单元5 表记录的检索

表记录的检索是指从数据库中获取所需要的数据，又称数据查询。它是数据库操作中最常用，也是最重要的操作。用户可以根据自己对数据的需求，使用不同的查询方式，获得不同的数据。在 MySQL 中，使用 SELECT 语句实现数据查询。本单元将对查询语句的基本语法、在单表上查询数据、使用聚合函数查询数据、合并查询结果等内容进行详细讲解。

本单元主要学习目标如下：

- 熟练掌握查询简单数据记录的方法。
- 熟练掌握查询条件数据记录的方法。
- 熟练掌握查询分组数据的方法。
- 熟练掌握查询多表连接的方法。
- 熟练掌握子查询的方法。
- 能使用图形管理工具和命令方式实现数据的各类查询操作。

5.1 基本查询语句

查询是关系数据库中使用最频繁的操作，也是其他 SQL 语句的基础。例如，当要删除或更新某些数据记录时，首先需要查询这些记录，然后再对其进行相应的 SQL 操作。因此，基于 SELECT 的查询操作就显得十分重要，其基本语法格式如下。

```
SELECT [ALL|DISTINCT]要查询的内容
FROM 表名列表
[WHERE 条件表达式]
[GROUP BY 字段名列表[HAVING 逻辑表达式]]
[ORDER BY 字段名[ASC|DESC]]
[LIMIT [OFFSET,] n];
```

参数说明如下。

(1) SELECT 要查询的内容。“要查询的内容”可以是一个字段、多个字段，甚至是全部字段，还可以是表达式或函数。若要查询部分字段，需要将各字段名用逗号分隔开，各字段名在 SELECT 子句中的顺序决定了它们在结果中显示的顺序。用“*”表示返回所有字段。

(2) ALL | DISTINCT 用来标识在查询结果集中对相同行的处理方式。默认值为 ALL。

① 关键字 ALL 表示返回查询结果集中的所有行,包括重复行。

② 关键字 DISTINCT 表示若查询结果集中有相同的行,则只显示一行。

(3) FROM 表名列表指定用于查询的数据表的名称以及它们之间的逻辑关系。

(4) WHERE 条件表达式用于指定数据查询的条件。

(5) GROUP BY 字段名列表用来指定将查询结果根据什么字段分组。

(6) HAVING 逻辑表达式用来指定对分组的过滤条件,选择出满足查询条件的分组记录集。

(7) ORDER BY 字段名[ASC|DESC]用来指定查询结果集的排序方式。ASC 表示结果集按指定的字段以升序排列,DESC 表示结果集按指定的字段以降序排列。默认为 ASC。

(8) LIMIT [OFFSET,] n 用于限制查询结果的数量。LIMIT 后面可以跟两个参数,第一个参数"OFFSET"表示偏移量,如果偏移量为 0,则从查询结果的第一条记录开始显示,如果偏移量为 1,则从查询结果的第二条记录开始显示,以此类推。OFFSET 为可选值,如果不指定具体的值,则其默认值为 0。第二个参数"n"表示返回的查询记录的条数。

注意:

(1) 在上述语法结构中,SELECT 查询语句共有七个子句,其中 SELECT 和 FROM 子句为必选子句,而 WHERE、GROUP BY、ORDER BY 和 LIMIT 子句为可选子句,HAVING 子句与 GROUP BY 子句联合使用,不能单独使用。

(2) SELECT 子句既可以实现数据的简单查询、结果集的统计查询,也可以实现多表查询。

5.2　单表查询

5.2.1　简单数据记录查询

MySQL 通过 SELECT 语句实现数据记录的查询。简单数据记录查询的语法格式如下。

```
SELECT * | <字段列表>
FROM 数据表;
```

在上述查询语句中,星号"*"表示查询数据表的所有字段值,"字段列表"表示查询指定字段的字段值,数据表表示所要查询数据记录的表名。根据查询需求不同,该 SQL 语句可以通过以下两种方式使用。

(1) 查询所有字段数据。

(2) 查询指定字段数据。

【例 5-1】 查询 jwsystem 数据库的 studentinfo 表,输出所有学生的详细信息。

提示: 查询结果要输出表或视图的特定字段时,要明确指出字段名,多个字段名之间用

逗号分开。

(1) 查看数据表 studentinfo 的表结构,SQL 语句如下。

```
DESC studentinfo;
```

执行结果如图 5-1 所示。

(2) 首先选择数据表 studentinfo 所在的数据库,然后执行 SELECT 语句查询所有字段的数据,对应的 SQL 语句如下。

```
USE jwsystem;
SELECT sno,sname,sgender,sbirth,sclass
FROM studentinfo;
```

执行结果如图 5-2 所示。

信息 | 结果 1 | 剖析 | 状态

Field	Type	Null	Key	Default	Extra
sno	char(8)	NO	PRI	(Null)	
sname	varchar(10)	NO		(Null)	
sgender	char(2)	YES		(Null)	
sbirth	date	YES		(Null)	
sclass	varchar(20)	YES		(Null)	

图 5-1 数据表 studentinfo 的结构

信息 | 结果 1 | 剖析 | 状态

sno	sname	sgender	sbirth	sclass
200101	王敏	女	2000-01-02	JAVA2001
200102	董政辉	男	2000-01-02	JAVA2001
200103	陈苏	女	2000-01-02	JAVA2001
200104	汪一娟	女	2000-01-02	JAVA2001
200105	朱杰礼	男	2000-01-02	JAVA2001
200106	李文赞	男	1999-01-02	JAVA2001
200107	张鑫源	男	1999-01-02	JAVA2001

图 5-2 数据表 studentinfo 中所有字段记录

注意:

(1) 在 SELECT 子句的查询字段列表中,字段的顺序是可以改变的,无须按照表中定义的顺序排列。例如,上述语句可以写为:

```
SELECT sname,sno,sgender,sbirth,sclass
FROM studentinfo;
```

(2) 当要查询的内容是数据表中所有列的集合时,可以用符号“*”代表所有字段名的集合。例如,上述语句可以写为:

```
SELECT * FROM studentinfo;
```

执行结果和图 5-2 是一样的。

【例 5-2】 查询 jwsystem 数据库的 studentinfo 表,输出所有学生的学号和姓名。

提示: 要从表中选择部分字段进行输出,则需要在 SELECT 后面给出所选字段的字段名,各字段名之间用逗号隔开。查询结果集中字段显示的顺序与 SELECT 子句中给出的字段顺序相同。

(1) 首先选择数据表 studentinfo 所在的数据库,然后执行 SELECT 语句查询指定字段的数据,对应的 SQL 语句如下。

```
USE jwsystem;
SELECT sno,sname
FROM studentinfo;
```

执行结果如图 5-3 所示。

（2）调整 SELECT 后字段的排列顺序，对应的 SQL 语句如下。

```
SELECT sname ,sno
FROM studentinfo;
```

运行结果如图 5-4 所示。

sno	sname
200101	王敏
200102	董政辉
200103	陈莎
200104	汪一娟
200105	朱杰礼
200106	李文赞
200107	张鑫源

图 5-3　studentinfo 表中指定字段的数据记录

sname	sno
王敏	200101
董政辉	200102
陈莎	200103
汪一娟	200104
朱杰礼	200105
李文赞	200106
张鑫源	200107

图 5-4　调整指定字段顺序的查询结果

（3）如果指定字段在数据表中不存在，则查询报错。例如，在 studentinfo 数据表中查询字段名为 price 的数据，对应的 SQL 语句如下。

```
SELECT price
FROM studentinfo;
```

执行结果将报错，错误信息“1054-Unknowncolumn'price' in 'fieldlist'”提示不存在 price 字段。

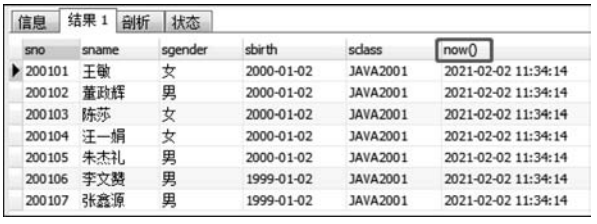
【例 5-3】 查询 jwsystem 数据库的 studentinfo 表，输出所有学生的详细信息，以及此次查询的日期和时间。

提示：可以使用 now()函数输出当前日期和时间。

对应的 SQL 语句如下。

```
SELECT *,now()
FROM studentinfo;
```

执行结果如图 5-5 所示。



sno	sname	sgender	sbirth	sclass	now()
200101	王敏	女	2000-01-02	JAVA2001	2021-02-02 11:34:14
200102	董政辉	男	2000-01-02	JAVA2001	2021-02-02 11:34:14
200103	陈莎	女	2000-01-02	JAVA2001	2021-02-02 11:34:14
200104	汪一娟	女	2000-01-02	JAVA2001	2021-02-02 11:34:14
200105	朱杰礼	男	2000-01-02	JAVA2001	2021-02-02 11:34:14
200106	李文赞	男	1999-01-02	JAVA2001	2021-02-02 11:34:14
200107	张鑫源	男	1999-01-02	JAVA2001	2021-02-02 11:34:14

图 5-5　显示查询记录的日期和时间

注意：使用 SELECT 语句进行查询时，查询结果集中字段的名称与 SELECT 子句中字段的名称相同。也可以在 SELECT 语句中，让查询结果集显示新的字段名，称为字段的别名。指定返回字段的别名有以下两种方法。

```
字段名 AS 别名
```

或

```
字段名 别名
```

【例 5-4】 查询 jwsystem 数据库的 studentinfo 表，输出所有学生的学号、姓名，以及此次查询的日期和时间，并分别使用“学号”“姓名”“查询日期”作为别名。对应的 SQL 语句如下。

```
SELECT sno 学号,sname AS 姓名, now() AS 查询日期
FROM studentinfo;
```

执行结果如图 5-6 所示。

信息 | 结果 1 | 剖析 | 状态

学号	姓名	查询日期
200101	王敏	2021-02-02 11:41:26
200102	董政辉	2021-02-02 11:41:26
200103	陈莎	2021-02-02 11:41:26
200104	汪一娟	2021-02-02 11:41:26
200105	朱杰礼	2021-02-02 11:41:26
200106	李文赞	2021-02-02 11:41:26
200107	张鑫源	2021-02-02 11:41:26

图 5-6 字段名以别名显示的查询结果

5.2.2 使用 DISTINCT 子句

当在 MySQL 中执行数据查询时，查询结果可能会包含重复的数据。如果需要消除这些重复数据，可以在 SELECT 语句中使用 DISTINCT 关键字。语法格式如下。

```
SELECT DISTINCT 字段名
FROM 表名;
```

【例 5-5】 查询 jwsystem 数据库的 studentinfo 表，输出学生所在的班级，每个班级只输出一次。

提示：使用 DISTINCT 关键字可以消除查询结果集中的重复行。否则，查询结果集中将包括所有满足条件的行。

(1) 首先选择数据表 studentinfo 所在的数据库，然后执行 SELECT 语句查询 sclass 字段的值，对应的 SQL 语句如下。

```
USE jwsystem;
SELECT sclass
FROM studentinfo;
```

(2) 上一步骤的返回结果中有重复数据，使用 DISTINCT 关键字消除重复数据，对应的 SQL 语句如下。

```
SELECT DISTINCT sclass
FROM studentinfo;
```

执行结果如图 5-7 所示。

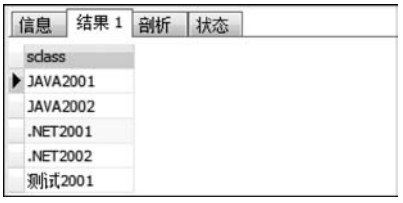


图 5-7　使用 DISTINCT 子句消除重复记录

注意：DISTINCT 关键字不能部分使用，一旦使用，将会应用于所有指定的字段，而不仅是某一个，也就是所有字段的组合值重复时才会被消除。

5.2.3　使用 WHERE 子句

WHERE 子句可以指定查询条件，用以从数据表中筛选出满足条件的数据行。其语法格式如下。

```
SELECT [ALL|DISTINCT] 要查询的内容
FROM 表名列表
WHERE 条件表达式;
```

WHERE 子句的条件表达式可以使用的运算符如表 5-1 所示。

表 5-1　条件表达式的运算符

运算符分类	运　算　符	说　　明
比较运算符	>、>=、=、<、<=、<>、!=、!>、!<	比较字段值的大小
范围运算符	BETWEEN…AND、NOT、BETWEEN…AND	判断字段值是否在指定范围内
列表运算符	IN、NOT IN	判断字段值是否在指定的列表中
模式匹配运算符	LIKE、NOT LIKE	判断字段值是否和指定的模式字符串匹配
空值判断运算符	IS NULL、IS NOT NULL	判断字段值是否为空
逻辑运算符	AND、OR、NOT	用于多个条件表达式的逻辑连接

1. 比较运算符的使用

【例 5-6】 查询 jwsystem 数据库的 studentinfo 表，输出 JAVA2001 班学生的详细信息。对应的 SQL 语句如下。

```
USE jwsystem;
SELECT *
FROM studentinfo
WHERE sclass = 'JAVA2001';
```

执行结果如图 5-8 所示。

信息 结果 1 剖析 状态

sno	sname	sgender	sbirth	sclass
200102	董政辉	男	2000-01-02	JAVA2001
200103	陈莎	女	2000-01-02	JAVA2001
200104	汪一娟	女	2000-01-02	JAVA2001
200105	朱杰礼	男	2000-01-02	JAVA2001
200106	李文赞	男	1999-01-02	JAVA2001
200107	张鑫源	男	1999-01-02	JAVA2001
200108	龚洁	女	1999-02-03	JAVA2001

图 5-8　班级为 JAVA2001 的学生详细信息

2. 范围运算符的使用

【例 5-7】 查询 jwsystem 数据库的 studentinfo 表，输出 1999 年出生的学生的详细信息。对应的 SQL 语句如下。

```
USE jwsystem;
SELECT *
FROM studentinfo
WHERE sbirth BETWEEN '1999-1-1' AND '1999-12-31';
```

执行结果如图 5-9 所示。

信息 结果 1 剖析 状态

sno	sname	sgender	sbirth	sclass
200106	李文赞	男	1999-01-02	JAVA2001
200107	张鑫源	男	1999-01-02	JAVA2001
200108	龚洁	女	1999-02-03	JAVA2001
200109	张刚玉	女	1999-02-08	JAVA2001
200112	李聪	男	1999-01-02	JAVA2001
200113	黄子加	男	1999-02-08	JAVA2001
200114	方涛	男	1999-02-08	JAVA2001

图 5-9　1999 年出生的学生的详细信息

注意：日期和时间类型是一个特殊的数据类型，它不仅可以作为一个连续的范围使用 BETWEEN…AND 运算符，还可以进行加、减以及比较大小操作。例 5-7 的 SQL 语句还可以写成如下形式：

```
USE jwsystem;
SELECT *
FROM studentinfo
WHERE sbirth >= '1999-1-1' AND sbirth <= '1999-12-31';
```

其执行结果和图 5-9 是一样的。

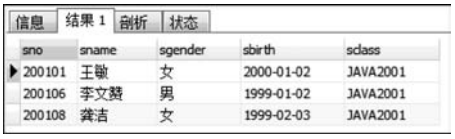
3. 列表运算符的使用

【例 5-8】 查询 jwsystem 数据库的 studentinfo 表，输出学号为 200101、200106、200108 的学生的详细信息。

对应的 SQL 语句如下。

```
USE jwsystem;
SELECT *
FROM studentinfo
WHERE sno IN ('200101', '200106', '200108');
```

执行结果如图 5-10 所示。



sno	sname	sgender	sbirth	sclass
200101	王敏	女	2000-01-02	JAVA2001
200106	李文赞	男	1999-01-02	JAVA2001
200108	龚洁	女	1999-02-03	JAVA2001

图 5-10　列表运算符使用的执行结果

4. 模式匹配运算符的使用

在指定的条件不是很明确的情况下，可以使用 LIKE 运算符与模式字符串进行匹配运算。其语法格式如下。

```
字段名 [NOT]  LIKE '模式字符串'
```

参数说明如下。

(1) 字段名：指明要进行匹配的字段。字段的数据类型可以是字符串类型或日期和时间类型。

(2) 模式字符串：可以是一般的字符串，也可以是包含通配符的字符串。通配符的种类如表 5-2 所示。

表 5-2　通配符的种类

通　配　符	含　　义
%	匹配任意长度(0 个或多个)的字符串
_	匹配任意单个字符

通配符和字符串必须括在单引号中。例如，表达式“LIKE 'a%'”匹配以字母 a 开头的字符串；表达式“LIKE '%101'”匹配以 101 结尾的字符串；表达式“LIKE'学%'”匹配第二个字符为“学”的字符串。

如果要查找的字符串本身就包括通配符，可以用符号“\”将通配符转义为普通字符。例如，表达式“LIKE 'A_'”表示要匹配的字符串长度为 2，且第一个字符为 A，第二个字符为“_”。

【例 5-9】 查询 jwsystem 数据库的 studentinfo 表，输出姓“张”的学生的详细信息。对应的 SQL 语句如下。

```
SELECT *
FROM studentinfo
WHERE sname LIKE '张%';
```

执行结果如图 5-11 所示。

【例 5-10】 查询 jwsystem 数据库的 studentinfo 表，输出姓“张”且名字长度为 2 的学生的详细信息。对应的 SQL 语句如下。

```
SELECT *
FROM studentinfo
WHERE sname LIKE '张_';
```

执行结果如图 5-12 所示。

信息 | 结果 1 | 剖析 | 状态

sno	sname	sgender	sbirth	sclass
200107	张鑫源	男	1999-01-02	JAVA2001
200109	张刚玉	女	1999-02-08	JAVA2001
200111	张凯	男	2000-01-02	JAVA2001
200137	张柳柳	女	2000-04-25	JAVA2002
200142	张忠凯	男	2000-04-25	.NET2001
200143	张仁荣	男	2000-03-25	.NET2001
200169	张晓琳	女	2000-08-20	.NET2002

图 5-11 studentinfo 表中姓“张”的学生的详细信息

信息 | 结果 1 | 剖析 | 状态

sno	sname	sgender	sbirth	sclass
200111	张凯	男	2000-01-02	JAVA2001
200194	张彤	女	2001-02-01	测试2001
200197	张璇	女	2000-01-02	测试2001
200201	张明	男	1998-08-05	JAVA2001

图 5-12 姓“张”且名字长度为 2 的学生详细信息

5. 空值判断运算符的使用

IS [NOT] NULL 运算符用于判断指定字段的值是否为空值。对于空值判断，不能使用比较运算符或模式匹配运算符。

【例 5-11】 查询 jwsystem 数据库的 teacher 表，输出性别为空的教师的信息。对应的 SQL 语句如下。

```
SELECT *
FROM teacher
WHERE tgender IS NULL;
```

执行结果如图 5-13 所示。

信息 | 结果 1 | 剖析 | 状态

tno	tname	tgender	tedu	tpro
1011	李杰	(Null)	博士研究生	副教授

图 5-13 性别为空的教师的信息

6. 逻辑运算符的使用

查询条件可以是一个条件表达式，也可以是多个条件表达式的组合。逻辑运算符能够连接多个条件表达式，构成一个复杂的查询条件。逻辑运算符包括 AND(逻辑与)、OR(逻辑或)、NOT(逻辑非)。

(1) AND 连接两个条件表达式。当且仅当两个条件表达式都成立时，组合起来的条件才成立。

(2) OR 连接两个条件表达式。两个条件表达式之一成立，组合起来的条件就成立。

(3) NOT 连接一个条件表达式。对给定条件取反。

【例 5-12】 查询 jwsystem 数据库的 studentinfo 表，输出姓“李”且是 JAVA2001 班的学生的信息。对应的 SQL 语句如下。

```
SELECT *
FROM studentinfo
WHERE sname LIKE '李%' AND sclass = 'JAVA2001';
```

执行结果如图 5-14 所示。

【例 5-13】 查询 jwsystem 数据库的 studentinfo 表，输出姓“李”或者是 JAVA2001 班的学生的信息。对应的 SQL 语句如下。

```
SELECT *
FROM studentinfo
WHERE sname LIKE '李%' OR sclass = 'JAVA2001';
```

执行结果如图 5-15 所示。

信息　结果 1　剖析　状态

sno	sname	sgender	sbirth	sclass
200106	李文赞	男	1999-01-02	JAVA2001
200112	李聪	男	1999-01-02	JAVA2001

图 5-14　姓“李”且是 JAVA2001 班的学生的信息

信息　结果 1　剖析　状态

sno	sname	sgender	sbirth	sclass
200120	杨明	男	2001-12-08	JAVA2001
200121	陈娟	女	2001-12-08	JAVA2001
200163	李玲	女	2000-07-20	.NET2002
200176	李立志	男	2000-09-20	.NET2002
200201	张明	男	1998-08-05	JAVA2001

图 5-15　姓“李”或者是 JAVA2001 班的学生的信息

注意：AND 运算符的优先级高于 OR 运算符，因此两个运算符一起使用时，应该先处理 AND 运算符两边的条件表达式，再处理 OR 运算符两边的条件表达式。

【例 5-14】 查询 jwsystem 数据库的 studentinfo 表，输出不是 2000 年出生的学生的信息。

提示：要得到指定日期类型数据的年份，可以使用函数 YEAR()。

对应的 SQL 语句如下。

```
SELECT *
FROM studentinfo
WHERE NOT(YEAR(sbirth) = 2000);
```

执行结果如图 5-16 所示。

信息　结果 1　剖析　状态

sno	sname	sgender	sbirth	sclass
200114	方涛	男	1999-02-08	JAVA2001
200115	彭攀	男	1999-02-08	JAVA2001
200116	鲁文达	男	1999-02-08	JAVA2001
200117	汪媛丽	女	2001-12-08	JAVA2001
200118	雷浩	男	2001-12-08	JAVA2001

图 5-16　不是 2000 年出生的学生的信息

5.2.4 使用 ORDER BY 子句

默认情况下，查询结果是按照数据记录最初添加到数据表中的顺序排序的。这样的查询结果顺序不能满足用户的需求，所以 MySQL 提供了 ORDER BY 关键字对查询结果进行排序。其语法格式如下。

```
SELECT [ALL|DISTINCT] 要查询的内容
FROM 表名列表
[WHERE 条件表达式]
ORDER BY 字段名[ASC|DESC] ;
```

参数说明如下。

(1) 可以规定数据行按升序排列(使用参数 ASC)，也可以规定数据行按降序排列(使用参数 DESC)，默认参数为 ASC。

(2) 可以在 ORDER BY 子句中指定多个字段，查询结果首先按照第一个字段的值排序，第一个字段的值相同的数据行，再按照第二个字段的值排序，以次类推。

(3) ORDER BY 子句要写在 WHERE 子句的后面。

【例 5-15】 查询 jwsystem 数据库的 elective 表，输出选修了 J001 号课程的学生信息，并将查询结果按成绩的降序排序。

对应的 SQL 语句如下。

```
SELECT *
FROM elective
WHERE cno = 'J001'
ORDER BY score DESC;
```

sno	cno	score
200132	J001	100
200158	J001	99
200141	J001	98
200145	J001	97
200144	J001	97
200107	J001	96

图 5-17 J001 号课程按成绩降序排序

执行结果如图 5-17 所示。

MySQL 中可以按照多个字段值的顺序对查询结果进行排序，字段之间须用逗号隔开。首先按照第一个字段的值排序，当字段值相同时，再按照第二个字段的值排序，以此类推。并且，每个字段都可以指定按照升序或者降序排序。例如，在 studentinfo 表中，将学生信息按学号升序和出生日期降序排序，SQL 语句如下。

```
SELECT *
FROM studentinfo
ORDER BY sno ASC,sbirth DESC;
```

5.2.5 使用 LIMIT 子句

当在 MySQL 中执行数据查询时，查询结果可能会包含很多数据。如果仅需要结果中的某些行数据，可以使用 LIMIT 关键字实现。其语法格式如下。

```
SELECT [ALL|DISTINCT] 要查询的内容
FROM 表名列表
[WHERE 条件表达式]
[ORDER BY 字段名 [ASC|DESC]]
LIMIT [OFFSET,] n;
```

LIMIT 子句接受一个或两个整数参数。其中，OFFSET 代表从第几行记录开始检索，n 代表检索多少行记录。需要注意的是，OFFSET 可以省略不写，默认取值为 0，代表从第一行记录开始检索。

【例 5-16】 查询 jwsystem 数据库的 studentinfo 表，输出前三条学生记录的信息。对应的 SQL 语句如下。

```
SELECT *
FROM studentinfo
LIMIT 3;
```

执行结果如图 5-18 所示。

【例 5-17】 查询 jwsystem 数据库的 studentinfo 表，输出表中第五行学生记录的信息。对应的 SQL 语句如下。

```
SELECT *
FROM studentinfo
LIMIT 4,1;
```

执行结果如图 5-19 所示。

信息　结果 1　剖析　状态

sno	sname	sgender	sbirth	sclass
200101	王敏	女	2000-01-02	JAVA2001
200102	董政辉	男	2000-01-02	JAVA2001
200103	陈莎	女	2000-01-02	JAVA2001

图 5-18　输出前三条学生记录的信息

信息　结果 1　剖析　状态

sno	sname	sgender	sbirth	sclass
200105	朱杰礼	男	2000-01-02	JAVA2001

图 5-19　输出表中第五行学生记录的信息

5.3　统计查询

MySQL 提供了一些对数据进行分析的统计函数，因为有时我们需要的并不是某些具体数据，而是对数据的统计分析结果。例如，统计某个班级的总人数、某个部门的平均薪资等。本节将介绍这些统计函数的作用和使用方法。

5.3.1　集合函数

集合函数用于对查询结果集中的指定字段进行统计，并输出统计值。常用的集合函数如表 5-3 所示。

表 5-3 集合函数

集合函数	功能描述
COUNT([DISTINCT|ALL]字段|*)	计算指定字段中值的个数。COUNT(*)返回满足条件的行数,包括含有空值的行,不能与 DISTINCT 一起使用
SUM([DISTINCT|ALL]字段)	计算指定字段中数据的总和(此字段为数值类型)
AVG([DISTINCT|ALL]字段)	计算指定字段中数据的平均值(此字段为数值类型)
MAX([DISTINCT|ALL]字段)	计算指定字段中数据的最大值
MIN([DISTINCT|ALL]字段)	计算指定字段中数据的最小值

表 5-3 中,ALL 为默认选项,表示计算所有的值; DISTINCT 选项则表示去掉重复值后再计算。

【例 5-18】 查询 jwsystem 数据库的 studentinfo 表,统计学生总人数。

提示: 统计学生总人数,就是统计学生表中的数据的行数。

对应的 SQL 语句如下。

```
SELECT COUNT(*) AS 学生总人数
FROM studentinfo;
```

执行结果如图 5-20 所示。

【例 5-19】 查询 jwsystem 数据库的 elective 表,统计选修了 J001 号课程的学生人数、总成绩、平均分、最高分和最低分。对应的 SQL 语句如下。

```
SELECT COUNT(*) AS 学生人数,SUM(score) AS 总成绩,
AVG(score) 平均分,MAX(score) 最高分,MIN(score) 最低分
FROM elective
WHERE cno = 'J001';
```

执行结果如图 5-21 所示。

图 5-20 统计学生总人数

图 5-21 J001 号课程的学生人数、总成绩、平均分、最高分和最低分

5.3.2 分组数据查询

在对表中数据进行统计时,可能需要按照一定的类别进行统计,例如,统计每一类书籍的在库总册数。这时我们首先需要对书籍按类别进行分组,然后统计每一组书籍的在库册数总和。在 MySQL 中,通过 GROUP BY 关键字按照某个字段或者多个字段的值对数据进行分组,字段值相同的数据记录为一组。其语法格式如下。

```
SELECT [ALL|DISTINCT] 要查询的内容
FROM 表名列表
[WHERE 条件表达式]
GROUP BY 字段名列表 [HAVING 条件表达式];
```

注意：使用 GROUP BY 子句进行分组统计时，SELECT 子句要查询的字段只能是以下两种情况。

（1）字段应用了集合函数。

（2）未应用集合函数的字段必须包含在 GROUP BY 子句中。

1. 字段分组查询

如果 GROUP BY 关键字后只有一个字段，则数据将按该字段的值进行分组，具体示例如下。

【例 5-20】 查询 studentinfo 表，分别统计男女生人数。对应的 SQL 语句如下。

```
SELECT sgender,COUNT( * ) AS 人数
FROM studentinfo
GROUP BY sgender;
```

执行结果如图 5-22 所示。

【例 5-21】 查询 elective 表，统计并输出每个学生所选课程数目及平均分。对应的 SQL 语句如下。

```
SELECT sno,COUNT(cno) AS 选修课程数目,AVG(score) AS 平均分
FROM elective
GROUP BY sno;
```

执行结果如图 5-23 所示。

信息　结果 1　剖析　状态

sgender	人数
女	37
男	63

图 5-22　分别统计男女生人数

信息　结果 1　剖析　状态

sno	选修课程数目	平均分
200101	3	85.3333
200102	3	62.3333
200103	3	66.6667
200104	3	75.3333
200105	3	68.3333

图 5-23　统计每个学生所选课程数目及平均分

【例 5-22】 查询 elective 表，统计并输出每门课程选课人数、最高分、最低分和平均分。对应的 SQL 语句如下。

```
SELECT cno, COUNT(sno) AS 选课人数,MAX(score) AS 最高分,
MIN(score) AS 最低分,AVG(score) 平均分
FROM elective
GROUP BY cno;
```

执行结果如图 5-24 所示。

cno	选课人数	最高分	最低分	平均分
J001	99	100	50	75.5051
Z001	40	99	51	70.2500
Z002	41	99	51	70.0732
Z003	58	100	52	77.7069
Z004	59	96	51	73.9153

图 5-24　每门课程选课人数、最高分、最低分和平均分

使用 GROUP BY 关键字还可以对多个字段按层次进行分组。首先按第一个字段分组，然后在第一个字段值相同的每个分组中再根据第二个字段值进行分组。

2. HAVING 子句限定分组查询

HAVING 关键字和 WHERE 关键字都用于设置条件表达式，两者的区别在于，HAVING 关键字后可以有集合函数，而 WHERE 关键字不能。WHERE 子句的作用是在对查询结果进行分组前，将不符合 WHERE 条件子句的行去掉，即在分组之前过滤数据。HAVING 子句的作用是筛选满足条件的组，即在分组之后过滤数据。

注意：HAVING 子句常和 GROUP BY 子句配合使用。HAVING 子句用于对分组后的结果进行条件筛选，HAVING 子句只能出现在 GROUP BY 子句后。

当一个语句中同时出现了 WHERE 子句、GROUP BY 子句和 HAVING 子句时，执行顺序如下。

(1) 执行 WHERE 子句，从数据表中选取满足条件的数据行。

(2) 由 GROUP BY 子句对选取的数据行进行分组。

(3) 执行集合函数。

(4) 执行 HAVING 子句，选取满足条件的分组。

【例 5-23】 查询 elective 表中每门课成绩都在 70～90 分内的学生的学号。对应的 SQL 语句如下。

```
SELECT sno AS 每门成绩都在 70—90 之间的学生
FROM elective
GROUP BY sno
HAVING MIN(score)>=70 AND MAX(score)<=90;
```

执行结果如图 5-25 所示。

【例 5-24】 查询至少选修了三门课程的学生的学号。对应的 SQL 语句如下。

```
SELECT sno,count(*) 选修课程数
FROM elective
GROUP BY sno
HAVING count(*)>=3;
```

执行结果如图 5-26 所示。

每门成绩都在70—90之间的学生
200104
200148
200168
200173
200183

图 5-25　每门课成绩都在 70～90 分内的学生的学号

sno	选修课程数
200101	3
200102	3
200103	3
200104	3
200105	3

图 5-26　至少选修了三门课程的学生的学号

5.4　多表查询

在实际查询中，很多情况下用户需要的数据并不全在一个表中，而是存在于多个不同的表中，这时就要使用多表查询。多表查询是通过各个表之间的共同列的相关性来查询数据的。多表查询首先要在这些表中建立连接，再在连接生成的结果集基础上进行筛选。

多表查询语法格式如下。

```
SELECT [表名.]目标字段表达式[AS 别名],…
FROM 左表名[AS 别名] 连接类型 右表名[AS 别名]
ON 连接条件
[WHERE 条件表达式];
```

其中，连接类型以及运算符有以下几种。

(1) CROSS JOIN：交叉连接。

(2) INNER JOIN 或 JOIN：内连接。

(3) LEFT JOIN 或 LEFT OUTER JOIN：左外连接。

(4) RIGHT JOIN 或 RIGHT OUTER JOIN：右外连接。

(5) FULL JOIN 或 FULL OUTER JOIN：完全连接。

5.4.1　交叉连接

交叉连接就是将要连接的两个表的所有行进行组合，也就是将第一个表的所有行分别与第二个表的每个行连接形成一个新的行。连接后生成的结果集的行数等于两个表的行数的乘积，字段个数等于两个表的字段个数的和。其语法格式如下。

```
SELECT 字段名列表
FROM 表名 1 CROSS JOIN 表名 2;
```

图 5-27 中的表 R 和表 S 进行交叉连接的结果集如图 5-28 所示。

R

A	B	C
1	2	3
4	5	6

S

A	D
1	2
3	4
5	6

图 5-27　示例表 R 和 S 的数据

R CROSS JOIN S

A	B	C	A	D
1	2	3	1	2
1	2	3	3	4
1	2	3	5	6
4	5	6	1	2
4	5	6	3	4
4	5	6	5	6

图 5-28　表 R 和表 S 交叉连接的结果集

注意：交叉连接的结果集称为笛卡儿积，笛卡儿积在实际应用中一般是没有任何意义的。

【例 5-25】 对 course 表和 teacher 表进行交叉连接，观察交叉连接后的结果集。对应的 SQL 语句如下。

```
SELECT *
FROM course CROSS JOIN teacher;
```

执行结果如图 5-29 所示。

信息 | 结果 1 | 剖析 | 状态

cno	cname	cperiod	credit	ctno	tno	tname	tgender	tedu	tpro
Z004	数据结构	64	4.0	1010	1001	杜倩颖	女	本科	讲师
Z003	数据库	64	4.0	1008	1001	杜倩颖	女	本科	讲师
Z002	C#程序设计	84	6.0	1009	1001	杜倩颖	女	本科	讲师
Z001	JAVA程序设计	84	6.0	1002	1001	杜倩颖	女	本科	讲师
J001	公共英语	36	2.0	1001	1001	杜倩颖	女	本科	讲师
Z004	数据结构	64	4.0	1010	1002	赵复前	男	本科	讲师
Z003	数据库	64	4.0	1008	1002	赵复前	男	本科	讲师
Z002	C#程序设计	84	6.0	1009	1002	赵复前	男	本科	讲师
Z001	JAVA程序设计	84	6.0	1002	1002	赵复前	男	本科	讲师
J001	公共英语	36	2.0	1001	1002	赵复前	男	本科	讲师
Z004	数据结构	64	4.0	1010	1003	胡祥	男	硕士研究生	副教授

图 5-29　course 表和 teacher 表交叉连接的结果集

course 表是 5 行 5 列的表，teacher 表是 13 行 5 列的表。这两个表进行交叉连接形成的就是 65 行 10 列的表。但可以看出来，这张表是没有实际意义的。

5.4.2　内连接

内连接又称简单连接或自然连接，是一种常见的关系运算。内连接使用条件运算符对两个表中的数据进行比较，并将符合连接条件的数据记录组合成新的数据记录。

内连接有以下两种语法格式。

```
SELECT 字段名列表
FROM 表名 1 [INNER] JOIN 表名 2
ON 表名 1.字段名 比较运算符 表名 2.字段名;
```

或者

```
SELECT 字段名列表
FROM 表名 1, 表名 2
WHERE 表名 1.字段名 比较运算符 表名 2.字段名;
```

内连接包括三种类型：等值连接、非等值连接和自然连接。

(1) 等值连接：在连接条件中使用等号(=)比较运算符来比较连接字段的值，其查询结果中包含被连接表的所有字段，包括重复字段。在等值连接中，两个表的连接条件通常采

用“表1.主键字段＝表2.外键字段”的形式。

（2）非等值连接：在连接条件中使用了除等号之外的比较运算符(>、<、>=、<=、!=)来比较连接字段的值。

（3）自然连接：与等值连接相同，都是在连接条件中使用比较运算符，但结果集不包括重复字段。图5-27中表R和表S进行等值连接、非等值连接和自然连接的结果集如图5-30所示。

R JOIN S(R.A=S.A)

R.A	B	C	S.A	D
1	2	3	1	2

(等值连接)

R JOIN S(R.A=S.A)

R.A	B	C	S.A	D
4	5	6	1	2
4	5	6	3	4

(非等值连接)

R JOIN S

R.A	B	C	D
1	2	3	2

(自然连接)

图5-30　表R和表S内连接的结果集

在上述语法格式中，如果要输出的字段是表1和表2都有的字段，则必须在输出的字段名前加上表名进行区分，用“表名.字段名”表示。如果表名太长，可以给表名定义一个简短的别名，这样在SELECT语句的输出字段名和连接条件中，用到表名的地方都可以用别名来代替。

【例5-26】 查询jwsystem数据库，输出考试成绩不及格学生的学号、姓名、课程号和成绩。

提示：需要输出四个字段作为查询结果，在jwsystem数据库中，没有一个表包含这四个字段，因此需要多表连接查询。多表连接查询首先确定需要哪几个表进行连接查询。进行连接查询的表要能够包含输出的所有字段，并且保证用到表的数量最少。进行连接的表之间要有含义相同的字段。

本例中，在studentinfo表和elective表中有“学号”字段，在studentinfo表中有“姓名”字段，在course表和elective表中有“课程号”字段，在elective表中有“成绩”字段。由此可知，要输出指定的四个字段，最少需要studentinfo表和elective表。这两个表可以进行连接的共同字段为“学号”。

对应的SQL语句如下。

```
SELECT s.sno,sname,cno,score
FROM studentinfo AS s JOIN elective AS e ON s.sno = e.sno
WHERE score < 60;
```

或者

```
SELECT s.sno,sname,cno,score
FROM studentinfo AS s,elective AS e
WHERE s.sno = e.sno AND score < 60;
```

执行结果如图5-31所示。

信息 | 结果 1 | 剖析 | 状态

sno	sname	cno	score
200102	董政辉	Z001	52
200103	陈莎	J001	54
200103	陈莎	Z001	58
200105	朱杰礼	Z001	57
200106	李文赞	Z001	58

图 5-31　成绩不及格学生的学号、姓名、课程号和成绩

【例 5-27】 查询 jwsystem 数据库，输出考试成绩不及格学生的学号、姓名、课程名和成绩。

提示：完成本查询需要用到三张表：studentinfo 表、course 表和 elective 表。这三张表的连接查询是通过表的两两连接来实现的。elective 表和 studentinfo 表有相同的“学号”字段，elective 表和 course 表有相同的“课程名”字段，所以 elective 表作为中间表，可以先和 studentinfo 表连接，再和 course 表连接。当然，这个连接顺序并不是固定的，elective 表也可以先和 course 表连接，再和 studentinfo 表连接。

对应的 SQL 语句如下。

```
SELECT s.sno,sname,cname,score
FROM studentinfo AS s JOIN elective AS e ON s.sno = e.sno
JOIN course AS c ON c.cname = e.cname
WHERE score < 60;
```

或者

```
SELECT s.sno,sname,cname,score
FROM studentinfo AS s,elective AS e,course AS c
WHERE s.sno = e.sno AND e.cname = c.cname AND score < 60;
```

执行结果如图 5-32 所示。

信息 | 结果 1 | 剖析 | 状态

sno	sname	cname	score
200178	乐冬杰	C#程序设计	51
200200	周航	C#程序设计	51
200102	董政辉	JAVA程序设计	52
200103	陈莎	JAVA程序设计	58
200105	朱杰礼	JAVA程序设计	57
200106	李文赞	JAVA程序设计	58

图 5-32　成绩不及格学生的学号、姓名、课程名和成绩

5.4.3　外连接

内连接查询中返回的查询结果只包含符合查询条件和连接条件的数据，然而，有时还需要包含左表（左外连接）或右表（右外连接）中的所有数据，此时就需要使用外连接查询。使用外连接时，以主表中每行数据去匹配从表中的数据行，如果符合连接条件，则返回到结果

集中；如果没有找到匹配的数据行，则在结果集中仍然保留主表的数据行，相对应的从表中的字段则被填上 NULL 值。

外连接的语法格式如下。

```
SELECT 字段名列表
FROM 表名 1 LEFT|RIGHT JOIN 表名 2
ON 表名 1.字段名 比较运算符 表名 2.字段名;
```

外连接包括三种类型：左外连接、右外连接和全外连接。

(1) 左外连接：即左表为主表，连接关键字为 LEFT JOIN。将左表中的所有数据行与右表中的每行按连接条件进行匹配，结果集中包括左表中所有的数据行。左表中与右表没有相匹配记录的行，在结果集中对应的右表字段都以 NULL 来填充。BIT 类型不允许为 NULL，就以 0 填充。

(2) 右外连接：即右表为主表，连接关键字为 RIGHT JOIN。将右表中的所有数据行与左表中的每行按连接条件进行匹配，结果集中包括右表中所有的数据行。右表中与左表没有相匹配记录的行，在结果集中对应的左表字段都以 NULL 来填充。

(3) 全外连接：连接关键字为 FULL JOIN。查询结果集中包括两个连接表的所有的数据行，若左表中每一行在右表中有匹配数据，则结果集中对应的右表的字段填入相应数据，否则填充为 NULL；若右表中某一行在左表中没有匹配数据，则结果集对应的左表字段填充为 NULL。

图 5-27 中表 R 和表 S 进行外连接的结果集如图 5-33 所示。

R LEFT JOIN S(R.A=S.A)

R.A	B	C	S.A	D
1	2	3	1	2
4	5	6	NULL	NULL

R RIGHT JOIN S(R.A=S.A)

R.A	B	C	S.A	D
1	2	3	1	2
NULL	NULL	NULL	3	4
NULL	NULL	NULL	5	6

R FULL JOIN S(R.A=S.A)

R.A	B	C	S.A	D
1	2	3	1	2
4	5	6	NULL	NULL
NULL	NULL	NULL	3	4
NULL	NULL	NULL	5	6

图 5-33　表 R 和表 S 外连接的结果集

注意：外连接查询只适用于两个表。

【例 5-28】 查询 jwsystem 数据库，输出所有教师教授的课程信息，没有教授课程的教师也要列出。

提示：要输出所有教师的授课信息，说明需要 teacher 表和 course 表。没有授课的教师也要列出，说明 teacher 表是主表。

对应的 SQL 语句如下。

```
SELECT *
FROM teacher AS t LEFT JOIN course AS c ON t.tno = c.ctno;
```

执行结果如图 5-34 所示。

信息 | 结果 1 | 剖析 | 状态

tno	tname	tgender	tedu	tpro	cno	cname	cperiod	credit	ctno
1008	黄阳	男	博士研究生	讲师	Z003	数据库	64	4.0	1008
1009	胡冬	男	博士研究生	讲师	Z002	C#程序设计	84	6.0	1009
1010	许杰	男	博士研究生	教授	Z004	数据结构	64	4.0	1010
1011	李杰	(Null)	博士研究生	副教授	(Null)	(Null)	(Null)	(Null)	(Null)
1012	李连杰	男	硕士研究生	讲师	(Null)	(Null)	(Null)	(Null)	(Null)

图 5-34　教师教授的课程信息

5.4.4　自连接

自连接就是一个表的两个副本之间的内连接，即同一个表名在 FROM 子句中出现两次，故为了区别，必须对表指定不同的别名，字段名前也要加上表的别名进行限定。

【例 5-29】 查询和学号为 200102 的学生在同一个班级的学生的学号和姓名。对应的 SQL 语句如下。

```
SELECT s2.sno,s2.sname
FROM studentinfo AS s1 JOIN studentinfo AS s2
ON s1.sclass = s2.sclass
WHERE s1.sno = '200102' AND s2.sno!= '200102';
```

执行结果如图 5-35 所示。

信息 | 结果 1 | 剖析 | 状态

sno	sname
200101	王敏
200103	陈莎
200104	汪一娟

图 5-35　与学号为 200102 的学生在同一个班级的学生的学号和姓名

5.5　子查询

子查询是指一个查询语句嵌套在另一个查询语句内部的查询，即在一个 SELECT 查询语句的 WHERE 或 FROM 子句中包含另一个 SELECT 查询语句。其中，外层 SELECT 查询语句称为主查询，WHERE 或 FROM 子句中的 SELECT 查询语句称为子查询。执行查询语句时，首先会执行子查询中的语句，然后将查询结果作为外层查询的过滤条件。子查询中常用的操作符有 ANY(SOME)、ALL、IN、EXISTS。本节将详细介绍如何在 SELECT 语句中嵌套子查询。

5.5.1　带比较运算符的子查询

子查询可以使用比较运算符，这些运算符包括＝、!＝、>、<、>＝、<＝和<>，其中!＝和<>是等价的。比较运算符在子查询中应用非常广泛，下面用具体示例说明子查询中比较运算符的使用方法。

【例 5-30】 查询 jwsystem 数据库，输出选修了“数据库”这门课的所有学生的学号和成绩。

提示：先用子查询查找出“数据库”这门课的课程号，再用主查询查找出课程号等于子查询找到的课程号的那些数据行，输出其学号和成绩。

对应的 SQL 语句如下。

```
SELECT sno,score AS 数据库的成绩
FROM elective
WHERE cno = (SELECT cno FROM course WHERE cname = '数据库');
```

执行结果如图 5-36 所示。

【例 5-31】 查询 jwsystem 数据库，输出年龄最大的学生的姓名。

提示：在 jwsystem 数据库的“学生”表中，只有学生的“出生日期”字段。要查找年龄最大的学生信息，先用子查询查找“出生日期”最小值，再用外查询查找出“出生日期”等于子查询找到的“出生日期”的数据行，并输出“姓名”字段。

对应的 SQL 语句如下。

```
SELECT sname AS 年龄最大的学生
FROM studentinfo
WHERE sbirth = (SELECT MIN(sbirth) FROM studentinfo);
```

执行结果如图 5-37 所示。

信息 | 结果 1 | 剖析 | 状态

sno	数据库的成绩
200101	96
200102	70
200103	88

图 5-36　例 5-30 执行结果

信息 | 结果 1 | 剖析 | 状态

年龄最大的学生
张明

图 5-37　年龄最大的学生的姓名

【例 5-32】 查询 jwsystem 数据库，输出“数据库”这门课不及格的学生的姓名。

提示：先用子查询从“课程”表中查找出“数据库”这门课的课程号，再用外查询从“成绩”表中查找出课程号等于子查询找到的课程号且成绩小于 60 分的数据行，得到这个数据行的学号值，再用外查询从“学生”表中查找学号等于子查询找到的那个学号的学生的姓名。

对应的 SQL 语句如下。

```
SELECT sname AS 数据库不及格的学生
FROM studentinfo
WHERE sno = (SELECT sno FROM elective WHERE score < 60 AND cno = (SELECT cno FROM course WHERE
cname = '数据库'));
```

执行结果如图 5-38 所示。

信息 | 结果 1 | 剖析 | 状态

数据库不及格的学生
张刚玉

图 5-38　“数据库”这门课不及格的学生姓名

注意：例 5-32 用到了子查询的多层嵌套。外层查询和子查询用比较运算符连接，这就要求每一层子查询得到的值最多只能有一个，也

就是说，最多只能有一个学生的“数据库”课成绩不及格。如果有两个或两个以上学生的“数据库”课成绩不及格，这个查询就要出错。

在实际情况中，不止一个学生在同一门课上成绩不及格的情况是普遍存在的。假如有多个学生的“数据库”课成绩不及格，该怎么书写 SQL 语句呢？对于子查询可能返回给主查询多个数值的情况，就要在主查询和子查询之间使用谓词 IN 或 NOT IN 进行连接。

5.5.2 IN 子查询

当主查询的条件是子查询的查询结果时，就可以通过 IN 关键字进行判断。相反，如果主查询的条件不是子查询的查询结果时，就可以通过 NOT IN 关键字实现。下面通过一个具体示例加以说明。

【例 5-33】 查询 jwsystem 数据库，输出考试不及格的学生的姓名。

提示：先用子查询在“选课”表中查找出成绩小于 60 分的学生的学号，查找到的学号可能是多个，是一个集合。再用主查询从“学生”表中查找出学号等于子查询找到的某个学号的学生的姓名。

对应的 SQL 语句如下。

```
SELECT sname AS 考试不及格的学生
FROM studentinfo
WHERE sno IN (SELECT sno FROM elective WHERE score < 60);
```

执行结果如图 5-39 所示。

注意：如果例 5-33 改为要查询考试成绩全部及格的学生的姓名，可把 IN 改为 NOT IN。对应的 SQL 语句如下。

```
SELECT sname AS 考试全及格的学生
FROM studentinfo
WHERE sno NOT IN (SELECT sno FROM elective WHERE score < 60);
```

执行结果如图 5-40 所示。

图 5-39 考试不及格的学生的姓名

图 5-40 NOT IN 实现查询考试成绩全部及格的学生的姓名

5.5.3 批量比较子查询

批量比较子查询是指子查询的结果不止一个，主查询和子查询之间需要用比较运算符进行连接。这时候，就需要在子查询前面加上谓词 ALL 或 ANY。

1. 使用 ANY 谓词

ANY 关键字表示主查询需要满足子查询结果的任一条件。使用 ANY 关键字时，只要满足子查询结果中的任意一条，就可以通过该条件执行外层查询语句。ANY 关键字通常与比较运算符一起使用，> ANY 表示大于子查询结果记录中的最小值；＝ANY 表示等于子查询结果记录中的任何一个值；< ANY 表示小于子查询结果记录中的最大值。

【例 5-34】 查询 jwsystem 数据库，输出需要补考的学生姓名。

对应的 SQL 语句如下。

```
SELECT sname AS 补考学生
FROM studentinfo
WHERE sno = ANY(SELECT sno FROM elective WHERE score < 60);
```

执行结果如图 5-41 所示。

2. 使用 ALL 谓词

ALL 关键字表示主查询需要满足所有子查询结果的所有条件。使用 ALL 关键字时，只有满足子查询语句返回的所有结果，才能执行外层查询语句。该关键字通常有两种使用方式：一种是> ALL，表示大于子查询结果记录中的最大值；另一种是< ALL，表示小于子查询结果记录中的最小值。

【例 5-35】 查询 jwsystem 数据库，输出不需要补考的学生的姓名。

对应的 SQL 语句如下。

```
SELECT sname AS 不需补考的学生
FROM studentinfo
WHERE sno != ALL(SELECT sno FROM elective WHERE score < 60);
```

查询结果如图 5-42 所示。

图 5-41　需要补考的学生姓名

图 5-42　不需要补考的学生的姓名

5.5.4　EXISTS 子查询

EXISTS 关键字返回一个布尔类型的结果。如果子查询结果至少能够返回一行记录，则 EXISTS 的结果为 true，此时主查询语句将被执行；反之，如果子查询结果没有返回任何一行记录，则 EXISTS 的结果为 false，此时主查询语句将不被执行。下面用一个具体示例说明 EXISTS 的使用方法。

【例 5-36】 查看 jwsystem 数据库的 teacher 表，若不存在具有教授职称的教师，则显示所有教师的姓名和职称。

对应的 SQL 语句如下。

```
SELECT tname,tpro
FROM teacher
WHERE NOT EXISTS (SELECT * FROM teacher WHERE tpro = '教授');
```

执行结果如图 5-43 所示。

【例 5-37】 查询 jwsystem 数据库的 elective 表，如果有需要补考的学生，就显示所有学生的成绩信息。如果没有需要补考的学生，就不输出任何信息。

对应的 SQL 语句如下。

```
SELECT *
FROM elective
WHERE EXISTS (SELECT * FROM elective WHERE score < 60);
```

执行结果如图 5-44 所示。

图 5-43 NOT EXISTS 子查询执行结果

图 5-44 EXISTS 子查询执行结果

NOT EXISTS 与 EXISTS 的作用相反，如果子查询至少返回一行记录，则 NOT EXISTS 的结果为 false，此时主查询语句将不被执行。反之，如果子查询不返回任何记录，则 NOT EXISTS 的结果为 true，此时主查询将被执行。

注意：子查询和连接查询在很多情况下可以互换。

(1) 对于例 5-29 查询和学号为 200102 的学生在同一个班级的学生学号和姓名的问题，可以用子查询来实现，对应的 SQL 语句如下。

```
SELECT sno,sname
FROM studentinfo
WHERE sno!= '200102' AND sclass = (SELECT sclass FROM studentinfo
WHERE sno = '200102');
```

(2) 对于例 5-34 查询 jwsystem 数据库，输出需要补考的学生姓名的问题，也可以用连接查询来实现，对应的 SQL 语句如下。

```
SELECT sname
FROM studentinfo AS s JOIN elective AS e ON s.sno = e.sno
WHERE score < 60;
```

至于什么时候使用连接查询，什么时候使用子查询，可以参考以下原则。

（1）当查询语句要输出的字段来自多个表时，用连接查询。

（2）当查询语句要输出的字段来自一个表，但其 WHERE 子句涉及另一个表时，常用子查询。

（3）当查询语句要输出的字段和 WHERE 子句都只涉及一个表，但是 WHERE 子句的查询条件涉及应用集合函数进行数值比较时，一般用子查询。

5.5.5　在增、删、改语句中使用子查询

1. 在 INSERT 语句中使用子查询

使用 INSERT…SELECT 语句可以将 SELECT 语句的查询结果添加到表中，一次可以添加多行。语法格式如下。

```
INSERT 表 1[(字段名列表 1)]
SELECT 字段名列表 2 FROM 表 2 [WHERE 条件表达式]
```

注意：使用本语句时，表 1 已经存在，且“字段名列表 1”中字段的个数、字段的顺序、字段的数据类型必须和“字段名列表 2”中对应的字段信息一样或兼容。

【例 5-38】 建立一个 Java 编程方向学生的信息表 studs，表里有学号、姓名、所在班级等字段，把 jwsystem 数据库中的 studentinfo 表中查询到的 Java 编程方向的学生的相关信息添加到本表中。

（1）建立 studs 表。

对应的 SQL 语句如下。

```
CREATE TABLE studs
(
  sno CHAR(8),
  sname VARCHAR(10),
  sclass VARCHAR(20)
);
```

图 5-45　建立 studs 表的执行结果

执行结果如图 5-45 所示。

（2）将 studentinfo 表中 Java 编程方向的学生信息插入 studs 表中。

对应的 SQL 语句如下。

```
INSERT INTO studs(sno,sname,sclass)
SELECT sno,sname,sclass FROM studentinfo WHERE sclass LIKE 'JAVA%';
```

执行结果如图 5-46 所示。

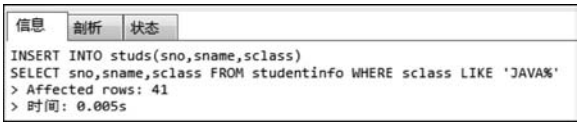


图 5-46　插入数据的执行结果

(3) 查询 studs 表中的数据。

对应的 SQL 语句如下。

```
SELECT * FROM studs;
```

执行结果如图 5-47 所示。

2. 在 UPDATE 语句中使用子查询

使用 UPDATE 语句时,可以在 WHERE 子句中使用子查询。

【例 5-39】 修改 jwsystem 数据库的 course 表,把职称为"副教授"的教师教授课程的学时减少 6 个。

对应的 SQL 语句如下。

```
UPDATE course SET cperiod = cperiod - 6
WHERE ctno IN (SELECT tno FROM teacher WHERE tpro = '副教授');
```

执行结果如图 5-48 所示。

信息 | 结果 1 | 剖析 | 状态

sno	sname	sclass
200101	王敏	JAVA2001
200102	董政辉	JAVA2001
200103	陈莎	JAVA2001

图 5-47 查询数据的执行结果

信息 | 剖析 | 状态

```
UPDATE course SET cperiod=cperiod-6
WHERE ctno IN (SELECT tno FROM teacher WHERE tpro='副教授')
> Affected rows: 1
> 时间: 0.004s
```

图 5-48 在 UPDATE 语句中使用子查询执行结果

3. 在 DELETE 语句中使用子查询

使用 DELETE 语句时,可以在 WHERE 子句中使用子查询。

【例 5-40】 将 elective 表中李聪的选课信息删除。

对应的 SQL 语句如下。

```
DELETE FROM elective
WHERE sno = (SELECT sno FROM studentinfo WHERE sname = '李聪');
```

执行结果如图 5-49 所示。

信息 | 剖析 | 状态

```
DELETE FROM elective
WHERE sno=(SELECT sno FROM studentinfo WHERE sname='李聪')
> Affected rows: 3
> 时间: 0.01s
```

图 5-49 在 DELETE 语句中使用子查询执行结果

5.6 合并查询结果

合并查询结果时将多条 SELECT 语句的查询结果合并到一起组合成单个结果集。进行合并操作时,两个结果集对应的列数和数据类型必须相同。每个 SELECT 语句之间使用

UNION 或 UNION ALL 关键字分隔。使用 UNION 时，需要注意以下四点。

（1）所有 SELECT 语句中的字段个数必须相同。

（2）所有 SELECT 语句中对应的字段的数据类型必须相同或兼容。

（3）合并后的结果集中的字段名是第一个 SELECT 语句中各字段的字段名。如果要为返回的字段指定别名，则必须在第一个 SELECT 语句中指定。

（4）使用 UNION 运算符合并结果集时，每一个 SELECT 语句本身不能包含 ORDER BY 子句，只能在最后使用一个 ORDER BY 子句对整个结果集进行排序，且在该 ORDER BY 子句中必须使用第一个 SELECT 语句中的字段名。

【例 5-41】 对 jwsystem 数据库进行查询，输出所有学生和教师的编号和姓名。

对应的 SQL 语句如下。

```
SELECT sno AS 编号,sname AS 姓名 FROM studentinfo
UNION
SELECT tno AS 编号,tname AS 姓名 FROM teacher;
```

执行结果如图 5-50 所示。

信息　结果 1　剖析　状态

编号	姓名
200199	董海霞
200200	周航
200201	张明
1001	杜倩颖
1002	赵复前

图 5-50　输出所有学生和教师的编号和姓名

单元小结

本单元主要介绍了 MySQL 软件对数据表进行查询的操作，具体包括单表查询、使用统计函数查询、分组查询、连接查询、子查询和合并查询结果等。本单元对单表查询，详细讲解了简单数据查询操作，使用 DISTINCT 关键字去除重复查询记录，限制查询结果数量，使用 ORDER BY 关键字对查询结果排序以及对条件数据查询；对使用统计函数查询，详细介绍了统计函数的作用和带统计功能的查询；对分组查询，详细介绍了单字段分组查询和多字段分组查询，以及带 HAVING 子句限定的分组查询；对连接查询，详细介绍了内连接和外连接查询；对子查询，详细介绍了带 IN、EXISTS、ANY、ALL 关键字，以及带比较运算符的子查询。最后，本单元详细介绍了如何使用 UNION 关键字合并多个查询结果。通过本单元的学习，读者能掌握各类数据查询的方法。

单元实训项目

项目一：在“网上书店”数据库中进行简单查询

目的：

掌握 SELECT 语句中 DISTINCT 子句、LIMIT 子句、WHERE 子句以及 ORDER BY 子句的使用。

内容：

（1）查询会员表。输出积分高于 500 分的会员昵称和联系电话。

(2) 查询会员表。输出积分低于200分的会员昵称和联系电话,分别用英文username、telephone指定别名。

(3) 查询会员表。输出E-mail是QQ邮箱的会员昵称及其E-mail。

(4) 查询订购表。输出订购日期是2016年10月的订单的详细信息。

(5) 查询订购表。输出订货的会员编号,要求删除重复行。

(6) 查询图书表。输出图书的名称和价格,并把查询结果按价格降序排列。

(7) 查询图书表。输出价格最高的三种图书的名称和价格。

项目二:在“网上书店”数据库查询中使用集合函数

目的:

掌握集合函数、GROUP BY子句、HAVING子句。

内容:

(1) 查询图书表。输出所有图书的最高价格、最低价格和平均价格。

(2) 查询图书表。输出每一类图书的数量。

(3) 查询图书表。输出每一类图书的最高价格、最低价格和平均价格。

(4) 查询订购表。输出订购数量超过3本的会员编号和订购数量。

项目三:在“网上书店”数据库查询中使用连接查询和子查询

目的:

掌握连接查询和子查询。

内容:

(1) 输出所有图书的图书名称、价格以及所属类别名称。

(2) 输出订购了《平凡的世界》的会员昵称、联系电话和订购数量。

(3) 输出订购了图书的会员昵称和联系电话。

(4) 输出无人订购的图书名称和价格。

(5) 输出详细订购信息,包括订购图书的会员昵称、联系电话、所订图书名称、数量、价格和折扣价。

单元练习题

一、选择题

1. 数据查询语句SELECT由多个子句构成,(　　)子句能够将查询结果按照指定字段的值进行分组。

A. ORDER BY　　B. LIMIT　　C. GROUP BY　　D. DISTINCT

2. WHERE子句用于指定(　　)。

A. 查询结果的分组条件　　B. 查询结果的统计方式

C. 查询结果的排序条件　　D. 查询结果的搜索条件

3. 要在“网上书店”数据库的“图书”表中查找图书名称包含“中国”两字的图书信息，可使用(　　)。

A. SELECT * FROM 图书 WHERE 图书名称 LIKE'中国%'

B. SELECT * FROM 图书 WHERE 图书名称 LIKE'%中国%'

C. SELECT * FROM 图书 WHERE 图书名称 LIKE'%中国'

D. SELECT * FROM 图书 WHERE 图书名称 LIKE'_中国%'

4. 在子查询语句中，下面子句(　　)用于将查询结果存储在另一张表中。

A. GROUP BY　　B. INSERT

C. WHERE　　D. DISTINCT

5. 集合函数(　　)可对指定字段求平均值。

A. SUM　　B. AVG　　C. MIN　　D. MAX

6. 对于“网上书店”数据库，以下 SELECT 语句的含义是(　　)。

```
SELECT 会员昵称 FROM 会员
WHERE 会员编号 NOT IN(SELECT 会员编号 FROM 订购)
```

A. 查询输出没有订购图书的会员昵称

B. 查询输出订购图书的会员昵称

C. 查询输出所有会员昵称

D. 查询输出没有编号的会员昵称

7. 子查询的结果不止一个值时，可以使用的运算符是(　　)。

A. IN　　B. LIKE　　C. =　　D. >

8. EXISTS 子查询的返回值是(　　)。

A. 数值类型　　B. 字符串类型　　C. 日期和时间类型　D. 逻辑类型

9. 执行以下 SQL 语句：

```
SELECT 学号,姓名 FROM 学生 LIMIT 2,2;
```

查询结果(　　)。

A. 返回了两行数据，分别是第 1 行和第 2 行数据

B. 返回了两行数据，分别是第 2 行和第 3 行数据

C. 返回了两行数据，分别是第 3 行和第 4 行数据

D. 返回了两行数据，分别是第 4 行和第 5 行数据

10. 输出 jwsystem 数据库中学生的成绩，在输出时把每个学生每门课程成绩都提高 10%，使用的 SQL 语句是(　　)。

A. SELECT sno,cno,score * 10 FROM elective

B. SELECT sno,cno,score+10 FROM elective

C. SELECT sno,cno,score * 0.1 FROM elective

D. SELECT sno,cno,score * 1.1 FROM elective

二、判断题

1. 在 MySQL 中，目前查询表中的记录只能使用 SELECT 语句。　　(　　)

2. 使用GROUP BY 实现分组时，可以指定多个分组字段进行分组，当多个字段取值都相同时就认为是同一组。（　　）

3. SELECT 语句中可以使用 AS 关键字指定表名的别名或字段的别名，AS 关键字也可以省略不写。（　　）

4. 在字段进行升序排列时，如果某条记录的字段值为 NULL，则这条记录会在最后一条显示。（　　）

三、简答题

1. 简述 MySQL 中通配符的类型以及它们各自的作用。

2. 简述 HAVING 关键字和 WHERE 关键字的区别（至少写两点）。

3. 现有一张表 score 记录所有学生数学和英语的成绩，表中字段有学号、姓名、学科和分数。要求如下：

（1）查询姓名为张三的学生成绩。

（2）查询英语成绩大于 90 分的同学。

（3）查询总分大于 180 分的所有同学的学号。