第5章

多表操作

学习目标

- 掌握多表查询，能够使用交叉连接、内连接、外连接及复合条件连接进行多表查询；
- 掌握子查询，能够使用子查询结合IN、EXISTS、ANY、ALL及比较运算符进行查询；
- 掌握外键约束的使用，能够为表添加和删除外键约束；
- 了解关联表的3种关联关系，能够说出如何向关联表中添加和删除数据。

之前章节对数据的操作都是基于一张数据表完成的，即单表操作，然而实际应用中业务逻辑较为复杂，表与表之间可能存在业务联系，有时需要基于两张或两张以上的数据表进行操作，即多表操作。本章将针对多表操作的相关知识进行讲解。

5.1 多表查询

在关系数据库中，一张数据表通常存储一个实体的信息。当两张或多张数据表中存在相同意义的字段时，如果需要同时显示多张数据表中的数据，便可以通过这些意义相同的字段将不同的数据表进行连接并对连接后的数据表进行查询，这样的查询通常称为连接查询。在 MySQL 中，连接查询包括交叉连接查询、内连接查询、外连接查询、复合条件连接查询，本节将对这些连接查询进行讲解。

5.1.1 交叉连接查询

交叉连接（CROSS JOIN）查询返回的结果是被连接的两张数据表中所有数据行的笛卡儿积。例如，数据库 ems 中的部门表 dept 有 3 条部门记录，员工表 emp 有 14 条员工记录，如果对这两张数据表进行交叉连接查询，那么交叉连接查询后的笛卡儿积就

有 42（3 × 14）条记录。

交叉连接的语法格式如下。

```
SELECT <字段名> FROM <数据表名 1> CROSS JOIN <数据表名 2> ;
```

或

```
SELECT <字段名> FROM <数据表名 1>, <数据表名 2> ;
```

在上述语法格式中，两种语法格式的返回结果相同，其中 < 字段名 > 指的是需要查询的字段名称；< 数据表名 1> 和 < 数据表名 2> 指的是需要交叉连接的数据表的名称；CROSS JOIN 用于连接两个要查询的数据表，通过 CROSS JOIN 语句可以查询两个表中所有的数据组合。

下面通过一个案例演示交叉连接查询。

例如，技术人员想要通过 SQL 语句对数据库 ems 中员工表 emp 和部门表 dept 进行交叉连接查询，具体如下。

部门表 dept 用于存储部门信息，由于在第 2 章的讲解中将部门表删除了，因此查询之前需要先创建一个部门表并完善部门表中的数据，具体 SQL 语句如下。

```
# 创建部门表
CREATE TABLE dept(
   deptno INT PRIMARY KEY,
   dname VARCHAR(20) UNIQUE
);
# 插入部门数据
INSERT INTO dept
VALUES
(10, '总裁办'),
(20, '研究院'),
(30, '销售部'),
(40, '运营部');
```

接着对员工表和部门表进行交叉连接查询，具体 SQL 语句及执行结果如下。

```
mysql>SELECT * FROM emp,dept;
+------+-------+------+------+---------+--------+--------+-------+--------+
|empno |ename  |job   |mgr   |sal      |comm    |deptno  |deptno | dname  |
+------+-------+------+------+---------+--------+--------+-------+--------+
| 9369 | 张三  | 保洁 | 9902 | 900.00  |  NULL  |    20  |    30 | 销售部 |
| 9369 | 张三  | 保洁 | 9902 | 900.00  |  NULL  |    20  |    40 | 运营部 |
| 9369 | 张三  | 保洁 | 9902 | 900.00  |  NULL  |    20  |    20 | 研究院 |
```

```
| 9369 | 张三   | 保洁 | 9902 | 900.00  |  NULL  |    20 |     10 | 总裁办 |
| 9499 | 孙七   | 销售 | 9698 | 2600.00 | 300.00 |    30 |     30 | 销售部 |
| 9499 | 孙七   | 销售 | 9698 | 2600.00 | 300.00 |    30 |     40 | 运营部 |
| 9499 | 孙七   | 销售 | 9698 | 2600.00 | 300.00 |    30 |     20 | 研究院 |
| 9499 | 孙七   | 销售 | 9698 | 2600.00 | 300.00 |    30 |     10 | 总裁办 |
| 9521 | 周八   | 销售 | 9698 | 2250.00 | 500.00 |    30 |     30 | 销售部 |
| 9521 | 周八   | 销售 | 9698 | 2250.00 | 500.00 |    30 |     40 | 运营部 |
| 9521 | 周八   | 销售 | 9698 | 2250.00 | 500.00 |    30 |     20 | 研究院 |
| 9521 | 周八   | 销售 | 9698 | 2250.00 | 500.00 |    30 |     10 | 总裁办 |
...因篇幅有限，此处省略了其他的记录
+------+-------+------+------+---------+--------+--------+-------+--------+
48 rows in set (0.00 sec)
```

从上述执行结果可以看出，交叉连接查询的结果就是两个连接表中所有数据的组合，查询出的记录数为 48，即员工表 emp 的记录数 12 乘以部门表 dept 的记录数 4；查询出的字段数为 9，即员工表 emp 的字段数 7 加上部门表 dept 的字段数 2。由于交叉连接查询的结果中存在很多不合理的数据，因此在实际应用中应避免交叉连接查询，而是使用具体的条件对数据进行有目的的查询。

5.1.2 内连接查询

内连接（INNER JOIN）查询又称简单连接查询或自然连接查询，是常见的连接查询。内连接查询根据连接条件可以对交叉连接查询的部分结果进行筛选，仅筛选出两张表中相互匹配的记录。

内连接查询的语法格式如下。

```
SELECT 查询字段 FROM 数据表 1 [INNER] JOIN 数据表 2 ON 匹配条件 ;
```

在上述语法格式中，INNER JOIN 用于连接两张数据表，其中 INNER 可以省略；ON 用于指定查询的匹配条件，即同时匹配两张数据表的条件。由于内连接查询是对两张数据表进行操作，因此需要在匹配条件中指定所操作的字段来源于哪一张数据表，如果为数据表设置了别名，也可以通过别名指定数据表。

例如，技术人员想要通过 SQL 语句查询已经分配了部门（部门号不为 NULL）的员工的信息，员工信息只需要显示员工姓名和对应部门的名称，具体 SQL 语句及执行结果如下。

```
mysql>SELECT ename,dname FROM emp e JOIN dept d ON e.deptno=d.deptno;
+--------+--------+
| ename  | dname  |
+--------+--------+
| 张三   | 研究院 |
```

```
| 孙七    | 销售部  |
| 周八    | 销售部  |
| 李四    | 研究院  |
| 吴九    | 销售部  |
| 刘一    | 总裁办  |
| 郑十    | 销售部  |
| 萧十一  | 销售部  |
| 赵六    | 研究院  |
| 陈二    | 总裁办  |
| 王五    | 研究院  |
+--------+--------+
11 rows in set (0.00 sec)
```

在上述查询语句中，通过匹配员工表 emp 和部门表 dept 中的字段 deptno，使用内连接查询分配了部门的员工信息；由执行结果可知，查询出了员工姓名和对应部门的名称。

如果在一个连接查询中，涉及的两张数据表是同一张数据表，则这种查询称为自连接查询。自连接是一种特殊的内连接，它是指相互连接的数据表在物理上为同一张数据表，但逻辑上分为两张数据表。

例如，技术人员想要通过 SQL 语句查询员工王五所在部门的所有员工信息。查询时可以使用自连接查询实现，具体 SQL 语句及执行结果如下。

```
mysql>SELECT e1.* FROM emp e1 JOIN emp e2 ON e1.deptno=e2.deptno
    -> WHERE e2.ename='王五';
+-------+-------+--------+------+---------+------+--------+
| empno | ename | job    | mgr  | sal     | comm | deptno |
+-------+-------+--------+------+---------+------+--------+
|  9369 | 张三  | 保洁   | 9902 |  900.00 | NULL |     20 |
|  9566 | 李四  | 经理   | 9839 | 3995.00 | NULL |     20 |
|  9902 | 赵六  | 分析员 | 9566 | 4000.00 | NULL |     20 |
|  9988 | 王五  | 分析员 | 9566 | 4000.00 | NULL |     20 |
+-------+-------+--------+------+---------+------+--------+
4 rows in set (0.00 sec)
```

在上述查询语句中，名称为 e1 和名称为 e2 的数据表在物理上是同一张数据表 emp，e1 和 e2 通过字段 deptno 进行关联，并且通过 WHERE 指定筛选的条件；执行结果中，返回了王五所在部门的所有员工信息。由执行结果可知，王五所在部门有 4 个员工，分别是张三、李四、赵六和王五。

5.1.3 外连接查询

内连接的查询结果是符合连接条件的记录，然而有时在查询时，除了要查询出符合

条件的数据外，还需要查询出其中一张数据表中符合条件之外的其他数据，此时就需要使用外连接查询。

外连接查询的语法格式如下。

```
SELECT 所查字段 FROM 数据表 1 LEFT|RIGHT [OUTER] JOIN 数据表 2 ON 匹配条件
```

外连接查询分为左连接（LEFT JOIN）查询和右连接（RIGHT JOIN）查询，一般上述语法格式中的数据表 1 被称为左表，数据表 2 被称为右表。使用左连接查询和右连接查询的区别如下。

- LEFT JOIN：返回左表中的所有记录和右表中符合连接条件的记录。
- RIGHT JOIN：返回右表中的所有记录和左表中符合连接条件的记录。

为了让初学者更好地理解外连接查询，下面分别对左连接查询和右连接查询进行讲解。

1. 左连接查询

左连接查询的结果包括 LEFT JOIN 子句中左表的所有记录以及右表中满足连接条件的记录。如果左表的某条记录在右表中不存在，则右表中对应字段的值显示为 NULL。

例如，技术人员想要通过 SQL 语句查询所有部门名称及部门对应员工的姓名。因为需要查询出所有部门的名称，所以查询时可以使用左连接查询，将部门表作为查询中的左表，具体 SQL 语句及执行结果如下。

```
mysql>SELECT d.dname,e.ename FROM dept d LEFT JOIN emp e ON e.deptno=d.
deptno;
+--------+--------+
| dname  | ename  |
+--------+--------+
| 总裁办 | 陈二   |
| 总裁办 | 刘一   |
| 研究院 | 王五   |
| 研究院 | 赵六   |
| 研究院 | 李四   |
| 研究院 | 张三   |
| 运营部 | NULL   |
| 销售部 | 萧十一 |
| 销售部 | 郑十   |
| 销售部 | 吴九   |
| 销售部 | 周八   |
| 销售部 | 孙七   |
+--------+--------+
12 rows in set (0.00 sec)
```

在上述查询语句中，使用左连接将部门表和员工表通过 deptno 字段进行连接；由执行结果可知，上述查询语句返回了 12 条记录，其中返回了左表 dept 中 dname 字段所有的数据，运营部没有员工，对应的员工姓名字段显示为 NULL。

2. 右连接查询

右连接查询的结果包括 RIGHT JOIN 子句中右表的所有记录以及左表中满足连接条件的记录。如果右表的某条记录在左表中没有匹配，则左表中对应字段的值显示为 NULL。

例如，技术人员想要通过 SQL 语句查询所有员工姓名及对应部门的名称，没有分配部门的员工也需要查询出来。因为需要查询出所有员工的名称，所以查询时可以使用右连接查询，将员工表作为查询中的右表，具体 SQL 语句及执行结果如下。

```
mysql>SELECT d.dname,e.ename FROM dept d RIGHT JOIN emp e ON e.deptno=d.
deptno;
+--------+--------+
| dname  | ename  |
+--------+--------+
| 研究院 | 张三   |
| 销售部 | 孙七   |
| 销售部 | 周八   |
| 研究院 | 李四   |
| 销售部 | 吴九   |
| 总裁办 | 刘一   |
| 销售部 | 郑十   |
| 销售部 | 萧十一 |
| 研究院 | 赵六   |
| NULL   | 张%一  |
| 总裁办 | 陈二   |
| 研究院 | 王五   |
+--------+--------+
12 rows in set (0.00 sec)
```

上述查询语句使用右连接将 dept 表和 emp 表通过 deptno 字段进行连接；由执行结果可知，上述查询语句返回了 12 条记录，员工张 % 一没有划分部门，其对应部门的名称显示为 NULL。

5.1.4 复合条件连接查询

复合条件连接查询是指在连接查询的过程中通过添加过滤条件限制执行结果，使执行结果更精确。

例如，技术人员想要通过 SQL 语句查询所有员工信息，员工信息包含员工所在部门的名称，并且按员工的工资降序排序。在查询时，可以根据 deptno 字段使用左连接

将部门表和员工表进行关联查询，并且使用 ORDER BY 根据 sal 字段的值对查询结果进行排序，具体 SQL 语句及执行结果如下。

```
mysql>SELECT e.*,d.dname FROM emp e LEFT JOIN dept d ON e.deptno=d.deptno
    ->ORDER BY e.sal DESC;
+-------+--------+---------+------+---------+---------+--------+--------+
| empno | ename  | job     | mgr  | sal     | comm    | deptno | dname  |
+-------+--------+---------+------+---------+---------+--------+--------+
|  9839 | 刘一   | 董事长  | NULL | 6000.00 |    NULL |     10 | 总裁办 |
|  9902 | 赵六   | 分析员  | 9566 | 4000.00 |    NULL |     20 | 研究院 |
|  9988 | 王五   | 分析员  | 9566 | 4000.00 |    NULL |     20 | 研究院 |
|  9566 | 李四   | 经理    | 9839 | 3995.00 |    NULL |     20 | 研究院 |
|  9982 | 陈二   | 经理    | 9839 | 3450.00 |    NULL |     10 | 总裁办 |
|  9499 | 孙七   | 销售    | 9698 | 2600.00 |  300.00 |     30 | 销售部 |
|  9844 | 郑十   | 销售    | 9698 | 2500.00 |    0.00 |     30 | 销售部 |
|  9521 | 周八   | 销售    | 9698 | 2250.00 |  500.00 |     30 | 销售部 |
|  9654 | 吴九   | 销售    | 9698 | 2250.00 | 1400.00 |     30 | 销售部 |
|  9936 | 张%一  | 保洁    | 9982 | 1200.00 |    NULL |   NULL | NULL   |
|  9900 | 萧十一 | 保洁    | 9698 | 1050.00 |    NULL |     30 | 销售部 |
|  9369 | 张三   | 保洁    | 9902 |  900.00 |    NULL |     20 | 研究院 |
+-------+--------+---------+------+---------+---------+--------+--------+
12 rows in set (0.00 sec)
```

从执行结果可以得出，使用复合条件查询的结果更精确，符合实际需求。

5.2 子查询

子查询是指一个查询语句嵌套在另一个语句内部的查询，当某个语句执行所需的过滤条件是另一个 SELECT 语句的结果时，可以使用子查询。子查询通常在 WHERE 子句中结合操作符一起使用，操作符可以是 IN、EXISTS、ANY、ALL、比较运算符。本节将对结合这几种操作符的子查询进行讲解。

5.2.1 IN关键字结合子查询

IN 关键字结合子查询使用时，需要内层子查询语句返回的结果是一个数据列，这个数据列中的值供外层语句进行比较操作。

下面通过一个案例演示查询语句中 IN 关键字结合子查询的使用。

例如，技术人员想要通过 SQL 语句查询工资大于 2900 的员工所属部门。查询时可以先通过子查询返回工资大于 2900 的员工所在部门的编号，接着使用 IN 关键字根据部门编号查询部门信息，具体 SQL 语句及执行结果如下所示。

```
mysql> SELECT * FROM dept WHERE deptno IN(SELECT deptno FROM emp WHERE
sal>2900);
+--------+--------+
| deptno | dname  |
+--------+--------+
|     10 | 总裁办 |
|     20 | 研究院 |
+--------+--------+
2 rows in set (0.00 sec)
```

从执行结果可知，只有总裁办和研究院存在工资大于 2900 的员工。

外层 SELECT 语句使用 NOT IN 关键字结合子查询使用时，其作用正好和使用 IN 相反。例如，技术人员想要通过 SQL 语句查询工资小于 2900 的员工所在的部门信息，具体 SQL 语句及执行结果如下。

```
mysql>SELECT * FROM dept WHERE deptno NOT IN(SELECT deptno FROM emp WHERE
sal>2900);
+--------+--------+
| deptno | dname  |
+--------+--------+
|     40 | 运营部 |
|     30 | 销售部 |
+--------+--------+
2 rows in set (0.00 sec)
```

从上述执行结果可以得出，使用 NOT IN 的查询结果是数据表中使用 IN 查询到的结果之外的其他数据，只有运营部和销售部不存在工资大于 2900 的员工。

5.2.2 EXISTS关键字结合子查询

EXISTS 关键字用于判断子查询的结果集是否为空，若子查询的结果集不为空，返回 TRUE，否则返回 FALSE。使用 EXISTS 关键字结合子查询进行查询时，会先执行外层查询语句，再根据 EXISTS 关键字后面子查询的查询结果，判断是否保留外层语句查询出的记录。EXISTS 的判断结果为 TRUE 时，保留对应的记录，否则去除记录。

下面通过一个案例演示查询语句中 EXISTS 关键字结合子查询的使用。

例如，技术人员想要通过 SQL 语句查询工资大于 2900 的员工所在的部门信息。首先查询出部门的所有信息，然后通过子查询筛选出工资大于 2900 的员工信息，接着使用 EXISTS 关键字将符合子查询结果的记录返回；具体 SQL 语句及执行结果如下。

```
mysql>SELECT * FROM dept  WHERE EXISTS
    ->(SELECT * FROM emp WHERE emp.deptno=dept.deptno AND emp.sal>2900);
+--------+--------+
| deptno | dname  |
+--------+--------+
|     10 | 总裁办  |
|     20 | 研究院  |
+--------+--------+
2 rows in set (0.00 sec)
```

从上述执行结果可以得出，只有总裁办和研究院存在工资大于 2900 的员工。

使用 EXISTS 关键字结合子查询和使用 IN 关键字结合子查询的结果一致，但在表数据量不同时，这两种方式的性能也不同。当外表数据量比较大而内表数据量比较小时，适合使用 IN 关键字结合子查询进行查询；当外表数据量比较小而内表数据量比较大时，适合使用 EXISTS 关键字结合子查询进行查询。

5.2.3 ANY关键字结合子查询

ANY 关键字表示“任意一个”的意思，必须和比较操作符一起使用，例如 ANY 和 > 结合起来使用表示大于任意一个。ANY 关键字结合子查询使用时，表示子查询的查询结果集中的任一查询结果，例如“值 1>ANY（子查询）”比较值 1 是否大于子查询返回的结果集中的任意一个结果。

下面通过一个案例演示查询语句中 ANY 关键字结合子查询的使用。

例如，技术人员想要通过 SQL 语句查询部门编号为 10 的员工信息，要求查询到的员工信息中工资都高于部门编号为 20 的部门中的最低工资。查询时可以先使用子查询语句查询出部门编号为 20 的部门中所有员工工资，接着查询部门编号为 10 的部门中所有员工信息，最后使用 ANY 连接两者的工资进行比较。具体 SQL 语句及执行结果如下。

```
mysql>SELECT * FROM emp WHERE deptno=10 AND sal>ANY(SELECT sal FROM emp
WHERE deptno=20);
+-------+-------+--------+------+---------+------+--------+
| empno | ename | job    | mgr  | sal     | comm | deptno |
+-------+-------+--------+------+---------+------+--------+
|  9839 | 刘一  | 董事长 | NULL | 6000.00 | NULL |     10 |
|  9982 | 陈二  | 经理   | 9839 | 3450.00 | NULL |     10 |
+-------+-------+--------+------+---------+------+--------+
2 rows in set (0.00 sec)
```

5.2.4 ALL关键字结合子查询

ALL 关键字表示“所有”的意思，该关键字结合子查询使用时，表示子查询结果集中的所有结果，例如“值 1>ALL（子查询）”比较值 1 是否大于子查询返回的结果集中的所有结果。

下面通过一个案例演示查询语句中 ALL 关键字结合子查询的使用。

例如，技术人员想要通过 SQL 语句查询部门编号为 10 的员工信息，要求查询到的员工信息中工资都高于部门编号为 20 的部门中的最高工资。查询时可以使用子查询将部门编号为 20 的所有员工工资查询出来，然后将部门编号为 10 的所有员工工资与子查询的结果进行比较，只要大于子查询中的任意一个值，就是符合查询条件的记录。具体 SQL 语句及执行结果如下。

```
    mysql>SELECT * FROM emp WHERE deptno=10 AND sal>ALL(SELECT sal FROM
emp WHERE deptno=20);
    +-------+-------+--------+------+---------+------+--------+
    | empno | ename | job    | mgr  | sal     | comm | deptno |
    +-------+-------+--------+------+---------+------+--------+
    |  9839 | 刘一  | 董事长 | NULL | 6000.00 | NULL |     10 |
    +-------+-------+--------+------+---------+------+--------+
    1 row in set (0.00 sec)
```

5.2.5 比较运算符结合子查询

前面讲解的 ANY 关键字和 ALL 关键字的子查询中使用了比较运算符 >。除了 > 运算符，子查询中还可以使用其他的比较运算符，如 <、=、!= 等。

下面通过一个案例演示查询语句中比较运算符结合子查询的使用。

例如，技术人员想要通过 SQL 语句查询与王五职位相同的员工信息。查询时可以先使用子查询获取王五的职位，接着根据子查询的结果筛选出职位和王五相同的员工信息，具体 SQL 语句及执行结果如下。

```
    mysql> SELECT * FROM emp WHERE job=(SELECT job FROM emp WHERE ename='王五')
        -> AND ename !='王五';
    +-------+-------+--------+------+---------+------+--------+
    | empno | ename | job    | mgr  | sal     | comm | deptno |
    +-------+-------+--------+------+---------+------+--------+
    |  9902 | 赵六  | 分析员 | 9566 | 4000.00 | NULL |     20 |
    +-------+-------+--------+------+---------+------+--------+
    1 row in set (0.00 sec)
```

从执行结果可知，王五的职位是分析员，和王五职位相同的员工只有赵六。

一般情况下，表连接查询都可以用子查询替换，但反过来却不一定适用。子查询相