第 5 章
单 表 操 作

学习目标

- 掌握复制表结构与数据的操作
- 掌握数据的排序、限量与分组
- 掌握常用聚合函数与比较函数
- 掌握 MySQL 常用运算符的使用

在前面的章节中已经学习了数据表的创建、数据类型、约束、字符集的设置，以及数据的基本增、删、改、查操作。但实际需求会更加复杂，前面学习过的内容不能够完全满足开发需求，所以需要深入学习更多的数据操作。例如，为数据表插入大量的测试数据，对查询的数据进行筛选、分组、排序或限量。本章将围绕数据库中的单表操作进行详细讲解。

5.1 数据操作

5.1.1 复制表结构和数据

1. 复制已有的表结构

在开发时，若需要创建一个与已有数据表相同结构的数据表时，可以通过以下的语法完成表结构的复制。基本语法格式如下。

```
CREATE [TEMPORARY] TABLE [IF NOT EXISTS]表名
{ LIKE 旧表名 | (LIKE 旧表名) }
```

在上述语法中，仅能从“旧表名”中复制一份相同的表结构，但不会复制表中保存的数据。其中，“{}”表示语法在使用时可以任选其中一种，“|”表示或的意思。因此，在复制已有的表结构时，可以使用“LIKE 旧表名”或“(LIKE 旧表名)”中的任意一种语法格式。

下面从第 4 章创建的 shop 数据库中，复制一份与 sh_goods 数据表相同结构的 my_goods 表到 mydb 数据库中，具体 SQL 语句如下。

```
mysql>USE shop;
Database changed
mysql>CREATE  TABLE  mydb.my_goods LIKE sh_goods;
Query OK, 0 rows affected (0.07 sec)
```

按以上步骤创建完成后，可以利用 SHOW CREATE TABLE 查看 my_goods 表的结构，具体 SQL 语句如下。

```
mysql> SHOW CREATE TABLE mydb.my_goods\G
*************************** 1. row ***************************
       Table: my_goods
Create Table: CREATE TABLE `my_goods` (
  `id` int(10) unsigned NOT NULL AUTO_INCREMENT COMMENT '商品 id',
  `category_id` int(10) unsigned NOT NULL DEFAULT '0' COMMENT '分类 id',
  ⋮此处省略部分字段
) ENGINE=InnoDB DEFAULT CHARSET=utf8
1 row in set (0.00 sec)
```

从上述结果可知，只需一行操作就可以依据已有的表创建出与其相同结构的表。

2. 复制已有的表数据

数据复制也可称为蠕虫复制，是新增数据的一种方式，它是从已有的数据中获取数据，并且将获取到的数据插入到对应的数据表中，实现成倍的增加。需要注意的是，此种方式获取数据与插入数据的表结构要相同，否则可能会遇到插入不成功的情况。基本语法格式如下。

```
INSERT [INTO] 数据表名 1 [(字段列表)] SELECT [(字段列表)] FROM 数据表名 2;
```

在上述语法中，数据表名 1 和数据表名 2 通常使用的是同一个表（如 my_goods 表），从而可在短期内快速增加表的数据量，测试表的压力以及效率等，相关内容会在本书数据库优化部分讲解，此处读者了解即可。

下面利用以上语法从 sh_goods 表中复制数据到 my_goods 表中，具体 SQL 语句如下。

```
mysql> INSERT INTO mydb.my_goods SELECT * FROM sh_goods;
Query OK, 10 rows affected (0.01 sec)
Records: 10  Duplicates: 0  Warnings: 0
```

执行完上述 SQL 语句后，使用 SELECT 查看商品数据的添加情况，会看到数据已从 sh_goods 表完全复制到了 my_goods 表中，这里不再演示。

需要注意的是，若数据表中含有主键，而主键又具有唯一性，所以在数据复制时还要考虑主键冲突的问题。例如，通过以下方式再向 my_goods 表中添加数据，系统会报“主键重复”的错误。具体 SQL 语句如下。

```
mysql> INSERT INTO mydb.my_goods SELECT * FROM sh_goods;
ERROR 1062 (23000): Duplicate entry '1' for key 'PRIMARY'
```

对于以上主键冲突的问题，数据复制时可以指定除 id 主键外的任意字段完成。具体 SQL 语句如下。

```
mysql> INSERT INTO mydb.my_goods (category_id, name, keyword, price,
    -> content) SELECT category_id, name, keyword, price, content
```

```
    ->FROM sh_goods;
Query OK, 10 rows affected (0.00 sec)
Records: 10  Duplicates: 0  Warnings: 0
```

多学一招：临时表的使用

临时表指的是一种仅在当前会话中可见，并在当前会话关闭时自动删除的数据表。它主要用于临时存储数据。临时表的语法很简单，只需在 CREATE 与 TABLE 关键字中间添加 TEMPORARY 即可。示例如下。

```
#方式 1：创建临时表
CREATE TEMPORARY TABLE mydb.tmp_table1 (id int);
#方式 2：创建临时表
CREATE TEMPORARY TABLE mydb.tmp_table2 SELECT id, name FROM shop.sh_goods;
```

在上述语句中，创建临时表时指定的数据库可以是 MySQL 服务器中存在的数据库，也可以是不存在的数据库。若数据库不存在，操作临时表时必须使用“数据库.临时表名”指定临时表所在的数据库。

除此之外，临时表中数据的操作与普通表相同，都可以进行 SELECT、INSERT、UPDATE 和 DELETE 操作。这里不再演示。

需要注意的是，SHOW TABLES 不能查看指定数据库下有哪些临时表，并且临时表的表名必须使用 ALTER TABLE 修改，而不能使用 RENAME TABLE…TO 修改。

5.1.2 解决主键冲突

在对数据表插入数据时，若表中的主键含有实际的业务意义，因此在插入数据时若不能确定对应的主键是否存在，往往会出现主键冲突的情况。例如，mydb.my_goods 表经过数据复制以后，再插入编号为 20 的商品信息（橡皮，文具类，用于修正书写错误），具体 SQL 语句及执行结果如下。

```
mysql>INSERT INTO mydb.my_goods(id, name, content, keyword)
    ->VALUES (20, '橡皮', '修正书写错误', '文具');
ERROR 1062 (23000): Duplicate entry '20' for key 'PRIMARY'
```

从上述的执行结果可知，系统提示插入数据的主键发生冲突。若要解决这类问题，MySQL 中提供了两种方式，分别为主键冲突更新和主键冲突替换。

1. 主键冲突更新

主键冲突更新操作指的是，当插入数据的过程中若发生主键冲突，则插入数据操作利用更新的方式实现。基于语法格式如下。

```
INSERT [INTO] 数据表名 [(字段列表)] {VALUES | VALUE} (值列表)
ON DUPLICATE KEY UPDATE 字段名 1 =新值 1[,字段名 2 =新值 2] …;
```

从上述语法可知，在 INSERT 语句后添加 ON DUPLICATE KEY UPDATE 可在发生

主键冲突时，更新此条记录中通过“字段名 1 = 新值 1[，字段名 2 = 新值 2] …”设置的字段名对应的新值。

例如，修改以上发生主键冲突的插入语句，具体 SQL 语句及执行结果如下。

```
mysql> INSERT INTO mydb.my_goods (id, name, content, keyword)
    -> VALUES (20, '橡皮', '修正书写错误', '文具')
    -> ON DUPLICATE KEY UPDATE name = '橡皮', content = '修正书写错误',
    -> keyword = '文具';
Query OK, 2 rows affected (0.00 sec)
mysql> SELECT name, content, keyword FROM mydb.my_goods WHERE id = 20;
+------+------------+----------+
| name | content    | keyword  |
+------+------------+----------+
|橡皮  | 修正书写错误 | 文具     |
+------+------------+----------+
1 row in set (0.00 sec)
```

以上执行结果中，当插入的记录与数据表中已存在的记录主键冲突时，返回的结果为“2 rows affected”。

2. 主键冲突替换

主键冲突替换操作指的是，当插入数据的过程中若发生主键冲突，则删除此条记录，并重新插入。基于语法格式如下。

```
REPLACE [INTO] 数据表名 [(字段列表)]
{VALUES | VALUE} (值列表) [, (值列表)] …;
```

从上述语法可知，REPLACE 语句与 INSERT 语句的使用类似，区别在于前者每执行一次就会发生两个操作(删除记录和插入记录)。例如，修改以上发生主键冲突的插入语句，具体 SQL 语句及执行结果如下。

```
mysql> REPLACE INTO mydb.my_goods (id, name, content, keyword)
    -> VALUES (20, '橡皮', '修正书写错误', '文具');
Query OK, 2 rows affected (0.00 sec)
mysql> SELECT name, content, keyword FROM mydb.my_goods WHERE id = 20;
+------+------------+----------+
| name | content    | keyword  |
+------+------------+----------+
|橡皮  | 修正书写错误 | 文具     |
+------+------------+----------+
1 row in set (0.00 sec)
```

从以上的执行结果可知，REPLACE 替换与 ON DUPLICATE KEY UPDATE 更新都能解决插入数据时主键冲突的问题，但 REPLACE 更适合插入数据字段特别多的情况。

5.1.3　清空数据

除了第 2 章讲解的 DELETE 语句可以删除数据外，在 MySQL 中还可以利用

TRUNCATE 清空指定数据表中的全部数据。其基本语法格式如下。

```
TRUNCATE [TABLE] 表名
```

下面演示清空 mydb 数据库下的 my_goods 表，具体 SQL 语句及执行结果如下。

```
mysql>TRUNCATE TABLE mydb.my_goods;
Query OK, 0 rows affected (0.08 sec)
```

需要注意的是，TRUNCATE 操作虽然与 DELETE 语句的使用非常相似，但是两者在本质上有一定的区别，具体如下。

- 实现方式不同：TRUNCATE 本质上先执行删除(DROP)数据表的操作，然后再根据有效的表结构文件(.frm)重新创建数据表的方式来实现数据清空操作。而 DELETE 语句则是逐条地删除数据表中保存的记录。
- 执行效率不同：在针对大型数据表(如千万级的数据记录)时，TRUNCATE 清空数据的实现方式决定了它比 DELETE 语句删除数据的方式执行效率更高。
- 对 AUTO_INCREMENT 的字段影响不同：TRUNCATE 清空数据后，再次向表中添加数据，自动增长字段会从默认的初始值重新开始，而使用 DELETE 语句删除表中的记录时，则不影响自动增长值。
- 删除数据的范围不同：TRUNCATE 语句只能用于清空表中的所有记录，而 DELETE 语句可通过 WHERE 指定删除满足条件的部分记录。
- 返回值含义不同：TRUNCATE 操作的返回值一般是无意义的，而 DELETE 语句则会返回符合条件被删除的记录数。
- 所属 SQL 语言的不同组成部分：DELETE 语句属于 DML 数据操作语句，而 TRUNCATE 通常被认为是 DDL 数据定义语句。

为了读者更好地理解，重新使用 5.1.1 小节中数据复制的方式完成 my_goods 表的数据添加，对比 TRUNCATE 与 DELETE 数据表 my_goods 后，重新插入一条记录的效果。

(1) 为 my_goods 插入 10 条记录后，使用 TRUNCATE 清空并重新插入数据。

```
mysql>TRUNCATE TABLE mydb.my_goods;
Query OK, 0 rows affected (0.01 sec)
mysql>INSERT INTO mydb.my_goods (name, content, keyword)
    ->VALUES ('苹果', '一种很有营养的水果', '水果');
Query OK, 1 row affected (0.00 sec)
mysql>SELECT id, name, content, keyword FROM mydb.my_goods;
+----+------+----------------------+-------------+
| id | name | content              | keyword     |
+----+------+----------------------+-------------+
| 1  | 苹果 | 一种很有营养的水果   | 水果        |
+----+------+----------------------+-------------+
1 row in set (0.00 sec)
```

(2) 为 my_goods 插入 10 条记录后，使用 DELETE 删除并重新插入数据。

```
mysql> DELETE FROM mydb.my_goods;
Query OK, 10 rows affected (0.00 sec)
mysql> INSERT INTO mydb.my_goods (name, content, keyword)
    ->VALUES ('苹果', '一种很有营养的水果', '水果');
Query OK, 1 row affected (0.00 sec)
mysql> SELECT id, name, content, keyword FROM mydb.my_goods;
+----+------+---------------------+------------+
| id | name | content             | keyword    |
+----+------+---------------------+------------+
| 11 | 苹果 | 一种很有营养的水果  | 水果       |
+----+------+---------------------+------------+
1 row in set (0.00 sec)
```

从上述的操作中不难看出 TRUNCATE 操作和 DELETE 语句在删除数据的区别，具体如表 5-1 所示。

表 5-1　TRUNCATE 与 DELETE 对比

操　作	返　回　值	id(插入字段增长值)	执 行 效 率
TRUNCATE	0 rows affected	1	0.01s
DELETE	10 rows affected	11	0.00s

在表 5-1 中，DELETE 的返回值表示有 10 条记录受影响，而 TRUNCATE 的返回值为 0，明显无实际意义；删除数据后，再次新增一条记录后，查询到的商品 id 值明显不同，TRUNCATE 后 id 字段从默认值 1 开始增长，而 DELETE 后 id 值则继续从 10 开始增长，因此最后结果为 11。

需要注意的是，当删除的数据量很小时，DELETE 的执行效率要比 TRUNCATE 高；只有删除的数据量很大时，才能看出 TRUNCATE 的执行效率比 DELETE 高。例如，my_goods 表含有 20480 条记录时，使用 TRUNCATE 清空数据的执行时间仍然为 0.01 秒，而 DELETE 语句的执行时间会增加到 0.17 秒左右。因此，在实际开发时具体使用哪种方式执行删除操作，需要根据实际需求进行合理的选择。

5.1.4　去除重复记录

实际应用中，出于对数据的分析需求，有时需要去除查询记录中重复的数据。例如，查看商品表中共有几种分类。此时可以使用 SELECT 语句的选项，其基本语法格式如下。

```
SELECT select 选项 字段列表 FROM 数据表
```

在上述语法中，“select 选项”默认值为 All，表示保存所有查询到的记录；当设置为 DISTINCT 时，表示去除重复记录，只保留一条。需要注意的是，当查询记录的字段有多个时，必须所有字段的值完全相同才被认为是重复记录。

为了方便案例的演示，本章以下所有小节均使用 shop 数据库下的 sh_goods 表进行操作。下面查看 sh_goods 表中所有的 keyword 字段的值，具体 SQL 语句如下。

```
mysql>SELECT keyword FROM sh_goods;
+-----------+
| keyword     |
+-----------+
| 文具          |
| 文具          |
| 文具          |
| 电子产品      |
| 电子产品      |
| 电子产品      |
| 电子产品      |
| 电子产品      |
| 服装          |
| 服装          |
+-----------+
10 rows in set (0.00 sec)
```

从上述执行结果可知，查询出的 keyword 字段值有 3 条为“文具”，5 条为“电子产品”，2 条为“服装”，即使有重复的数据，默认情况下依然保存了所有查询到的记录。

接下来，查看 sh_goods 表中去除重复记录的 keyword 字段值，具体 SQL 语句如下。

```
mysql>SELECT DISTINCT keyword FROM sh_goods;
+-------------+
| keyword       |
+-------------+
| 文具            |
| 电子产品        |
| 服装            |
+-------------+
3 rows in set (0.00 sec)
```

从上述执行结果可知，此次的查询结果仅有 3 条记录，分别为文具、电子产品和服装，不再包含重复的记录。

5.2 排序与限量

在电子商务网站迅速发展的今天，商品的种类与数量数以万计，甚至更多。因此，人们在查看某种商品时经常需要对其进行排序，让满足要求的数据显示到最前面，方便进一步操作。同时，为了提高执行的效率，经常需要对操作的数据进行限量。例如，仅查看符合要求的 10 条记录。本节将针对 MySQL 中排序和限量操作进行详细讲解。

5.2.1 排序

在项目开发时，为了使查询的数据结果满足用户的要求，通常会对查询出的数据进行上升或下降的排序。MySQL 针对不同的开发需求提供两种排序的方式，分别为单字段排序和多字段排序，接下来将对这两种排序方式的语法及使用进行详细讲解。

1. 单字段排序

单字段排序指的是查询时仅按照一个指定字段进行升序或降序排序。其基本语法格式如下。

```
SELECT  * |{字段列表} FROM 数据表名
ORDER BY 字段名 [ASC | DESC];
```

在上述语法中，ASC 表示升序，DESC 表示降序。而 ORDER BY 默认值为 ASC。

下面按照商品价格从高到低依次显示 sh_goods 表中的所有商品。具体 SQL 语句及执行结果如下。

```
mysql> SELECT id, name, price FROM sh_goods ORDER BY price DESC;
+----+-----------+----------+
| id  | name          | price      |
+----+-----------+----------+
|  4  | 超薄笔记本    | 5999.00    |
|  8  | 办公电脑      | 2000.00    |
|  5  | 智能手机      | 1999.00    |
|  9  | 收腰风衣      |  299.00    |
|  7  | 头戴耳机      |  109.00    |
|  6  | 桌面音箱      |   69.00    |
| 10  | 薄毛衣        |   48.00    |
|  2  | 钢笔          |   15.00    |
|  3  | 碳素笔        |    1.00    |
|  1  | 2B 铅笔       |    0.50    |
+----+-----------+----------+
10 rows in set (0.00 sec)
```

从以上的执行结果可知，查询到的商品显示顺序以价格为标准，从高到低依次显示。

2. 多字段排序

当在开发中需要根据多个条件对查询的数据进行排序时，可以采用多字段排序。其基本语法格式如下。

```
SELECT  * |{字段列表} FROM 数据表名
ORDER BY 字段名 1 [ASC | DESC] [, 字段名 2 [ASC | DESC]]…;
```

在上述语法中，多字段排序首先按照字段名 1 进行排序，当字段 1 的值相同时，再按照字段名 2 进行排序，依此类推。

下面查询 sh_goods 表中的数据，让数据在显示时首先按商品分类 category_id 升序排序，然后再按商品价格 price 从高到低依次排序。具体 SQL 语句及执行结果如下。

```
mysql> SELECT category_id, id, name, price FROM sh_goods
    ->ORDER BY category_id, price DESC;
+-------------+----+-------------+----------+
| category_id    | id  | name          | price      |
```

```
+-------------+----+-------------+----------+
|           3 |  2 | 钢笔        |   15.00  |
|           3 |  3 | 碳素笔      |    1.00  |
|           3 |  1 | 2B 铅笔     |    0.50  |
|           6 |  5 | 智能手机    | 1999.00  |
|           8 |  6 | 桌面音箱    |   69.00  |
|           9 |  7 | 头戴耳机    |  109.00  |
|          10 |  8 | 办公电脑    | 2000.00  |
|          12 |  4 | 超薄笔记本  | 5999.00  |
|          15 |  9 | 收腰风衣    |  299.00  |
|          16 | 10 | 薄毛衣      |   48.00  |
+-------------+----+-------------+----------+
10 rows in set (0.00 sec)
```

在上述执行结果中，查询的所有数据首先按 category_id 升序排序，相同 category_id 值的记录按照 price 字段降序排序。

此外由于数据表的字符集是 utf8，当排序的字段为中文时，默认不会按照中文拼音的顺序排序。那么在不改变数据表结构的情况下，可以使用“CONVERT(字段名 USING gbk)”函数强制让指定的字段按中文排序。

值得一提的是，在按照指定字段进行升序排列时，如果某条记录的字段值为 NULL，则系统会将 NULL 看作是最小的值，从而将其显示在查询结果中的第一条记录的位置。

5.2.2 限量

对于一次性查询出的大量记录，不仅不便于阅读查看，还会浪费系统效率。为此，MySQL 中提供了一个关键字 LIMIT，可以限定记录的数量，也可以指定查询从哪一条记录开始。其基本语法格式如下。

```
SELECT [select 选项] 字段列表 FROM 数据表名
[WHERE 条件表达式] [ORDER BY 字段 ASC|DESC]
LIMIT [OFFSET,] 记录数;
```

在上述语法中，“记录数”表示限定获取的最大记录数量，也就是说，在“记录数”大于数据表符合要求的实际记录数量时，以实际记录数为准；当 LIMIT 后仅含有此参数时，表示从第 1 条记录开始获取；可选项 OFFSET 表示偏移量，用于设置从哪条记录开始，MySQL 中默认第 1 条记录的偏移量值为 0，第 2 条记录的偏移量值为 1，依此类推。

下面以 sh_goods 表为例，演示限量查询记录与如何获取指定区间的记录。

(1) 限制记录数。查询 sh_goods 表中价格最贵的商品，具体 SQL 语句及执行结果如下。

```
mysql> SELECT id, name, price FROM sh_goods
    ->ORDER BY price DESC LIMIT 1;
+----+------------+----------+
| id | name       | price    |
+----+------------+----------+
|  4 | 超薄笔记本 | 5999.00  |
+----+------------+----------+
1 row in set (0.00 sec)
```

在上述 SQL 语句中，利用商品价格从高到低依次排序，然后再利用 LIMIT 限制查询出的记录数为 1 条，即可获取查询记录中的第 1 条记录，也就是价格最贵的商品。

(2) 获取指定区间的记录。获取指定区间的记录通常在项目开发中用于实现数据的分页展示，从而缓解网络和服务器的压力。例如，从第 1 条记录开始，获取 5 条商品记录，商品记录中包含 id、name 和 price，具体 SQL 语句及执行结果如下。

```
mysql>SELECT id, name, price FROM sh_goods LIMIT 0, 5;
+----+------------+----------+
| id  | name          | price     |
+----+------------+----------+
|  1  | 2B 铅笔       |    0.50   |
|  2  | 钢笔          |   15.00   |
|  3  | 碳素笔        |    1.00   |
|  4  | 超薄笔记本    | 5999.00   |
|  5  | 智能手机      | 1999.00   |
+----+------------+----------+
5 rows in set (0.00 sec)
```

在上述 SQL 语句中，LIMIT 关键字后的“0”表示第 1 条记录的偏移量，“5”表示从第 1(偏移量＋1)条记录开始最多获取 5 条记录。

多学一招：排序后限量更新或删除数据

在 MySQL 中除了对查询记录进行排序和限量外，对数据表中记录的更新与删除操作也可以进行排序和限量。其基本语法格式如下。

```
#数据更新的排序与限量
UPDATE 数据表名 SET 字段=新值, … [WHERE 条件表达式]
ORDER BY 字段 ASC|DESC LIMIT 记录数;
#数据删除的排序与限量
DELETE FROM 数据表名 [WHERE 条件表达式]
ORDER BY 字段 ASC|DESC LIMIT 记录数;
```

在上述语法中，UPDATE 和 DELETE 操作中添加 ORDER BY 表示根据指定的字段，按顺序更新或删除符合条件的记录。如果 UPDATE 和 DELETE 操作没有添加 WHERE 条件，则可以使用 LIMIT 来限制更新和删除的数量。

例如，为 sh_goods 表中价格最便宜的两种商品库存设置为 500，具体 SQL 语句及查询结果如下。

```
mysql>UPDATE sh_goods SET stock =500
    ->ORDER BY price ASC
    ->LIMIT 2;
Query OK, 2 rows affected (0.00 sec)
Rows matched: 2  Changed: 2  Warnings: 0
mysql>SELECT id, name, price, stock FROM sh_goods ORDER BY price;
+----+------------+----------+-------+
| id  | name          | price     | stock  |
```