第 6 章

分组查询与子查询

单元概述

本章将首先介绍常用的几个 Oracle 内置分组函数，并使用分组函数进行数据的统计查询，然后将介绍子查询，包括子查询的概念、类型、无关子查询与相关子查询的应用等。通过本章的学习，读者可以掌握常用的分组函数的使用，以及不同类型子查询的应用，为将来学习高级 SQL 语句应用奠定基础。

教学重点与难点

重点：

(1)分组函数、分组查询。

(2)无关子查询、相关子查询。

难点：

(1)分组的选择。

(2)相关子查询。

东软电子出版社

6.1 何时使用分组查询和子查询

在对数据库进行数据查询时，经常需要对现有数据进行一些统计与分析操作，例如：

- 查询所有员工每个月工资总和与平均工资？
- 查询工资最高和最低的工资是多少？
- 查询公司的总人数？
- 查询有奖金的总人数？

这些操作都需要使用 Oracle 提供的分组函数，对表中数据进行分组，然后进行统计。

在数据查询中，还经常需要在一个查询语句中使用另一个查询语句的运行结果，例如：

- 查询工资比 Jones 工资高的员工信息？
- 查询工资最低的员工姓名？

此时，就可以使用子查询，将一个查询语句嵌套在另外一个语句的内部，从而简化了查询操作。

在数据库应用系统中，数据分组查询、查询语句之间的相互嵌套（子查询）都是使用非常频繁的查询技术，也是熟练使用 SQL 语句所必需的技术。

6.2 分组查询

6.2.1 分组查询概述

在数据查询过程中，经常需要将数据进行分组，以便对各个组进行统计分析，即分组查询。如图 6-1 所示，统计不同部门的员工平均工资，此时就需要把表中数据根据部门号的不同进行分组，然后对每一组数据进行统计。对查询信息的统计通常使用内置的分组函数实现。

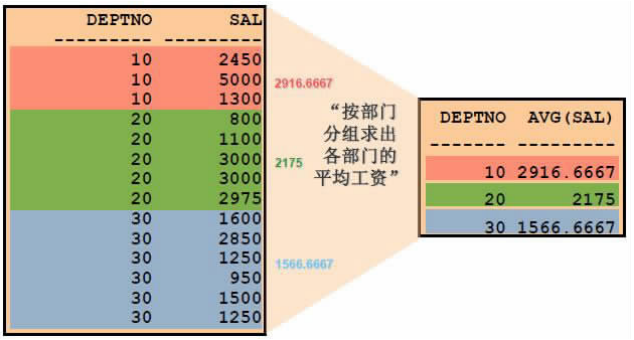

图 6-1 分组查询示例

东软电子出版社

6.2.2　常用的分组函数

分组函数是对表中一组记录进行操作，每组只返回一个结果，即首先要对表记录进行分组，然后再进行操作汇总，每组返回一个结果，分组时可能是整个表分为一组，也可能根据条件分成多组。

Oracle 10g 提供了很多分组函数，常用的分组函数见表 6-1。

表 6-1　　常用的分组函数

函　数	格　式	功　能
COUNT	COUNT([DISTINCT|ALL] *)	返回结果集中记录个数
COUNT	COUNT([DISTINCT|ALL] column)	返回结果集中非空记录个数
AVG	AVG([DISTINCT|ALL] column)	返回列或表达式的平均值
MAX	MAX([DISTINCT|ALL] column)	返回列或表达式的最大值
MIN	MIN([DISTINCT|ALL] column)	返回列或表达式的最小值
SUM	SUM([DISTINCT|ALL] column)	返回列或表达式的总和

例如，查询职位以 SALES 开头的所有员工平均工资、最低工资、最高工资、工资和，语句为：

```
SQL>SELECT AVG(sal), MAX(sal),MIN(sal), SUM(sal) FROM emp WHERE job LIKE 'SALES%';
AVG(SAL)      MAX(SAL)      MIN(SAL)      SUM(SAL)
-------------  -------------  -------------  -------------
  1400           1600           1250           5600
```

使用分组函数时需要注意以下几点。

(1)除了 COUNT(*)函数外，其他的分组函数都不考虑返回值或表达式为 NULL 的情况。

例如，统计所有员工的人数以及有奖金的员工的人数，语句为：

```
SQL>SELECT count( * ) ,count(comm) FROM emp;
COUNT( * )        COUNT(COMM)
--------------    ---------------------
    14                    4
```

(2)分组函数只能出现在目标列表达式、ORDER BY 子句、HAVING 子句中，不能出现在 WHERE 子句和 GROUP BY 子句中。

(3)默认对所有的返回行进行统计，包括重复的行；如果要统计不重复的行信息，则可以使用 DISTINCT 选项。

例如，查询有员工的部门的数量，语句为：

```
SQL>SELECT count(DISTINCT deptno) FROM emp;
COUNT(DISTINCT DEPTNO)
-------------------------------
                    3
```

(4)如果对查询结果进行了分组，则分组函数的作用范围为各个组，否则分组函数作用

东软电子出版社

于整个查询结果。

6.2.3 使用 GROUP BY 子句进行分组查询

在 Oracle 数据库中，分组查询是由 GROUP BY 子句实现的。分组查询的基本语法为：

```
SELECT column, group_function, …
FROM table
[WHERE condition]
[GROUP BY group_by_expression]
[ORDER BY column[ASC|DESC]];
```

其中：

- GROUP BY 子句用于指定分组列或分组表达式。
- 在分组查询中，SELECT 子句后的所有目标列或目标表达式要么是分组列，要么是分组表达式，要么是集合函数；但分组列或分组表达式不一定出现在 SELECT 子句后的目标列中。

1.单列分组查询

单列分组查询是指将查询出来的记录按照某一个指定的列进行分组，分组列的值相等的记录为一组，然后对每一个组进行统计。

例如，查询每个部门的部门号、人数和平均工资，语句为：

```
SQL>SELECT deptno,count( * ),avg(sal) FROM emp GROUP BY deptno;
DEPTNO       COUNT( * )          AVG(SAL)
-----------  ----------------  ---------------------
    10               4          2583.33333
    20               5                1835
    30               6          1183.33333
```

目标列中出现的 deptno 为分组列，count 和 avg 是分组函数，如果还有其他非分组的列或非分组函数，则将导致错误。例如：

```
SQL>SELECT ename,count( * ),avg(sal) FROM emp GROUP BY deptno;
SELECT ename,count( * ),avg(sal) FROM emp GROUP BY deptno
       *
ERROR 位于第 1 行：
ORA-00979：不是 GROUP BY 表达式
```

在连接查询中也可以进行分组查询，但一定要注意 SELECT 语句后的目标列的限制。例如：

```
SQL>SELECT e.deptno,d.dname,count(empno)
     FROM emp e,dept d
     WHERE e.deptno = d.deptno
     GROUP BY e.deptno;
SELECT e.deptno,d.dname,count(empno)
                 *
第 1 行出现错误：
```


东软电子出版社

```
ORA-00979：不是 GROUP BY 表达式
```

该语句执行错误的原因是 dname 列既不是分组列，也不是分组函数或分组表达式。正确语句应该为：

```
SQL>SELECT e.deptno,d.dname,count(empno)
      FROM emp e,dept d
      WHERE e.deptno = d.deptno
      GROUP BY e.deptno,d.dname;
DEPTNO        DNAME          COUNT(EMPNO)
------------  ------------------  ----------------------
    10        ACCOUNTING                  3
    20        RESEARCH                    5
    30        SALES                       6
```

2.多列分组查询

多列分组是指在 GROUP BY 子句中指定了两个或多个分组列。在多列分组统计时，系统根据分组列组合的不同值进行查询分组并进行统计。

例如，查询各个部门中不同职位的员工人数和平均工资，语句为：

```
SQL>SELECT deptno,job,count( * ),avg(sal) FROM emp  GROUP BY deptno,job;
  DEPTNO       JOB              COUNT( * )          AVG(SAL)
------------  ------------------  ------------------  ----------------------
    10        CLERK                   1                 900
    10        MANAGER                 1                2150
    10        PRESIDENT               1                4700
    20        CLERK                   2                 550
    …
    30        SALESMAN                4                1000
```

6.2.4　使用 HAVING 子句进行分组选择

对数据进行分组统计后，有时希望输出满足条件的分组，如图 6-2 所示，只有组内最高工资大于 2900 的组才输出，其他的组并不输出。

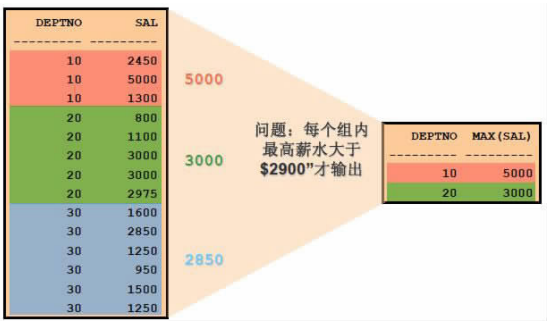

图 6-2　组选择示例

在分组查询中，组的选择通过 HAVING 子句实现，语法为：

```
SELECT column, group_function
```

```
FROM table
[WHERE condition]
[GROUP BY group_by_expression]
[HAVING group_condition]
[ORDER BY column];
```

例如，查询每个部门最高工资大于 2900 的部门编号、最高工资，语句为：

```
SQL>SELECT deptno, max(sal) FROM emp GROUP BY deptno HAVING max(sal)>2900;
DEPTNO        MAX(SAL)
----------    ------------
    20            3000
    10            5000
```

在分组查询中，HAVING 子句作用于组，而 WHERE 子句作用于记录。先根据 WHERE 条件查询记录，最后根据 HAVING 条件决定哪些组返回。例如，统计 10 号部门中各个职位的员工人数和平均工资，并返回平均工资高于 1000 的职位人数和平均工资，语句为：

```
SQL>SELECT job,count( * ),avg(sal) FROM emp WHERE deptno=10
     GROUP BY job HAVING avg(sal)>1000;
JOB            COUNT( * )      AVG(SAL)
------------   -----------------   ------------------
MANAGER              1              2150
PRESIDENT            1              4700
```

6.2.5 SELECT 语句执行过程

在进行数据的分组统计以及组的选择操作过程中，SELECT 语句的执行步骤为：

(1)通过 FROM 子句中找到需要查询的表；

(2)通过 WHERE 子句进行非分组函数筛选判断；

(3)通过 GROUP BY 子句完成分组操作；

(4)通过 HAVING 子句完成组函数筛选判断；

(5)通过 SELECT 子句选择显示的列或表达式及组函数；

(6)通过 ORDER BY 子句进行排序操作。

下面用一个实际的例子说明 SELECT 语句的执行过程。

例如，查询不同部门的、职位为 SALESMAN 或 MANAGER 或 CLERK、平均工资大于 1000 的部门号、职位名称及其平均工资，并按平均工资从高到低排序。语句为：

```
SQL>SELECT deptno,job,avg(sal)
     FROM emp
     WHERE job in ('SALESMAN','MANAGER','CLERK')
     GROUP BY deptno,job
     HAVING avg(sal)>1000
     ORDER BY 3 DESC;
DEPTNO       JOB          AVG(SAL)
```

东软电子出版社

```
----------  --------------  --------------
    10      MANAGER              3550
    20      MANAGER              3175
    30      MANAGER              2950
    10      CLERK                2300
    30      SALESMAN             1400
    20      CLERK                1050
```

已选择 6 行。

执行过程如图 6-3 所示。

(1)根据 FROM 子句从 emp 表中获取数：

```
SELECT deptno,job,sal FROM emp;
```

(2)根据 WHERE 子句对步骤(1)的结果集进行选择：

```
SELECT deptno,job,sal FROM emp
WHERE job IN ('SALESMAN','MANAGER','CLERK');
```

(3)对步骤(2)的结果集根据 deptno 和 job 两列进行分组统计：

```
SELECT deptno,job,sal FROM emp
WHERE job IN ('SALESMAN','MANAGER','CLERK')
GROUP BY deptno,job;
```

(4)对步骤(3)的结果集进行选择，平均工资大于 1000 的保留：

```
SELECT deptno,job,sal FROM emp
WHERE job IN ('SALESMAN','MANAGER','CLERK')
GROUP BY deptno,job
HAVING avg(sal)>1000;
```

(5)把经过步骤(4)处理过程的结果输出。

(6)对步骤(5)输出的无序结果进行排序，重新输出有序结果。

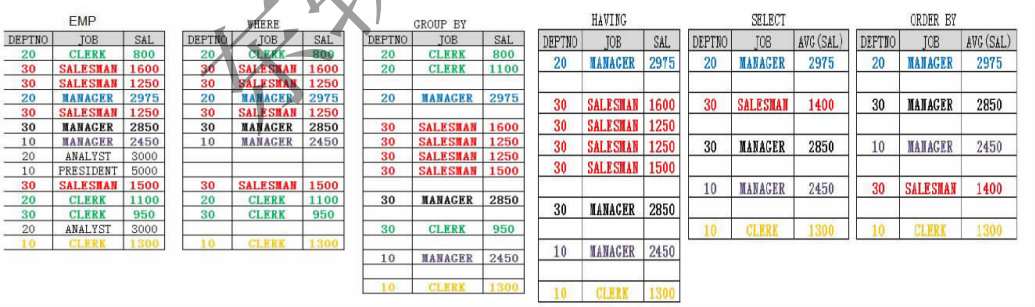

图 6-3　SELECT 语句执行过程

由 SELECT 语句的执行顺序可以看出，分组查询是在 GROUP BY 子句执行后进行的，而 WHERE 子句的执行是在 GROUP BY 子句之前，因此在 WHERE 子句中不能使用分组函数。例如：

```
SQL>SELECT deptno,count( * ),avg(sal) FROM emp WHERE avg(sal)>1000;
SELECT deptno,count( * ),avg(sal) FROM emp WHERE avg(sal)>1000
                                                 *
```

第 1 行出现错误：

长软电子出版社

```
ORA-00934：此处不允许使用分组函数
```

6.2.6 分组函数嵌套

与单行函数类似，分组函数可以独立使用，也可以嵌套使用。但是需要注意的是，分组函数最多只能嵌套两层。

例如，查询部门平均工资最高的部门平均工资、员工数最多的部门的员工数量，语句为：

```
SQL>SELECT max(avg(sal)) avgsal,max(count( * )) num FROM emp GROUP BY deptno;
  AVGSAL          NUM
---------------   -----------
3983.33333          6
```

由于内层的分组函数执行一次分组查询后，得到的结果只能再进行一次整体统计查询，而无法再继续分组。因此，分组函数最多嵌套两层。例如，上例执行过程为：

(1)执行内层的分组查询，统计各个部门的平均工资与人数。实际执行语句为：

```
SQL>SELECT avg(sal) avgsal,count( * ) num FROM emp GROUP BY deptno;
       AVGSAL            NUM
-------------------   ----------------
  1583.33333              6
        2335              5
  3983.33333              3
```

(2)在步骤(1)的执行结果基础上，利用 max 函数求各列的最大值，此时已经无法再进行分组了，只能是对整体进行统计。

6.3 无关子查询

6.3.1 子查询概述

子查询是指嵌套在其他 SQL 语句中的 SELECT 语句，也称为内部查询。在执行时，由里向外，先处理子查询，再将子查询的返回结果用于其父语句(外部语句)的执行。图 6-4 是一个子查询应用的示例。

图 6-4 子查询示例

可以在 WHERE 子句、HAVING 子句、FROM 子句中使用子查询，基本形式为：

东软电子出版社

```
SELECT select_list FROM table
WHERE expr operator (SELECT select_list  FROM table);
```

根据返回结果的不同，子查询可以分为单行子查询、多行子查询、多列子查询 3 种类型。根据与外部父语句的关系，子查询又可分为无关子查询和相关子查询两类。

6.3.2　单行子查询

单行子查询是指子查询只返回一行数据，而且只返回一列的数据。当在 WHERE 子句中使用单行子查询时，可以使用单行比较运算符，如＝、＞、＜、＞＝、＜＝、！＝等。

例如，查询比 JONES 工资高的其他员工的名字，语句为：

```
SQL>SELECT ename FROM emp WHERE sal > (SELECT sal FROM emp WHERE ename='JONES');
ENAME
----------
SCOTT
KING
FORD
```

在子查询中可以使用分组函数。例如，查询工资最低的员工姓名、职位及工资，语句为：

```
SQL>SELECT ename, job, sal FROM emp WHERE sal =(SELECT MIN(sal)FROM emp);
ENAME       JOB        SAL
----------  ---------  ----------
SMITH       CLERK      800
```

可以在 HAVING 子句中使用子查询。例如，查询部门最低工资比 20 部门最低工资高的部门编号及最低工资，语句为：

```
SQL>SELECT deptno, MIN(sal) FROM emp GROUP BY deptno
     HAVING MIN(sal) >(SELECT MIN(sal) FROM emp WHERE deptno = 20);
DEPTNO        MIN(SAL)
------------  ------------
    30            950
    10           1300
```

6.3.3　多行子查询

多行子查询是指返回多行数据，且只返回一列的数据。如果子查询返回多行数据，而比较运算符采用的是单行运算法，那么执行将出现错误，例如：

```
SQL>SELECT empno,ename FROM emp
     WHERE sal = (SELECT MIN(sal) FROM emp GROUP BY deptno);
SELECT empno,ename FROM emp WHERE sal = (SELECT MIN(sal) FROM emp GROUP BY deptno)
                                         *
第 1 行出现错误：
ORA-01427：单行子查询返回多个行
```

因此，当子查询为多行子查询时，必须使用多行比较运算符，包括 IN、NOT IN、＞ANY、＝ANY、＜ANY、＞ALL、＜ALL 等，其含义见表 6-2。

东软电子出版社

表 6-2　　多行比较运算符

运　算　符	含　义
IN	与子查询返回结果中任何一个值相等
NOT IN	与子查询返回结果中任何一个值都不等
>ANY	比子查询返回结果中某一个值大
=ANY	与子查询返回结果中某一个值相等
<ANY	比子查询返回结果中某一个值小
>ALL	比子查询返回结果中所有值都大
<ALL	比子查询返回结果中任何一个值都小
=ALL	

例如，查询是经理的员工的姓名与工资，语句为：

```
SQL>SELECT ename, sal FROM emp WHERE empno IN (SELECT mgr FROM emp);
ENAME              SAL
---------          ---------
JONES              2975
BLAKE              2850
CLARK              2450
SCOTT              3000
KING               5000
FORD               3000
已选择 6 行。
```

由于=ANY 运算符与 IN 运算符的作用相同，所以上面的语句还可以表示为：

```
SQL>SELECT ename, sal FROM emp WHERE empno = ANY (SELECT mgr FROM emp);
```

>ANY 运算符表示比子查询结果集中的某个值大，即比最小值大。例如，查询部门编号不为 10，且工资比 10 部门任意一名员工工资高的员工编号、姓名、职位、工资，语句为：

```
SQL>SELECT empno, ename, job, sal FROM emp
     WHERE sal > ANY (SELECT sal FROM emp WHERE deptno = 10) AND deptno <> 10;
EMPNO   ENAME     JOB          SAL
-------  ---------  ----------   ---------
 7788   SCOTT     ANALYST      3000
 7902   FORD      ANALYST      3000
 7566   JONES     MANAGER      2975
 7698   BLAKE     MANAGER      2850
 7499   ALLEN     SALESMAN     1600
 7844   TURNER    SALESMAN     1500
已选择 6 行。
```

<ANY 运算符表示比子查询结果集中的某个值小，即比最大值小。例如，查询部门编号不为 10，且工资比 10 部门任意一名工资低的员工编号、姓名、职位、工资，语句为：

```
SQL>SELECT empno, ename, job, sal FROM emp
```

东软电子出版社

```
    WHERE sal < ANY (SELECT sal FROM emp WHERE deptno = 10) AND deptno <> 10;
EMPNO  ENAME      JOB         SAL
------ --------- ----------- ---------------
7369   SMITH     CLERK        800
7900   JAMES     CLERK        950
7876   ADAMS     CLERK       1100
7521   WARD      SALESMAN    1250
7654   MARTIN    SALESMAN    1250
…
已选择 11 行。
```

>ALL 表示大于子查询结果集中的所有行，即大于最大值。例如，查询部门编号不为 20，且工资比 20 部门所有员工工资高的员工编号、姓名、职位、工资，语句为：

```
SQL>SELECT empno, ename,job, sal FROM emp WHERE sal > ALL (
SELECT sal FROM emp WHERE deptno= 20) AND deptno <> 20;
EMPNO    ENAME    JOB             SAL
-------- -------- --------------- ----------
  7839   KING     PRESIDENT        5000
```

< ALL 表示小于子查询结果集中的所有行，即小于最小值。例如，查询部门编号不为 10，且工资比 10 部门所有员工工资低的员工编号、姓名、职位、工资，语句为：

```
SQL>SELECT empno, ename,job, sal FROM emp WHERE sal < ALL (
    SELECT sal FROM emp WHERE deptno= 10) AND deptno <> 10;
EMPNO    ENAME    JOB             SAL
-------- -------- ------------- ----------
  7369   SMITH    CLERK            800
  7521   WARD     SALESMAN        1250
  7654   MARTIN   SALESMAN        1250
  7876   ADAMS    CLERK           1100
  7900   JAMES    CLERK            950
```

6.3.4　多列子查询

之前讲的子查询都是在一个条件表达式内与子查询的一个列进行比较，多列子查询可以在一个条件表达式内同时与子查询的多个列进行比较。多列子查询通常用 IN 操作符完成。

例如，查询与 1981 年入职的任意一个员工的部门和职位相同的员工姓名、部门、职位、入职日期，不包括 1981 年入职员工，语句为：

```
SQL>SELECT ename, deptno, job, hiredate FROM emp WHERE (deptno, job) IN (
    SELECT deptno,job FROM emp WHERE to_char(hiredate,'YYYY')='1981')
    AND to_char(hiredate,'YYYY')<>'1981';
ENAME     DEPTNO    JOB           HIREDATE
--------- --------- ------------- ----------------
SCOTT          20   ANALYST       19-4 月 -87
```

东软电子出版社

例如，查询每个部门工资最低的员工姓名及工资。

```
SQL>SELECT ename,sal FROM emp
     WHERE (deptno,sal) IN(SELECT deptno,min(sal) FROM emp GROUP BY deptno);
ENAME          SAL
---------     ---------
JAMES          950
SMITH          900
MILLER        2300
```

6.3.5 子查询中的空值

如果子查询的结果集中包含有空值(NULL)，那么在比较时不能使用 NOT IN 运算符。例如，查询不是领导的员工的信息，使用下列语句将获取不到正确结果。

```
SQL>SELECT ename FROM emp WHERE empno NOT IN(SELECT mgr FROM emp);
未选定行
```

上面的 SQL 语句试图查找出不是领导的员工，逻辑上，这个 SQL 语句应该会返回 8 条记录，但是却一条也没返回，其原因在于子查询的结果集中有一个空值，这个空值导致主查询没有记录返回。这是因为所有的条件和空值比较结果都是空值。因此无论什么时候只要空值有可能成为子查询结果集中的一部分时，就不能使用 NOT IN 运算符。

如果子查询中存在空值，可以通过限定子查询的查询条件排除空值。例如，上例可以采用下列方法解决：

```
SQL>SELECT ename FROM emp
     WHERE empno NOT IN(SELECT mgr FROM emp WHERE mgr IS NOT NULL);
ENAME
---------
TURNER
WARD
MARTIN
ALLEN
…
已选择 8 行。
```

6.3.6 在 FROM 子句中使用子查询

当在 FROM 子句中使用子查询时，该子查询被作为视图对待，必须为该子查询指定别名。在执行时，先执行 FROM 子句中的子查询，返回一个临时的虚表，可以像真实表一样使用。

例如，查询比自己部门平均工资高的员工姓名、工资、部门编号、部门平均工资，语句为：

```
SQL>SELECT a.ename, a.sal, a.deptno, b.salavg FROM emp a,
     (SELECT deptno, avg(sal) salavg FROM emp GROUP BY deptno) b
     WHERE a.deptno = b.deptno AND a.sal > b.salavg;
ENAME      SAL       DEPTNO            SALAVG
```


东软电子出版社

```
---------  --------  ---------------  ------------------
ALLEN      1600          30           1566.66667
JONES      2975          20                 2175
BLAKE      2850          30           1566.66667
SCOTT      3000          20                 2175
KING       5000          10           2916.66667
FORD       3000          20                 2175
已选择 6 行。
```

该语句的执行过程为：

(1)执行子程序，得到的临时数据，命名为表 b。

```
SQL>SELECT deptno, avg(sal) salavg FROM emp GROUP BY deptno;
DEPTNO          SALAVG
-----------   --------------
     30      1583.33333
     20            2335
     10      3983.33333
```

(2)将表 b 与表 a(emp 表)进行连接查询。

查询每个部门工资最低的员工姓名及工资，可以采用前面的多列子查询的方式进行，也可以在 FROM 子句中使用子查询的方式进行。例如：

```
SQL>SELECT a.ename, a.sal FROM emp a,
     (SELECT deptno, min(sal) minsal FROM emp GROUP BY deptno) b
     WHERE a.deptno = b.deptno AND a.sal=b.minsal;
ENAME          SAL
---------     ---------
SMITH           900
JAMES           950
MILLER         2300
```

6.3.7　分页查询

在 Oracle 10g 中，如果要返回查询结果中前 N 条记录，或对查询结果进行分页显示，需要使用伪列 ROWNUM，对查询结果集进行排序，并返回符合条件的记录。

1.ROWNUM

伪列 ROWNUM 的功能是在每次查询时，为返回的每条记录赋一个序号，该序号表示 Oracle 从表或连接结果集中返回该记录的顺序。第一条记录显示为 1，第二条记录显示为 2，以此类推。但是需要注意的是，每条记录的序号是排序前的序号，与排序没有关系。

例如，查询员工号、员工名及职位，同时显示检索顺序，语句为：

```
SQL> SELECT rownum,empno, ename, job FROM emp;
ROWNUM   EMPNO    ENAME     JOB
--------  --------  ---------  -------------
    1    7369      SMITH     CLERK
```

东软电子出版社

```
    2    7499     ALLEN    SALESMAN
    3    7521     WARD     SALESMAN
    4    7566     JONES    MANAGER
    5    7654     MARTIN   SALESMAN
…
已选择 14 行。
```

使用伪列 ROWNUM 时需要注意：

- ROWNUM 和 ORDER BY 一起使用时，因为 ROWNUM 在记录输出时生成，而 ORDER BY 子句在最后执行，所以当两者一起使用时，需要注意 ROWNUM 实际是已经被排了序的 ROWNUM。
- 如下 SQL 语句，SELECT * FROM EMP WHERE ROWNUM>2；查询不到任何记录，因为 ROWNUM 是在记录输出时才生成，且总是从 1 开始，所以输出的第一条记录不满足>2 的条件，被过滤掉，第二条的 ROWNUM 又成了 1，又不满足>2 的条件，又被过滤掉，依此类推，所以永远没有满足条件的记录，返回为空。所以，对 ROWNUM 只能执行<、<=运算，不能执行>、>=或表示区间的 BETWEEN…AND 运算。

2. TOP-N 查询

Top-N 查询主要是实现表中按照某个列排序，输出最大或最小的 N 条记录功能。使用 ROWNUM 伪列进行 TOP-N 查询的语法为：

```
SELECT [列名], ROWNUM
FROM (SELECT [列名]FROM 表名 ORDER BY Top-N 操作的列 ASC|DESC)
WHERE ROWNUM <= N;
```

其中：

- ASC：表示查询最小的 N 条记录。
- DESC：表示查询最大的 N 条记录。

例如，查询工资排序在前 1～5 名的员工号、员工名、工资及其工资排序号，语句为：

```
SQL>SELECT ROWNUM,empno,ename,sal
     FROM (SELECT empno,ename,sal FROM emp ORDER BY sal DESC) inner_emp
     WHERE ROWNUM<=5;
ROWNUM     EMPNO     ENAME       SAL
---------  ---------  ---------  ---------
     1      7839      KING       5000
     2      7788      SCOTT      3000
     3      7902      FORD       3000
     4      7566      JONES      2975
     5      7698      BLAKE      2850
```

3. 分页查询

在 Oracle 10g 中，利用 ROWNUM 的特性，可以实现数据库端的分页查询。

(1)当未指定需要按照某列排序时，语法为：

```
SELECT b. *
FROM (SELECT ROWNUM rn,[列名 1,列名 2,....列名 n]
```


东软电子出版社

```
    FROM 表名 1,[表名 2,...表名 n]
    WHERE [条件表达式 AND ] ROWNUM <=目标页数 * 每页记录数) b
WHERE rn > (目标页数-1) * 每页记录数
```

或

```
SELECT b. *
FROM (SELECT ROWNUM rn,[列名 1,列名 2,....列名 n]
    FROM 表名 1,[表名 2,...表名 n] [WHERE 条件表达式]) b
WHERE rn <=目标页数 * 每页记录数 and rn > (目标页数-1) * 每页记录数
```

这两种方法执行结果是一样的，但第 1 种方法的执行效率要高于第 2 种方法，表中数据量越大，第 1 种方法的效率越高。原因在于，在第 1 种方法中，子查询语句(SELECT ROWNUM rn,[列名 1,列名 2,…列名 n] FROM 表名 1,[表名 2,…表名 n] WHERE [条件表达式 AND] ROWNUM <=目标页数 * 每页记录数)b 获取的是指定页数据及其之前的所有数据，而其后的数据没有提取出来，而在第 2 种方法中，子查询(SELECT ROWNUM rn,[列名 1,列名 2,…列名 n]FROM 表名 1,[表名 2,…表名 n] [WHERE 条件表达式]) b 将表中所有数据全部提取出来了。例如，表中有 10000 条记录，每页显示 100 条，当前要查询第 2 页的数据，在第 1 种方法中，子查询返回 200 条记录，而第 2 种方法中，子查询返回 10000 条记录。

例如，分页显示员工信息，每页显示 5 条记录，查询第 2 页的数据，语句为：

```
SQL> SELECT b. * FROM (
    SELECT ROWNUM rn,empno,ename,sal FROM emp WHERE ROWNUM<=2 * 5) b
    WHERE rn >(2-1) * 5;
RN     EMPNO       ENAME          SAL
----  ----------  ---------- ----------
 6     7698        BLAKE         2950
 7     7782        CLARK         3550
 8     7788        SCOTT         3200
 9     7839        KING          6100
10     7844        TURNER        1500
```

或

```
SQL>SELECT b. * FROM (SELECT ROWNUM rn,empno,ename,sal FROM emp ) b
    WHERE rn <=2 * 5 AND rn > (2-1) * 5;
RN     EMPNO     ENAME       SAL
----  ----------  ----------  ---------
 6     7698      BLAKE       2950
 7     7782      CLARK       3550
 8     7788      SCOTT       3200
 9     7839      KING        6100
10     7844      TURNER      1500
```

(2)当指定需要按照某列排序时，语法为：

```
SELECT *
FROM (SELECT ROWNUM rn, b. *
```

东软电子出版社

```
    FROM (SELECT 列名 1 [,列名 2,…列名 n] FROM 表名 1,[表名 2,…表名 n]
          [WHERE 子句] ORDER BY 要排序的列 ASC|DESC ) b
    WHERE ROWNUM <=目标页数 * 每页记录数)
WHERE rn > (目标页数-1) * 每页记录数 ;
```

或

```
SELECT *
FROM (SELECT ROWNUM rn, b. *
    FROM (SELECT 列名 1 [,列名 2,…列名 n] FROM 表名 1,[表名 2,…表名 n]
        [WHERE 子句] ORDER BY 要排序的列 ASC|DESC ) b )
WHERE rn <=目标页数 * 每页记录数 and rn > (目标页数-1) * 每页记录数;
```

与未指定需要按照某列排序类似,第 1 种方法的执行效率高于第 2 种方法。

例如,按员工工资降序分页显示员工信息,每页显示 5 条记录,查询第 2 页的数据,语句为:

```
SQL>SELECT *
    FROM (SELECT ROWNUM rn,empno,ename,sal
          FROM (SELECT empno,ename,sal FROM emp ORDER BY sal DESC) nested_order1
          WHERE ROWNUM<=2 * 5)nested_order2
   WHERE num>(2-1) * 5;
RN      EMPNO     ENAME       SAL
-----  ---------  ----------  ---------
 6      7782      CLARK       2450
 7      7499      ALLEN       1600
 8      7844      TURNER      1500
 9      7934      MILLER      1300
10      7521      WARD        1250
```

或

```
SQL>SELECT *
    FROM (SELECT ROWNUM rn,empno,ename,sal
    FROM (SELECT empno,ename,sal FROM emp ORDER BY sal DESC) nested_order1)nested_order2
   WHERE rn>(2-1) * 5 AND rn<=2 * 5;
RN       EMPNO     ENAME       SAL
-------  ---------  ---------  ---------
 6       7782      CLARK       2450
 7       7499      ALLEN       1600
 8       7844      TURNER      1500
 9       7934      MILLER      1300
10       7521      WARD        1250
```

东软电子出版社

6.4　相关子查询

6.4.1　相关子查询概述

在前面介绍的各种子查询中，子查询在执行时并不需要外部父查询的信息，这种查询称为无关子查询。如果子查询在执行时需要引用外部父查询的信息，那么这种查询就称为相关子查询。

在无关子查询中，子查询是以嵌套的方式写在父查询的 WHERE、HAVING、FROM 子句中，所以被称为嵌套子查询。嵌套子查询的执行过程为：

(1)子查询首先执行一次，返回结果集。

(2)利用子查询的结果集确认或取消父查询的候选行。

例如，查询比本部门平均薪水高的员工姓名、薪水，使用无关子查询实现的语句为：

```
SQL>SELECT empno,ename,sal,avgsal
     FROM emp e ,(SELECT deptno,avg(sal) avgsal FROM emp GROUP BY deptno) d
     WHERE e.deptno =d.deptno AND e.sal >d.avgsal;
```

使用相关子查询也可以解决上述问题，并且更加容易理解。在相关子查询中，当子查询中引用父查询表中的列一次时，Oracle 服务器执行一次相关子查询，即父查询中的行每被处理一次，子查询就执行一次。

利用相关子查询解决上述问问题的思路是：

(1)取得父查询第一条候选行记录的 sal 和 deptno；

(2)根据取得的 deptno，获取该部门的平均工资；

(3)用第一步取得的 sal 和第二步取得的平均工资作比较，如果 sal>平均工资，则第一条候选行记录被显示；否则，不被显示；

(4)依次取得父查询中的第 2 行到最后一行，重复执行 1～3。

在 Oracle 数据库中，实现相关子查询的基本形式为：

```
SELECT column1, column2, ...FROM table1  outer WHERE column1 operator
(SELECT column1, column2 FROM table2 WHERE column= outer  .column);
```

相关子查询的执行过程：

(1)取得父查询的候选行；

(2)用候选行被子查询引用列的值执行子查询；

(3)用来自子查询的值确认或取消候选行；

(4)重复步骤 1、2、3，直到父查询中无剩余的候选行。

查询比本部门平均薪水高的员工姓名、薪水，使用相关子查询实现的语句为：

```
SQL>SELECT empno,ename, sal,deptno FROM emp outer WHERE sal >(
     SELECT AVG(sal) FROM emp WHERE deptno = outer.deptno) ;
EMPNO     ENAME      SAL     DEPTNO
-------  ----------  ----------  ----------
```

东软电子出版社

```
7499      ALLEN      1600        30
7566      JONES      2975        20
7698      BLAKE      2850        30
7788      SCOTT      3000        20
…
已选择 6 行。
```

例如，查询是领导的员工的员工号、员工名、工资，利用相关子查询实现的语句为：

```
SQL>SELECT ename , job, sal, deptno FROM emp e
     WHERE 1<=(SELECT count(empno) FROM EMP WHERE mgr =e.empno);
ENAME       JOB            SAL       DEPTNO
---------   --------------  ---------  -------------
JONES       MANAGER        2975          20
BLAKE       MANAGER        2850          30
CLARK       MANAGER        2450          10
SCOTT       ANALYST        3000          20
KING        PRESIDENT      5000          10
FORD        ANALYST        3000          20
已选择 6 行。
```

例如，查询每个部门工资最低的员工姓名及工资，除了可以使用前面介绍的无关子查询实现，也可以使用下列两种相关子查询实现：

```
SQL>SELECT ename,sal FROM emp e
     WHERE sal IN (SELECT min(sal) FROM emp WHERE deptno = e.deptno);
SQL>SELECT ename,sal FROM emp e
     WHERE (SELECT count(empno) FROM emp WHERE deptno = e.deptno AND sal <e.sal)<1;
```

运行结果相同，如下所示：

```
ENAME              SAL
---------          --------
SMITH               900
JAMES               950
MILLER             2300
```

6.4.2 EXISTS 和 NOT EXISTS 操作符

相关子查询中经常使用 EXISTS 和 NOT EXISTS 操作符来进行操作。

EXISTS 操作符判断是否“存在”，具体操作如下：

- 子查询中如果有记录找到，子查询语句不会继续执行，返回值为 TRUE；
- 子查询中如果到表的末尾也没有记录找到，返回值为 FALSE。

EXISTS 子查询并不需要确切记录返回，只判断是否有记录存在，只要找到相关记录，子查询就不需要再执行，然后再进行下面的操作。这样大大提高了语句的执行效率。

NOT EXISTS 操作符正好相反，判断子查询是否没有返回值。如果没有返回值，表达式为真，如果找到一条返回值，则为假。

东软电子出版社

例如，利用 EXISTS 操作符查询是领导的员工的员工名、职位、工资、部门号。语句为：

```
SQL>SELECT ename , job, sal, deptno FROM emp e
     WHERE EXISTS (SELECT 1 FROM emp WHERE mgr= e.empno);
ENAME       JOB              SAL      DEPTNO
---------   --------------   ----------   ----------
JONES       MANAGER        2975         20
BLAKE       MANAGER        2850         30
CLARK       MANAGER        2450         10
SCOTT       ANALYST        3000         20
KING        PRESIDENT      5000         10
FORD        ANALYST        3000         20
已选择 6 行。
```

因为 EXISTS 子句中，并没有确切记录返回，只返回真或假。所以 1 只是占位用，无实际意义。

例如，查询由有员工的部门的信息。语句为：

```
SQL>SELECT * FROM dept WHERE EXISTS(SELECT 1 FROM emp WHERE deptno=dept.deptno);
DEPTNO     DNAME          LOC
---------  ----------------   ------------
    20     RESEARCH       DALLAS
    30     SALES          CHICAGO
    10     ACCOUNTING     NEW YORK
```

如果查询不是领导的员工的员工名、职位、工资、部门号信息，可以使用 NOT EXISTS 相关子查询，语句为：

```
SQL>SELECT ename, job, sal, deptno FROM emp e
     WHERE NOT EXISTS (SELECT '1' FROM emp WHERE mgr= e.empno);
ENAME       JOB              SAL      DEPTNO
---------   --------------   ----------   ---------
SMITH       CLERK           800         20
ALLEN       SALESMAN       1600         30
WARD        SALESMAN       1250         30
MARTIN      SALESMAN       1250         30
TURNER      SALESMAN       1500         30
ADAMS       CLERK          1100         20
JAMES       CLERK           950         30
MILLER      CLERK          1300         10
已选择 8 行。
```

NOT EXISTS 操作符因为运算方法与 NOT IN 不同，只会返回 TRUE 或 FALSE，不会返回空值，所以不需要考虑子查询去除空值的问题。

例如，查询不是领导的员工的信息。语法为：

```
SQL>SELECT empno,ename,deptno FROM emp e
    WHERE NOT EXISTS(SELECT 1 FROM emp WHERE mgr=e.empno);
```

东软电子出版社

```
EMPNO      ENAME       DEPTNO
---------  ----------  ---------
  7844     TURNER          30
  7521     WARD            30
  7654     MARTIN          30
  7499     ALLEN           30
  7934     MILLER          10
  …
```

已选择 8 行。

也可以使用 NOT EXISTS 关键字，实现查询每个部门工资最低的员工姓名及工资。语句为：

```
SQL>SELECT ename,sal FROM emp e
     WHERE NOT EXISTS(SELECT '1' FROM emp WHERE deptno = e.deptno AND sal <e.sal);
```

6.5 课后作业

(1)查询部门 20 的员工，每个月的工资总和及平均工资。

(2)查询工作在 CHICAGO 的员工人数、最高工资及最低工资。

(3)查询员工表中一共有几种岗位类型。

(4)查询每个部门的部门编号、部门名称、部门人数、最高工资、最低工资、工资总和、平均工资。

(5)查询每个部门及每个岗位的部门编号、部门名称、岗位名称、部门人数、最高工资、最低工资、工资总和、平均工资。

(6)查询每个经理所管理的人数、经理编号、经理姓名，要求包括没有经理的人员信息。

(7)查询部门人数大于 2 的部门编号、部门名称、部门人数。

(8)查询部门平均工资大于 2000，且人数大于 2 的部门编号、部门名称、部门人数、部门平均工资，并按照部门人数升序排序。

(9)查询部门平均工资在 2500 元以上的部门名称及平均工资。

(10)查询员工岗位中不是以“SA”开头并且平均工资在 2500 元以上的岗位及平均工资，并按平均工资降序排序。

(11)查询部门人数在 2 人以上的部门名称、最低工资、最高工资，并对求得的工资进行四舍五入到整数位。

(12)查询岗位不为 SALESMAN，工资和大于等于 2500 的岗位及每种岗位的工资和。

(13)显示经理号码和经理姓名，这个经理所管理员工的最低工资，没有经理的 KING 也要显示，不包括最低工资小于 3000 的，按最低工资由高到低排序。

(14)写一个查询，显示每个部门最高工资和最低工资的差额。

(15)查询入职日期最早的员工姓名、入职日期。

(16)查询工资比 SMITH 工资高并且工作地点在 CHICAGO 的员工姓名、工资、部门

东软电子出版社

名称。

(17)查询入职日期比 20 部门入职日期最早的员工还要早的员工姓名、入职日期。

(18)查询部门人数大于所有部门平均人数的的部门编号、部门名称、部门人数。

(19)查询入职日期比 10 部门任意一个员工晚的员工姓名、入职日期,不包括 10 部门员工。

(20)查询入职日期比 10 部门所有员工晚的员工姓名、入职日期,不包括 10 部门员工。

(21)查询职位和 10 部门任意一个员工职位相同的员工姓名、职位,不包括 10 部门员工。

(22)查询职位及经理和 10 部门任意一个员工职位及经理相同的员工姓名、职位,不包括 10 部门员工。

(23)查询职位及经理和 10 部门任意一个员工职位或经理相同的员工姓名、职位,不包括 10 部门员工。

(24)查询比自己职位平均工资高的员工姓名、职位、部门名称、职位平均工资。

(25)查询职位和经理同员工 SCOTT 或 BLAKE 完全相同的员工姓名、职位,不包括 SCOOT 和 BLAKE 本人。

(26)查询不是经理的员工姓名。

(27)查询入职日期最早的前 5 名员工姓名。

(28)查询工资高于编号为 7782 的员工工资,并且和 7369 号员工从事相同工作的员工的编号、姓名及工资。

(29)查询工资最高的员工姓名和工资。

(30)查询部门最低工资高于 10 号部门最低工资的部门的编号、名称及部门最低工资。

(31)查询员工工资为其部门最低工资的员工的编号和姓名及工资。

(32)显示经理是 KING 的员工姓名、工资。

(33)显示比员工 SMITH 参加工作时间晚的员工姓名、工资、参加工作时间。

(34)使用子查询的方式查询哪些职员在 NEW YORK 工作。

(35)写一个查询显示和员工 SMITH 工作在同一个部门的员工姓名、雇用日期,查询结果中排除 SMITH。

(36)写一个查询显示其工资比全体职员平均工资高的员工编号、姓名。

(37)写一个查询显示其上级领导是 King 的员工姓名、工资。

(38)显示所有工作在 RESEARCH 部门的员工姓名、职位。

(39)查询每个部门的部门编号、平均工资,要求部门的平均工资高于部门 20 的平均工资。

(40)查询大于自己部门平均工资的员工姓名、工资、所在部门平均工资、高于部门平均工资的额度。

(41)列出至少有一个雇员的所有部门。

(42)列出薪金比“SMITH”多的所有雇员。

(43)列出入职日期早于其直接上级的所有雇员。

(44)找员工姓名和直接上级的名字。

东软电子出版社

(45)显示部门名称和人数。

(46)显示每个部门的最高工资的员工。

(47)显示出和员工号 7369 部门相同的员工姓名、工资。

(48)显示出和姓名中包含“W”的员工相同部门的员工姓名。

(49)显示出工资大于平均工资的员工姓名、工资。

(50)显示出工资大于本部门平均工资的员工姓名、工资。

(51)显示每位经理管理员工的最低工资,及最低工资者的姓名。

(52)显示比工资最高的员工参加工作时间晚的员工姓名、参加工作时间。

(53)显示出平均工资最高的的部门平均工资及部门名称。

(54)查询工作在 CHICAGO 并且入职日期最早的前 2 名员工姓名、入职日期。

(55)按照每页显示 5 条记录,分别查询第 1 页、第 2 页、第 3 页信息,要求显示员工姓名、入职日期、部门名称。

(56)按照每页显示 5 条记录,分别查询工资最高的第 1 页、第 2 页、第 3 页信息,要求显示员工姓名、入职日期、部门名称、工资。

(57)查询比所在职位平均工资高的员工姓名、职位。

(58)查询工资为其部门最低工资的员工编号、姓名、工资。

(59)查询所有雇员编号、名字和部门名字。

(60)查询每个部门工资最低的两个员工编号、姓名、工资。

(61)列出至少有一个雇员的所有部门名称。

(62)列出一个雇员都没有的所有部门名称。

(63)查询每个部门工资前两名高的员工姓名、工资。

东软电子出版社