第5章

数据查询

本章要点

- 数据查询概述
- 简单查询
- 连接查询
- 集合查询
- 子查询
- 排名函数的使用
- 使用正则表达式查询

由于数据库查询是数据库的核心操作，本章重点讲述使用 PL/SQL 的 SELECT 查询语句对数据库进行各种查询的方法。

5.1 数据查询概述

PL/SQL 中最重要的部分是它的查询功能，PL/SQL 的 SELECT 语句具有灵活的使用方式和强大的功能，能够实现选择、投影和连接等操作。

语法格式：

```
SELECT <列>                                    /* SELECT 子句，指定列 */
    FROM  <表或视图>                           /* FROM 子句，指定表或视图 */
    [ WHERE  <条件表达式>]                     /* WHERE 子句，指定行 */
    [ GROUP BY <分组表达式>]                   /* GROUP BY 子句，指定分组表达式 */
    [ HAVING <分组条件表达式>]                 /* HAVING 子句，指定分组统计条件 */
    [ ORDER BY <排序表达式>[ ASC | DESC ]]     /* ORDER 子句，指定排序表达式和顺序 */
```

SQL(Structured Query Language)是目前主流的关系型数据库执行数据操作、数据检索及数据库维护所需要的标准语言，是用户与数据库之间进行交流的接口，许多关系型数据库管理系统都支持 SQL，但不同的数据库管理系统之间的 SQL 不能完全通用，Oracle 数据库使用的 SQL 是 Procedural Language/SQL(简称 PL/SQL)。

5.2 简单查询

5.2.1 投影查询

投影查询用于选择列,投影查询通过 SELECT 语句的 SELECT 子句来表示。

语法格式:

```
SELECT [ ALL | DISTINCT ] <列名列表>
```

其中,<列名列表>指出了查询结果的形式,其格式为:

```
{ *                                  /* 选择当前表或视图的所有列 */
  |<表名>|<视图>|.*                   /* 选择指定的表或视图的所有列 */
  |{|<列名>|<表达式>}
      [[AS] <列别名>]                 /* 选择指定的列,为列指定别名 */
  | <列标题>=<列名表达式>             /* 选择指定的列并更改列标题,为列指定别名 */
}[,… n ]
```

1. 投影指定的列

使用 SELECT 语句可选择表中的一个列或多个列,如果是多个列,各列名中间要用逗号分开。

语法格式:

```
SELECT <列名 1>[ , <列名 2>[,…n] ]
   FROM <表名>
   [WHERE <条件表达式>]
```

该语句的功能为在 FROM 子句指定表中检索符合条件的列。

【例 5.1】 查询 student 表中所有学生的学号、姓名和出生日期。

代码如下:

```
SELECT sid, sname, sbirthday
FROM student;
```

查询结果:

```
SID       SNAME   SBIRTHDAY
--------- ------- -----------
221001    何德明  2001-07-16
221002    王丽    2002-09-21
221004    田桂芳  2002-12-05
224001    周思远  2001-03-18
224002    许月琴  2002-06-23
224003    孙俊松  2001-10-07
```

2. 投影全部列

在 SELECT 子句指定列的位置上使用 * 号时，则为查询表中所有列。

【例 5.2】　查询 student 表中所有列。

代码如下：

```
SELECT *
FROM student;
```

该语句与下面语句等价：

```
SELECT sid, sname, ssex, sbirthday, speciality, tc
FROM student;
```

查询结果：

```
SID       SNAME   SSEX  SBIRTHDAY    SPECIALITY        TC
--------- ------- ----- ------------ ------------- ------
221001    何德明  男    2001-07-16   计算机            52
221002    王丽    女    2002-09-21   计算机            50
221004    田桂芳  女    2002-12-05   计算机            52
224001    周思远  男    2001-03-18   通信              52
224002    许月琴  女    2002-06-23   通信              48
224003    孙俊松  男    2001-10-07   通信              50
```

3. 修改查询结果的列标题

为了改变查询结果中显示的列标题，可以在列名后使用 AS ＜列别名＞。

【例 5.3】　查询 student 表中所有学生的 sid、sname、speciality，并将结果中各列的标题分别修改为学号、姓名、专业。

代码如下：

```
SELECT sid AS 学号, sname AS 姓名, speciality AS 专业
FROM student;
```

查询结果：

```
学号      姓名      专业
--------- ------- ---------
221001    何德明    计算机
221002    王丽      计算机
221004    田桂芳    计算机
224001    周思远    通信
224002    许月琴    通信
224003    孙俊松    通信
```

4. 计算列值

使用 SELECT 子句对列进行查询时，可以对数字类型的列进行计算，可以使用加(＋)、减(－)、乘(＊)、除(/)等算术运算符，SELECT 子句可使用表达式。

语法格式：

```
SELECT <表达式>[ , <表达式>]
```

5. 去掉重复行

如果要去掉结果集中的重复行，可使用 DISTINCT 关键字。

语法格式：

```
SELECT DISTINCT<列名>[ , <列名>…]
```

【例 5.4】 查询 student 表中 speciality 列，消除结果中的重复行。

```
SELECT DISTINCT speciality
FROM student;
```

查询结果：

```
SPECIALITY
-------------
计算机
通信
```

5.2.2 选择查询

投影查询用于选择行，选择查询通过 WHERE 子句实现，WHERE 子句通过条件表达式给出查询条件，该子句必须紧跟 FROM 子句之后。

语法格式：

```
WHERE<条件表达式>
```

其中，<条件表达式>为查询条件，格式为：

```
<条件表达式>::=
    { [ NOT ] <判定运算> | (<条件表达式>) }
    [ { AND | OR } [ NOT ] { <判定运算> | (<条件表达式>) } ]
    [ ,…n ]
```

其中，<判定运算>的结果为 TRUE、FALSE 或 UNKNOWN，其格式为：

```
<判定运算>::=
{   <表达式 1>{ = | < | <= | > | >= | <> | != } <表达式 2>          /* 比较运算 */
    | <字符串表达式 1>[ NOT ] LIKE <字符串表达式 2>[ ESCAPE '<转义字符>' ]
                                                                  /* 字符串模式匹配 */
    | <表达式>[ NOT ] BETWEEN <表达式 1>AND <表达式 2>             /* 指定范围 */
    | <表达式>IS [ NOT ] NULL                                     /* 是否空值判断 */
    | <表达式>[ NOT ] IN ( <子查询> | <表达式>[,…n] )             /* IN 子句 */
    | EXISTS(<子查询>)                                            /* EXISTS 子查询 */
}
```

说明：

(1) 判定运算包括比较运算、模式匹配、指定范围、空值判断、子查询等，判定运算的结果为 TRUE、FALSE 或 UNKNOWN。

(2) 逻辑运算符包括 AND(与)、OR(或)、NOT(非)，NOT、AND 和 OR 的使用是有优先级的，三者之中，NOT 优先级最高，AND 次之，OR 优先级最低。

(3) 条件表达式可以使用多个判定运算通过逻辑运算符构成复杂的查询条件。

(4) 字符串和日期必须用单引号括起来。

注意： 在 SQL 中，返回逻辑值的运算符或关键字都称为谓词。

1. 表达式比较

比较运算符用于比较两个表达式值，共有 7 个运算符：=(等于)、<(小于)、<=(小于或等于)、>(大于)、>=(大于或等于)、<>(不等于)、!=(不等于)。

语法格式：

```
<表达式 1>{ =| <| <=| >| >=| <>| !=} <表达式 2>
```

【例 5.5】 查询 student 表中专业为计算机或性别为女的学生。

代码如下：

```
SELECT *
FROM student
WHERE speciality='计算机' OR ssex='女';
```

查询结果：

```
SID       SNAME   SSEX  SBIRTHDAY    SPECIALITY        TC
--------- ------- ----- ------------ ------------- ------
221001    何德明  男    2001-07-16   计算机            52
221002    王丽    女    2002-09-21   计算机            50
221004    田桂芳  女    2002-12-05   计算机            52
224002    许月琴  女    2002-06-23   通信              48
```

2. 指定范围

BETWEEN、NOT BETWEEN、IN 是用于指定范围的 3 个关键字，用于查找字段值在(或不在)指定范围的行。

当要查询的条件是某个值的范围时，可以使用 BETWEEN 关键字。BETWEEN 关键字用于指出查询范围。

语法格式：

```
<表达式>[ NOT ] BETWEEN <表达式 1>AND <表达式 2>
```

【例 5.6】 查询 score 表成绩为 85、92、94 的记录。

代码如下：

```
SELECT *
```

```
FROM score
WHERE grade IN (85,92,94);
```

查询结果：

```
SID      CID       GRADE
-------- ----- -----------
221001   1004          94
221004   1201          92
224001   4002          92
221004   8001          85
```

3. 模式匹配

模式匹配使用 LIKE 谓词，LIKE 谓词用于指出一个字符串是否与指定的字符串相匹配，其运算对象可以是 char、varchar2 和 date 类型的数据，返回逻辑值 TRUE 或 FALSE。

语法格式：

```
<字符串表达式 1>[ NOT ] LIKE <字符串表达式 2>[ ESCAPE '<转义字符>' ]
```

在使用 LIKE 谓词时，<字符串表达式 2>可以含有通配符，通配符有以下两种。

- %：代表 0 或多个字符。
- _：代表 1 个字符。

LIKE 匹配中使用通配符的查询也称模糊查询。

【例 5.7】 查询 student 表中姓孙的学生情况。

代码如下：

```
SELECT *
FROM student
WHERE sname LIKE '孙%';
```

查询结果：

```
SID      SNAME   SSEX  SBIRTHDAY    SPECIALITY        TC
-------- ------- ----- ------------ -------------  ------
224003   孙俊松  男    2001-10-07   通信               50
```

4. 空值判断

判定一个表达式的值是否为空值时，可以使用 IS NULL 关键字。

语法格式：

```
<表达式>IS [ NOT ] NULL
```

【例 5.8】 查询已选课但未参加考试的学生情况。

代码如下：

```
SELECT *
FROM score
```

```
WHERE grade IS NULL;
```

查询结果：

```
SID      CID       GRADE
-------- ----- -----------
224002   8001
```

5.2.3　分组查询和统计计算

查询数据常常需要进行统计计算，本节介绍使用聚合函数、GROUP BY 子句、HAVING 子句进行统计计算的方法。

1. 聚合函数

聚合函数可以实现数据的统计计算，用于计算表中的数据，返回单个计算结果。聚合函数包括 COUNT、SUM、AVG、MAX、MIN 等函数，下面分别介绍。

1）COUNT 函数

COUNT 函数用于计算组中满足条件的行数或总行数。

语法格式：

```
COUNT( { [ ALL | DISTINCT ] <表达式>} | * )
```

其中，ALL 表示对所有值进行计算，ALL 为默认值，DISTINCT 指去掉重复值；选择 * 时将统计总行数。COUNT 函数用于计算时忽略 NULL 值。

【例 5.9】 求学生的总人数。

代码如下：

```
SELECT COUNT( * ) AS 总人数
FROM student;
```

该语句采用 COUNT 函数计算总行数，总人数与总行数一致。

查询结果：

```
    总人数
-----------
          6
```

2）SUM 和 AVG 函数

SUM 函数用于求出一组数据的总和，AVG 函数用于求出一组数据的平均值，这两个函数只能针对数值类型的数据。

语法格式：

```
SUM / AVG([ ALL | DISTINCT ] <表达式>)
```

其中，ALL 表示对所有值进行计算，为默认值，DISTINCT 指去掉重复值，SUM/AVG 函数用于计算时可忽略 NULL 值。

【例 5.10】 查询 1201 课程总分。

代码如下：

```
SELECT SUM(grade) AS 课程 1201 总分
FROM score
WHERE cid='1201';
```

该语句采用 SUM()计算课程总分，并用 WHERE 子句指定的条件进行限定为 1201 课程。

查询结果：

```
  课程 1201 总分
----------------
           509
```

3) MAX 和 MIN 函数

MAX 函数用于求出一组数据的最大值，MIN 函数用于求出一组数据的最小值，这两个函数都可以适用于任意类型数据。

语法格式：

```
MAX / MIN([ ALL | DISTINCT ] <表达式>)
```

其中，ALL 表示对所有值进行计算，ALL 为默认值，DISTINCT 指去掉重复值，MAX / MIN 函数用于计算时可忽略 NULL 值。

2. GROUP BY 子句

GROUP BY 子句用于指定需要分组的列。

语法格式：

```
GROUP BY [ ALL ] <分组表达式>[,…n]
```

其中，分组表达式通常包含字段名，ALL 显示所有分组。

注意：如果 SELECT 子句的列名表包含聚合函数，则该列名表只能包含聚合函数指定的列名和 GROUP BY 子句指定的列名。聚合函数常与 GROUP BY 子句一起使用。

【例 5.11】 查询各门课程的最高分、最低分、平均成绩。

代码如下：

```
SELECT cid AS 课程号, MAX(grade) AS 最高分, MIN(grade) AS 最低分, AVG(grade) AS 平均
成绩
FROM score
WHERE NOT grade IS NULL
GROUP BY cid;
```

该语句采用 MAX、MIN、AVG 等聚合函数，并用 GROUP BY 子句对 cid(课程号)进行分组。

查询结果：

```
课程号      最高分     最低分          平均成绩
--------  --------  ----------  ------------
8001          93        85            88.8
1201          93        75       84.8333333
1004          94        86              90
4002          92        78       86.3333333
```

3. HAVING 子句

HAVING 子句用于对分组按指定条件进一步进行筛选，过滤出满足指定条件的分组。

语法格式：

```
[ HAVING <条件表达式>]
```

其中，条件表达式为筛选条件，可以使用聚合函数。

注意：HAVING 子句可以使用聚合函数，WHERE 子句不可以使用聚合函数。

当 WHERE 子句、GROUP BY 子句、HAVING 子句、ORDER BY 子句在一个 SELECT 语句中时，执行顺序如下。

(1) 执行 WHERE 子句，在表中选择行。

(2) 执行 GROUP BY 子句，对选取行进行分组。

(3) 执行聚合函数。

(4) 执行 HAVING 子句，筛选满足条件的分组。

(5) 执行 ORDER BY 子句，进行排序。

注意：HAVING 子句要放在 GROUP BY 子句的后面，ORDER BY 子句放在 HAVING 子句后面。

【例 5.12】 查询至少有 5 名学生选修且以 8 开头的课程号和平均分数。

代码如下：

```
SELECT cid AS 课程号, AVG(grade) AS 平均分数
FROM score
WHERE cid LIKE '8%'
GROUP BY cid
HAVING COUNT(*)>5;
```

该语句采用 AVG 聚合函数、WHERE 子句、GROUP BY 子句、HAVING 子句。

查询结果：

```
课程号        平均分数
-------   ----------
8001            88.8
```

5.2.4　排序查询

在 Oracle 数据库中，ORDER BY 子句用于对查询结果进行排序。

语法格式：

```
[ ORDER BY { <排序表达式>[ ASC | DESC ] } [ ,…n ]
```

其中，排序表达式可以是列名、表达式或一个正整数，ASC 表示升序排列，它是系统默认排序方式，DESC 表示降序排列。

提示

排序操作可对数值、日期、字符 3 种数据类型使用，ORDER BY 子句只能出现在整个 SELECT 语句的最后。

【例 5.13】 将通信专业的学生按出生日期降序排序。

代码如下：

```
SELECT *
FROM student
WHERE speciality='通信'
ORDER BY sbirthday DESC;
```

该语句采用 ORDER BY 子句进行排序。

查询结果：

```
SID       SNAME   SSEX  SBIRTHDAY    SPECIALITY        TC
--------- ------- ----- ------------ -------------  ------
224002    许月琴  女    2002-06-23   通信               48
224003    孙俊松  男    2001-10-07   通信               50
224001    周思远  男    2001-03-18   通信               52
```

5.3 连接查询

在关系数据库管理系统中，经常把一个实体的信息存储在一个表里，当查询相关数据时，通过连接运算就可以查询存放在多个表中不同实体的信息，把多个表按照一定的关系连接起来，在用户看来好像是查询一个表一样。连接是关系数据库模型的主要特征，也是区别于其他类型数据库管理系统的一个标志。

在 PL/SQL 中，连接查询有两大类表示形式：一类是使用连接谓词指定的连接，另一类是使用 JOIN 关键字指定的连接。

5.3.1 使用连接谓词指定的连接

在连接谓词表示形式中，连接条件由比较运算符在 WHERE 子句中给出，将这种表示形式称为连接谓词表示形式，连接谓词又称为连接条件。

语法格式：

```
[<表名 1.>] <列名 1><比较运算符>[<表名 2.>] <列名 2>
```