysql> SELECT * FROM 学生信息表;
+---------+---------+---------+
| 学号    | 姓名    | 年龄    |
+---------+---------+---------+
|       1 | 小明    |       8 |
|       3 | 小花    |       8 |
+---------+---------+---------+
2 rows in set (0.00 sec)
```

从运行结果可以看出，“小红”的信息已经被删除。

任务 4-4

实现“学生信息表”的插入、更新和删除功能


实现“学生信息表”的插入、更新和删除功能

任务描述

（1）插入“李磊”的信息。

（2）更新“李磊”的年龄。

（2）删除“小花”的信息。

任务实施

1. 查询现有数据

选择bookshop数据库，然后查询“学生信息表”的数据内容，效果如下所示。

```
mysql> USE bookshop;
Database changed
mysql> SELECT * FROM 学生信息表;
+---------+---------+---------+
| 学号    | 姓名    | 年龄    |
+---------+---------+---------+
|       1 | 小明    |       8 |
|       3 | 小花    |       8 |
+---------+---------+---------+
2 rows in set (0.00 sec)
```

2. 添加数据

向“学生信息表”中添加“李磊”的信息数据，效果如下所示。

```
mysql> INSERT INTO 学生信息表 VALUES (2,'李磊',55);
Query OK, 1 row affected (0.01 sec)
```

从运行结果可以看出添加了一行数据。

3. 再次查询数据

通过查询语句再次查询“学生信息表”中的数据，效果如下所示。

```
mysql> SELECT * FROM 学生信息表;
+---------+---------+---------+
| 学号    | 姓名    | 年龄    |
+---------+---------+---------+
|       1 | 小明    |       8 |
|       2 | 李磊    |      55 |
|       3 | 小花    |       8 |
+---------+---------+---------+
3 rows in set (0.00 sec)
```

从运行结果可以看出，成功插入了“李磊”的信息。

4. 更新数据

通过更新语句更新“李磊”的年龄为8，效果如下所示。

```
mysql> UPDATE 学生信息表 SET 年龄 = 8 WHERE 姓名= '李磊';
Query OK, 1 row affected (0.01 sec)
Rows matched: 1  Changed: 1  Warnings: 0
```

从运行结果可以看出更新了 1 条数据。

5. 再次查看数据

再次查询“学生信息表”，确认数据更新成功，效果如下所示。

```
mysql> SELECT * FROM 学生信息表;
+---------+---------+---------+
| 学号    | 姓名    | 年龄    |
+---------+---------+---------+
|       1 | 小明    |       8 |
|       2 | 李磊    |       8 |
|       3 | 小花    |       8 |
+---------+---------+---------+
3 rows in set (0.00 sec)
```

从运行结果可以看出，“学生信息表”中“李磊”的年龄更新为了 8。

6. 删除数据

删除“小花”的相关数据行，效果如下所示。

```
mysql> DELETE FROM 学生信息表 WHERE 姓名='小花';
Query OK, 1 row affected (0.01 sec)
```

从运行效果可以看出删除了 1 条信息。

7. 再次查询数据表

通过查询语句再次查询“学生信息表”中的数据，确认删除数据成功，效果如下所示。

```
mysql> SELECT * FROM 学生信息表;
+---------+---------+---------+
| 学号    | 姓名    | 年龄    |
+---------+---------+---------+
|       1 | 小明    |       8 |
|       2 | 李磊    |       8 |
+---------+---------+---------+
2 rows in set (0.00 sec)
```

从运行结果可以看出，“小花”的信息已经被删除。

1. 正则表达式

正则表达式是对字符串操作的一种逻辑公式，就是用事先定义好的特定字符或与这些特定字符的组合，组成一个“规则字符串”，这个“规则字符串”用来表达对字符串的一种过滤逻辑。

扫一扫
看一看

2. 正则表达式的语法

正则表达式的语法如下所示。

```
^[条件]{位数}$
```

其中，各部分功能如下所示。

（1）^：表示正则表达式为匹配输入字符串的开始位置。

（2）[]：指定要匹配的字符内容，包括字符、数字以及特殊符号。

（3）{}：指定匹配长度。例如，{3,5} 表示要匹配的数据为 3~15 个字符。就像匹配手机号码时可以限制为 11 个字符的内容。

（4）$：表示匹配的结束标记。

如果单独使用开始符号（^）表示要匹配以某个字符或数字开始的字符串。例如，匹配以“王”开头的字符串内容，正则表达式语句如下所示。

```
^王
```

如果单独使用结束标记符（$）表示匹配某个字符结束的字符串内容。例如，匹配以“王”结束的字符串内容，正则表达式语句如下所示。

```
$王
```

华为逆境中崛起——HMS 与鸿蒙系统引领技术革新

自 21 世纪 10 年代末期，华为公司在国际舞台上遭遇了前所未有的挑战。美国及其西方盟友对华为实施了一系列严厉的限制措施，其中包括一项极具争议的“芯片禁令”。这一政策的实施直接导致了华为在全球智能手机市场的领先地位受到冲击，其手机销量和市场份额遭受了严重打击，从昔日的全球第一宝座跌落。此外，华为手机长期以来依赖的谷歌移动服务（GMS）也受到了限制，这无疑给华为的海外业务带来了巨大的挑战。面对这样

的逆境，华为并没有选择退缩，而是展现出了顽强的斗志和创新精神。

2019 年，华为重磅推出了自家的 HMS（华为移动服务）和鸿蒙操作系统。HMS 的推出，不仅填补了 GMS 留下的空白，还为全球开发者提供了一个全新的平台，以支持他们在华为设备上开发和分发应用。而鸿蒙操作系统的问世，更是标志着华为在操作系统领域的重大突破，它不仅性能卓越，而且具有高度的安全性和隐私保护能力。华为的这一系列举措，不仅令全球科技界为之震撼，更让那些试图限制华为发展的国外势力不得不重新评估华为的实力。华为的坚韧不拔和不断创新，展现了中国科技企业的真正实力，也向世界证明了，即使在重重压力之下，华为依然能够迎难而上，创造出令人瞩目的成就。

（资料来源：根据相关资料整理编写）

一、填空题

1. 向 MySQL 数据表中插入完整的行记录时，可以选择是否指定插入数据的____或____。

2. 实现对数据表中的数据进行更新操作，需要使用 MySQL 中的____语句。

二、选择题

1. 以下选项中可以实现数据删除的语句是（　　）。

A. INTO　　B. INSERT　　C. UPDATE　　D. DELETE

2. 以下选项中可以实现使用正则表达式的关键字是（　　）。

A. BETWEEN　　B. LIKE　　C. REGEXP　　D. UPDATE

三、判断题

1. 删除数据需要使用 DELETE 语句来实现，并且可以配合使用 WHERE 字句指定删除的数据。（　　）

2. 更新操作时不可以同时更新多行数据内容。（　　）

四、操作题

实现向“学生信息表”中插入新的数据，数据如下所示。

学号：3
姓名：韩梅梅
年龄：8