

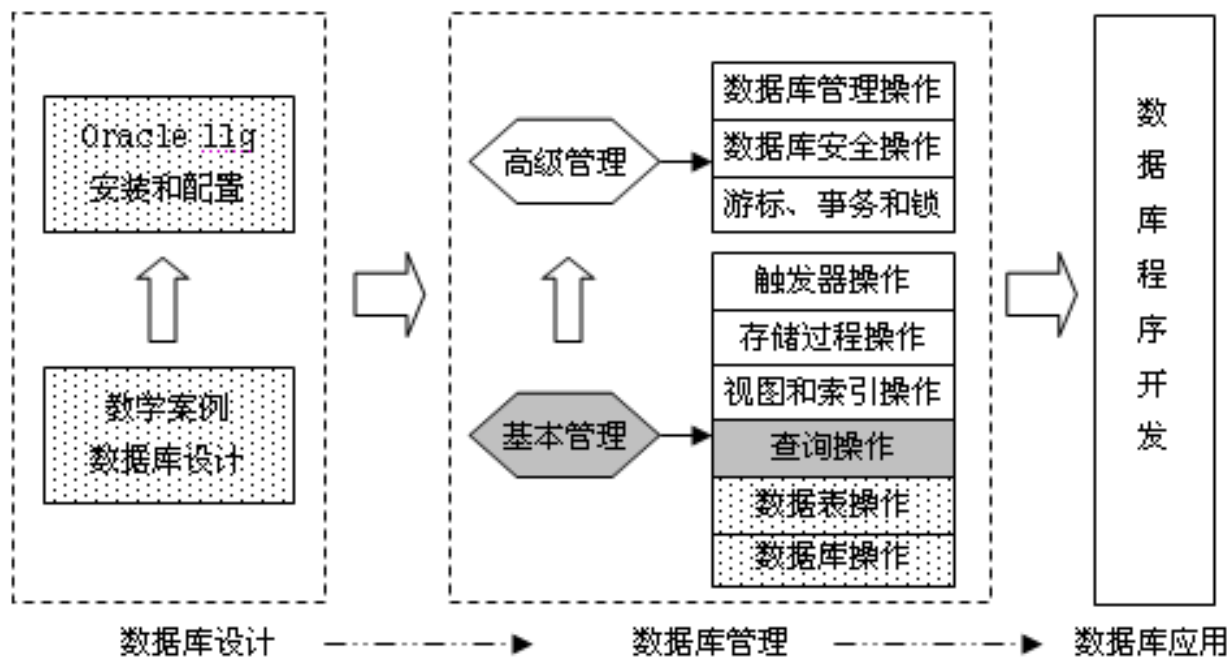


# 第 5 章 数据查询操作

主编：刘志成



# 本章学习导航



本章学习导航



# 本章学习要点

- 
- ( 1 ) 查询的**基本语法**。
  - ( 2 ) **简单查询**的形式和实现。
  - ( 3 ) **内连接查询**的形式和实现。
  - ( 4 ) **外连接查询**的形式和实现。
  - ( 5 ) **子查询**的实现和应用。
  - ( 6 ) **联合查询**及其应用。

建议课时： 6 课时



# 5.1 查询的基本语法

## 查询基本语法

- PL/SQL 的数据查询是一种从数据库中检索符合搜索条件的记录生成数据记录集合，并把它们存入数据记录集对象中的操作。

```
SELECT          <选择表达式>
FROM            [用户方案.]<表或视图>
[ WHERE        <查询条件> ]
[ GROUP BY    <分组表达式>
[ HAVING      <分组统计条件> ] ]
[ ORDER BY    <排序表达式> ]
```



## 5.2 简单查询

### 选择列

- 1 . 选择所有列
- 2 . 选择指定列
- 3 . 计算列
- 4 . 使用别名

- 【例 1-1】 查询商品的所有信息。

```
SELECT * FROM SCOTT.Goods;
```

- 【例 1-2】 网站管理人员在了解商品信息时只需要了解所有商品的商品号、商品名称和商品单价。

```
SELECT g_ID, g_Name, g_Price FROM SCOTT.Goods
```

- 【例 1-3】 在 Goods 表中存储有商品数量和商品单价，现在需要了解所有商品的商品号、商品名称和商品总额。

```
SELECT g_ID, g_Name, g_Price*g_Number FROM Goods
```





## 5.2 简单查询

### 选择列

- 【例 1-4】 要求了解所有商品的商品号、商品名称和总价值，但希望分别以汉字标题商品号、商品名称和总价值表示 g\_ID、g\_Name 和 g\_Price\*g\_Number。

```
SELECT g_ID 商品号, g_Name 商品名称, g_Price*g_Number 总价值  
FROM SCOTT.Goods
```

- 【例 1-5】 显示所有会员的编号、姓名、性别和年龄

```
SELECT c_ID 编号, c_NAME 姓名, c_GENDER 性  
别, TO_CHAR(SYSDATE, 'YYYY') - TO_CHAR(c_BIRTH, 'YYYY') AS  
年龄 FROM SCOTT.CUSTOMERS;
```





## 5.2 简单查询

### 选择行 - 条件查询

- 【例 2-1】 需要了解所有商品中的“热点”商品的所有信息。

```
SELECT * FROM SCOTT.Goods WHERE g_Status = '热点 '
```

- 【例 2-2】 需要了解商品类别为“01”，商品单价在2500元以上的商品信息，要求以汉字标题显示商品号、商品名称、商品类别号和价格。

```
SELECT g_ID 商品号 ,g_Name 商品名称 ,t_ID 类别号 ,g_Price 价格  
FROM Goods WHERE t_ID='01' AND g_Price>2500
```

- 【例 2-3】 需要了解所有年龄在20 ~ 25岁之间的会员的名称和年龄（用NI表示，不是基本表中的字段，是计算出来的列）。

```
SELECT c_Name, TO_CHAR(SYSDATE,'YYYY') -  
TO_CHAR(c_BIRTH,'YYYY') NI  
FROM Customers  
WHERE TO_CHAR(SYSDATE,'YYYY') - TO_CHAR(c_BIRTH,'YYYY')  
BETWEEN 20 AND 25
```



## 5.2 简单查询

### 选择行 - 条件查询 - 使用 IN

- 【例 2-5】 需要了解来自“湖南株洲”和“湖南长沙”两地会员的详细信息。

```
SELECT c_ID,c_Name,c_Address
```

```
FROM Customers
```

```
WHERE SUBSTR(c_Address,1,4) IN ('湖南株洲','湖南长沙')
```

- 【例 2-6】 需要了解家庭地址不是“湖南株洲”和“湖南长沙”的商品的详细信息。

```
SELECT c_ID,c_Name,c_Address
```

```
FROM Customers
```

```
WHERE SUBSTR(c_Address,1,4) NOT IN ('湖南株洲','湖南长沙')
```



## 5.2 简单查询

### 选择行 - 条件查询 - 使用通配符

| 通配符  | 举例                 | 说明                                   |
|------|--------------------|--------------------------------------|
| _    | 'T00_'             | 长度为 4，前 3 个字符为“T00”，第 4 个字符为任意字符的字符串 |
| %    | 'T%'               | 首字符为“T”，后面为任意多个字符的字符串                |
| []   | '[PT]001'          | 首字符为“P”或“T”，其后为“001”的字符串             |
|      | '[P-T]001'         | 首字符为“P”至“T”之间的任意字符，其后为“001”的字符串      |
|      | '[%]001'           | 首字符为“%”，其后为“001”的字符串                 |
| [^]  | '[^PT]001'         | 首字符不是“P”或“T”，其后为“001”的字符串            |
| 换码字符 | 'T\_00' ESCAPE '\' | 首字符为“T”，第二个字符为“_”，其后为“01”的字符串        |

- **【例 2-7】** 需要了解所有商品中以“三星”两个字开头的商品的详细信息。

```
SELECT * FROM Goods WHERE g_Name LIKE '三星 %'
```

- **【例 2-8】** 需要了解姓“黄”且名字中只有两个汉字的会员的会员名、真实姓名、电话和电子邮箱。

```
SELECT c_Name, c_TrueName, c_Phone, c_E-mail  
FROM Customers  
WHERE c_TrueName LIKE '黄_'
```



## 5.2 简单查询

### 选择行 - 条件查询 - 空值判断

- 对于那些允许空值的列，可以使用 **IS NULL** 或 **IS NOT NULL** 来判断其值是否为空。对于使用 **IS NULL** 的查询表达式，如果返回值为逻辑真，则说明当前数据记录对应列的值为空，否则为非空。**IS NOT NULL** 的含义与 **IS NULL** 恰好相反。

- 【例 2-10】 查询暂时没有商品图片的商品信息

```
SELECT *  
FROM Goods  
WHERE g_Image IS NULL
```



## 5.2 简单查询

### 选择行 - 使用 DISTINCT

- 【例 2-11】 需要了解在 WebShop 网站进行了购物并下了订单的会员编号。

```
SELECT g_ID FROM OrderDetails
```

```
SELECT DISTINCT g_ID FROM OrderDetails
```

```
G_ID
-----
010008
010007
010001
040001
020002
010006
060001
030003

8 rows selected
```

```
G_ID
-----
010001
060001
060001
010007
040001
010008
030003
020002
040001
060001
010006
010006

12 rows selected
```

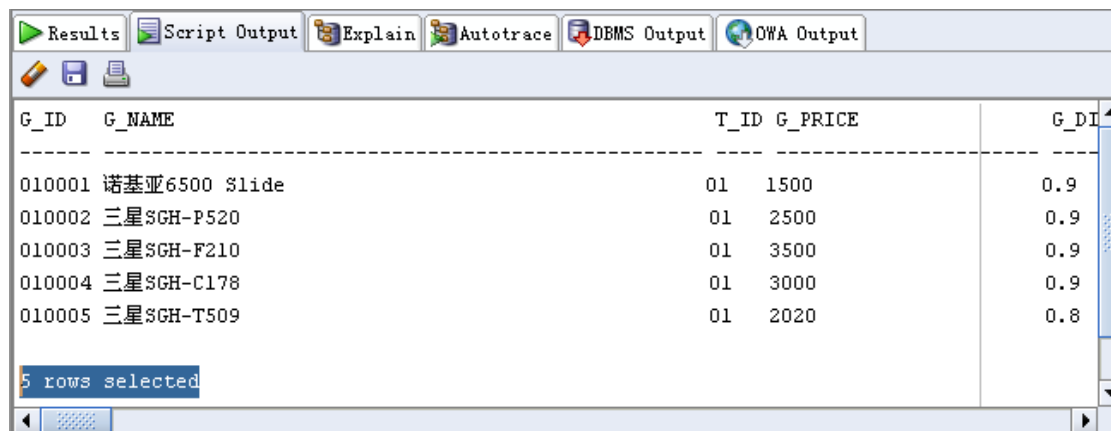
## 5.2 简单查询

### 选择行 - 使用 ROWNUM

- 【例 2-12】查询商品表中前 5 条商品的详细信息。

```
SELECT      *  
FROM        SCOTT.GOODS  
WHERE       ROWNUM <= 5;
```

在 eBuy 电子商城中，“新品推荐”功能中的商品就可以通过这种方式进行选择



| G_ID   | G_NAME        | T_ID | G_PRICE | G_DT |
|--------|---------------|------|---------|------|
| 010001 | 诺基亚6500 Slide | 01   | 1500    | 0.9  |
| 010002 | 三星SGH-P520    | 01   | 2500    | 0.9  |
| 010003 | 三星SGH-F210    | 01   | 3500    | 0.9  |
| 010004 | 三星SGH-C178    | 01   | 3000    | 0.9  |
| 010005 | 三星SGH-T509    | 01   | 2020    | 0.8  |

5 rows selected



## 5.2 简单查询

### 选择行 - 查询结果排序

- 在输出查询数据记录集时，默认的数据记录显示顺序是按这些数据记录在原表中的逻辑排列顺序。如果需要定制查询数据记录集的输出顺序，则可以使用 **ORDER BY** 子句来实现。 **ORDER BY** 子句能够按照指定的一个或多个列（表达式）的升序或降序来重新排列查询数据记录集的输出顺序。

- 【例 3-1】需要了解商品类别号为 “01” 的商品的商品号、商品名称和商品单价，并要求根据商品的价格进行降序（价格由高到低）排列。

```
SELECT g_ID, g_Name, g_Price  
FROM SCOTT.Goods  
WHERE t_ID='01'  
ORDER BY g_Price DESC
```



## 5.2 简单查询

### 选择行 - 查询结果排序

- 【例 3-2】在 “【例 3-1】” 中，如果商品的价格相同，要求根据商品名称进行升序排列。

```
SELECT g_ID, g_Name, g_Price  
FROM SCOTT.Goods  
WHERE t_ID='01'  
ORDER BY g_Price DESC,g_Name ASC
```

```
SELECT g_ID, g_Name, g_Price  
FROM SCOTT.Goods  
WHERE t_ID='01'  
ORDER BY 3 DESC,2 ASC;
```

#### 【说明】

- 由于 ASC 为排序默认关键字，该处的 ASC 关键字也可以省略；
- 除了使用列名进行排序以外，还可以使用列位置编号进行排序。





## 5.2 简单查询

### 选择行 - 查询结果分组

- 有时候需要对查询数据记录集按列或表达式进行分组，以利于分析数据，此时可以通过使用 **GROUP BY** 子句来实现。如果需要在分组的基础上进行组的过滤，则可以结合 **GROUP BY** 子句再使用 **HAVING** 子句来实现。对于需要对查询数据集进行汇总以生成统计报表的情况，则需要使用 **COMPUTE** 子句。

表 5-2 PL/SQL 中常用的聚合函数

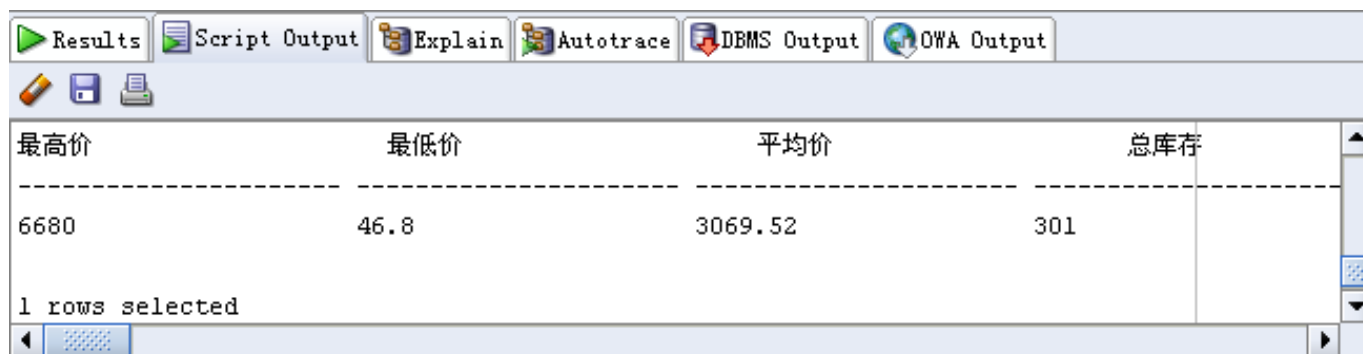
| 函数名          | 说明                                         |
|--------------|--------------------------------------------|
| AVG(列或表达式)   | 返回列或表达式的平均值                                |
| COUNT(列或表达式) | 返回列或表达式的出现数量，如果列或表达式使用 “*”，则表示返回查询数据记录集的行数 |
| MAX(列或表达式)   | 返回列或表达式的最大值                                |
| MIN(列或表达式)   | 返回列或表达式的最小值                                |
| SUM(列或表达式)   | 返回列或表达式的所有值之和                              |

## 5.2 简单查询

### 选择行 - 查询结果分组 - 使用聚合函数

- 【例 4-1】 查询所有商品的最高价、最低价、平均价和所有库存量之和。

**SELECT MAX(g\_Price) 最高价, MIN(g\_Price) 最低价, AVG(g\_Price) 平均价, SUM(g\_Number) 总库存**  
**FROM SCOTT.Goods;**



| 最高价  | 最低价  | 平均价     | 总库存 |
|------|------|---------|-----|
| 6680 | 46.8 | 3069.52 | 301 |

1 rows selected



## 5.2 简单查询

### 选择行 - 查询结果分组 - 使用 GROUP

- 【例 4-2】 需要了解每一类别的商品总数。

```
SELECT t_ID 类别号, COUNT(t_ID) 商品数
FROM SCOTT.Goods
GROUP BY t_ID;
```

执行该 **PL/SQL** 查询语句，将先按商品类别编号 **t\_ID** 进行分组，相同商品类别编号的数据记录位于同一个组内，然后通过使用聚合函数进行统计，计算出每一组内具有的商品编号数目。

| 类别号 | 商品数 |
|-----|-----|
| 04  | 1   |
| 01  | 8   |
| 02  | 2   |
| 03  | 3   |
| 06  | 1   |

5 rows selected



## 5.2 简单查询

### 选择行 - 查询结果分组 - 使用 GROUP

- 【例 4-3】 查询商品单价超过 2000 的商品类别编号和库存量。

对于该例，可以考虑分两步执行：**第一步**，使用 WHERE 条件过滤掉价格在 2000 元以下的所有商品记录，被过滤掉的数据记录不再参与后面的操作；**第二步**，按商品编号对过滤后的数据记录集进行分组，得到的分组数据记录集即为最终所需要的结果。

```
SELECT t_ID, SUM(g_Number) 库存量
FROM   SCOTT.Goods
WHERE  g_PRICE >=2000
GROUP BY t_ID;
```

## 5.2 简单查询

### 选择行 - 查询结果分组 - 使用 GROUP

- 如果需要使查询数据记录集生成数据统计，以及横向小计统计，则需要在 **GROUP BY** 子句中使用 **ROLLUP** 关键字。

- 【例 4-4】显示每个商品类别的商品库存量及商品总库存量。

```
SELECT g_ID, t_ID, SUM(g_Number) 库存量
FROM   SCOTT.Goods
GROUP BY ROLLUP(t_ID,g_ID);
```

|                  |           |            |
|------------------|-----------|------------|
|                  | <u>01</u> | <u>125</u> |
| 020001           | 02        | 18         |
| 020002           | 02        | 18         |
|                  | <u>02</u> | <u>36</u>  |
| 030001           | 03        | 10         |
| 030002           | 03        | 15         |
| 030003           | 03        | 10         |
|                  | <u>03</u> | <u>35</u>  |
| 040001           | 04        | 60         |
|                  | <u>04</u> | <u>60</u>  |
| 060001           | 06        | 45         |
|                  | <u>06</u> | <u>45</u>  |
|                  |           | 301        |
| 21 rows selected |           |            |



## 5.2 简单查询

### 选择行 - 查询结果分组 - 使用 GROUP

- 如果需要对查询数据记录集生成数据统计、横向小计及纵向小计结果，则可以在 **GROUP BY** 子句中使用 **CUBE** 关键字。

- **【例 4-5】** 显示商品总库存量、每个种类商品的商品总量和每一商品的数量。

```
SELECT g_ID, t_ID, SUM(g_Number) 库存量
FROM   SCOTT.Goods
GROUP BY CUBE(g_ID,t_ID);
```

| G_ID      | T_ID | 库存量 |
|-----------|------|-----|
| -----     |      |     |
|           |      | 301 |
| 01        |      | 125 |
| 02        |      | 36  |
| 03        |      | 35  |
| 04        |      | 60  |
| 06        |      | 45  |
| 010001    |      | 20  |
| 010001 01 |      | 20  |
| 010002    |      | 10  |
| 010002 01 |      | 10  |
| 010003    |      | 30  |
| 010003 01 |      | 30  |



## 5.2 简单查询

### 选择行 - 查询结果分组 - 使用 GROUP

- 在对查询数据记录集进行分组的基础上，再对每组数据记录集进行过滤时，不能使用 **WHERE** 子句来进行过滤，而应该使用 **HAVING** 子句进行过滤，因为 **WHERE** 子句在分组之前执行过滤，而 **HAVING** 则在分组之后执行过滤。

- **【例 4-6】** 需要了解订单总额大于 5000 的订单信息，并按升序排列。

```
SELECT o_ID 订单编号, sum(d_Price*d_Number) 总金额  
FROM SCOTT.OrderDetails  
GROUP BY o_ID  
HAVING sum(d_Price*d_Number)>5000  
ORDER BY sum(d_Price*d_Number)
```





## 5.3 连接查询

### 内连接查询

实际上，数据库实例中的各个表之间可能存在某些**内在关联**，通过这些关联，可以为应用程序提供一些涉及多个表的复杂信息，如主表和外表之间就存在主键和外键的关联。**PL/SQL** 为这种多个表之间存在关联的查询提供了检索数据的方法，称为**连接查询**。

内连接是使用比较运算符作为连接条件的连接方式。内连接作为一种典型的默认连接方式，关键字 **INNER** 默认提供。使用内连接方式时，只有那些满足连接条件的数据记录被显示，不满足连接条件的数据记录将不被显示。

根据连接条件中的关系运算符是否使用“**=**”，内连接可以分为**等值连接**和**非等值连接**。若用于连接的两个表或视图来源于同一个表或视图，这样的内连接也被称之为**自连接**。



## 5.3 连接查询

### 等值连接

- **【例 5-1】** 需要了解每个商品的商品号、商品名称和商品类别名称。

**【分析】** 商品基本信息存放在 Goods 表中，商品分类信息存放在 Types 表，所以本查询实际上同时涉及 Goods 与 Types 两个表中的数据。这两个表之间的联系是通过两个表都具有的属性 t\_ID 实现的。要查询商品及其类别名称，就必须将这两个表中商品号相同的记录连接起来，这是一个等值连接。

```
SELECT Goods.g_ID, Goods.t_ID, Types.t_Name, Goods.g_Name  
FROM SCOTT.Goods  
JOIN SCOTT.Types  
ON SCOTT.Goods.t_ID= SCOTT.Types.t_ID;
```





## 5.3 连接查询

### 等值连接

- **【例 5-2】** 需要了解所有订单中订购的商品信息（商品名称、购买价格和购买数量）和订单日期。

**【分析】**在“订单表”中存放了订单号和订单产生日期等信息，而该订单所购买的商品的信息（商品号、购买价格和购买数量）存放在“订单详情”表中，商品的名称存放在“商品表”中，因此，订单表需要和订单详情表通过订单号进行连接以获得订单中所购商品的商品号等信息，而订单详情表需要和商品表进行连接以通过商品号获得商品名称信息。主要涉及三个表的查询。

```
SELECT Orders.o_ID,o_Date,g_Name,d_Price,d_Number
FROM SCOTT.Orders
JOIN SCOTT.OrderDetails
ON Orders.o_ID=OrderDetails.o_ID
JOIN Goods
ON OrderDetails.g_ID=Goods.g_ID;
```



## 5.3 连接查询

### 自身连接

连接操作一般在两个表之间进行，也可以在一个表与其自身之间进行连接，这样的连接操作称为**自连接**。为了分别表示一个表与其自身，需要引入表的别名。

● **【例 5-3】** 需要了解不低于“三星 SGH-C178”价格的商品号、商品名称和商品单价，查询后的结果要求按商品单价升序排列。

```
SELECT G2.g_ID 商品号 ,G2.g_Name 商品名称 ,G2.g_Price 价格  
FROM SCOTT.Goods G1  
JOIN SCOTT.Goods G2  
ON G1.g_Name='三星 SGH-C178' AND G1.g_Price<=G2.g_Price  
ORDER By G2.g_Price;
```



## 5.3 连接查询

### 外连接查询 - 左外连接

- 查询数据记录集包含来自一个表的所有数据记录和另一个表中的匹配数据记录的连接称为左外连接。对于左外连接，**第一个表**中的所有数据记录将被显示，**第二个表**（匹配表）如果找不到相匹配的数据记录，相应的列将显示为空值（**NULL**），否则显示匹配数据记录。

- **【例 5-4】** 需要了解所有商品类别及其对应商品信息，如果该商品类别没有对应商品也需要显示其类别信息。

**【分析】** 将 **Types** 表和 **Goods** 表进行左外连接，**Types** 为左表，**Goods** 表为右表。完成语句如下所示。

```
SELECT Types.t_ID, t_Name, g_ID, g_Name, g_Price, g_Number  
FROM SCOTT.Types  
LEFT OUTER JOIN SCOTT.Goods on Types.t_ID= Goods.t_ID;
```



## 5.3 连接查询

### 外连接查询 - 右外连接

- 查询数据记录集包含来自第二个表的所有数据记录和第一个表中的匹配数据记录的连接称为右外连接。对于右外连接，第二个表中的所有数据记录将被显示，第一个表（匹配表）如果找不到相匹配的数据记录，相应的列将显示为空值（**NULL**），否则显示匹配数据记录。

- **【例 5-5】** 需要了解所有商品的信息（即使是不存在对应的商品类别信息，实际上这种情况是不存在的）。

**【分析】** 将 **Types** 表和 **Goods** 表进行右外连接，**Goods** 为左表，**Types** 表为右表。完成语句如下所示。

```
SELECT Types.t_ID, t_Name, g_ID, g_Name, g_Price, g_Number  
FROM SCOTT.Types  
RIGHT OUTER JOIN SCOTT.Goods on Types.t_ID= Goods.t_ID;
```



## 5.3 连接查询

### 外连接查询 - 完全外部连接

- 查询数据记录集的两个连接表中所有行的连接操作称为完全外部连接。对于完全外部连接，两个连接表无论是否匹配，它们的数据记录都将被显示。

- 【例 5-6】需要了解所有商品的基本信息和类别信息。

【分析】在 **Types** 表和 **Goods** 表之间的建立完整外部连接。完成语句如下所示。

```
SELECT Types.t_ID, t_Name, g_ID, g_Name, g_Price, g_Number  
FROM SCOTT.Types  
FULL OUTER JOIN SCOTT.Goods on Types.t_ID= Goods.t_ID;
```



## 5.3 连接查询

### 外连接查询 - 交叉连接查询

- 交叉连接是使用 **CROSS** 关键字进行的连接，它的输出为笛卡儿积，即第一个表的每一条数据记录与第二个表的每一条数据记录进行连接。笛卡儿积的结果通常很大，其数据记录数目等于两个表的数据记录数目之积，数据记录的列数等于两个表的列数之和。

- 对商品信息表和商品类别表进行交叉连接。其完成语句有下面两种。

[ 语句一 ]

```
SELECT * FROM SCOTT.Types
```

```
CROSS JOIN SCOTT.Goods
```

或

[ 语句二 ]

```
SELECT Types.*,Goods.*
```

```
FROM SCOTT.Types,SCOTT.Goods
```



## 5.4 子查询

### IN子查询

- 将一个查询语句嵌套在另一个查询语句中的查询称为嵌套查询或子查询。被嵌入在其他查询语句中的查询语句称为子查询语句，子查询语句的载体查询语句称为父查询语句。子查询语句一般嵌入在另一个查询语句的 **WHERE** 子句或 **HAVING** 子句中，另外，子查询语句也可以嵌入在一个数据记录更新语句的 **WHERE** 子句中。

- 对列表操作的子查询通过 **IN** 关键字实现父查询和子查询之间的连接，判断指定列的值是否出现在子查询的查询数据记录集中。使用 **IN** 的子查询语句返回的查询数据记录集一般由单列多行值组成，这也是子查询中最常用的一种形式。

对于使用 **IN** 的子查询的连接条件，其语法格式为：

**WHERE 表达式 [ NOT ] IN (子查询)**





## 5.4 子查询

### IN子查询

●【例 6-1】需要了解和“摩托罗拉 W380”为同类商品的商品号、商品名称和类别号。

【分析】要查询与“摩托罗拉 W380”的同类商品，首先要知道“摩托罗拉 W380”的商品类别，再根据该类别获取同类商品的相关信息。

(1) 确定“摩托罗拉 W380”所属类别名。

```
SELECT t_ID FROM SCOTT.Goods  
WHERE g_Name='摩托罗拉 W380'
```

(2) 查找类别号为‘01’的商品信息。

```
SELECT g_ID, g_Name ,t_ID  
FROM SCOTT.Goods WHERE t_ID='01';
```

使用子查询的语句如下。

```
SELECT g_ID, g_Name ,t_ID  
FROM SCOTT.Goods  
WHERE t_ID IN (SELECT t_ID FROM Goods WHERE g_Name='摩  
托罗拉 W380');
```



## 5.4 子查询

### IN子查询

- 【例 6-2】需要了解购买了“红双喜牌乒乓球拍”的订单号、订单时间和订单总金额。

```
SELECT o_ID, o_Date, o_Sum
FROM SCOTT.Orders WHERE o_ID IN
(SELECT o_ID FROM OrderDetails WHERE g_ID IN
(SELECT g_ID FROM Goods WHERE g_Name='红双喜牌乒乓球拍'));
```

- 【例 6-3】需要了解购买了商品号为“060001”的会员 e\_ID（会员号）、e\_Name（会员名称）和 e\_Address（籍贯）。

```
SELECT c_ID, c_Name, c_Address
FROM SCOTT.Customers WHERE c_ID IN
(SELECT c_ID
FROM Orders
JOIN OrderDetails
ON Orders.o_ID=OrderDetails.o_ID
WHERE g_ID= '060001');
```



## 5.4 子查询

### 比较运算符子查询

- 子查询也可以使用比较运算符引入。此时，子查询结果为一个单行单列的值，并可以在父查询中通过比较运算符（“>”、“>=”、“<”、“<=”、“=”、“!=”或“<>”）连接子查询，如果子查询返回不止一个值，整个查询语句将会产生错误。
- 【例 6-4】 需要了解购买了“红双喜牌乒乓球拍”的订单号、订单时间和订单总金额。（使用“=”完成）

```
SELECT o_ID, o_Date, o_Sum
```

```
FROM Orders
```

```
WHERE o_ID IN
```

```
(SELECT o_ID FROM OrderDetails WHERE g_ID =
```

```
(SELECT g_ID FROM Goods WHERE g_Name='红双喜牌乒乓球拍')
```

```
);
```



## 5.4 子查询

### ANY 或 ALL 子查询

- 【例 6-5】需要了解比籍贯为“湖南长沙”任一会员年龄小的会员信息，查询结果按降序排列。

【分析】比任一会员的年龄小，即比最小的还要小。反过来，如果是大于 ALL，则要比最大的还要大，完成语句如下所示。

```
SELECT c_ID, c_Name, TO_CHAR(SYSDATE)-TO_CHAR(c_Birth)
Age, c_Address
FROM Customers
WHERE SUBSTR(c_Address,1,4) < > '湖南长沙' AND c_Birth > ALL
(SELECT c_Birth FROM Customers WHERE
SUBSTR(c_Address,1,4)='湖南长沙')
ORDER BY Age DESC
```



## 5.4 子查询

### EXIST 子查询

● 【例 6-6】 针对 Employees 表中的每一名员工，在 Orders 表中查找处理过订单并且送货模式为“邮寄”的所有订单信息。

【分析】 第一步要查找处理过订单的员工编号，第二步再根据员工处理订单的送货模式显示订单详细信息。

```
SELECT *  
FROM Orders  
WHERE o_Sendmode = ' 邮寄 '  
AND EXISTS (SELECT e_ID FROM Employees AS Emp WHERE  
Emp.e_ID = Orders.e_ID)
```



## 5.4 子查询

### INSERT 子查询

- **INSERT INTO** 语句不但可以将一条数据记录插入到表中，也可以通过使用子查询的形式将一个数据记录集插入到表中，实现批量插入。

- **【例 6-7】** 求每一类商品的平均价格，并将结果保存到数据库中。

( 1 ) 在数据库中建立一个有两个属性列的新表，其中一列存放类别名，另一列存放相应类别的商品平均价格。

```
CREATE TABLE AvgGoods(t_ID CHAR(2),a_avg FLOAT) ;
```

其中 t\_ID 代表商品类别号， a\_avg 代表平均价格。

( 2 ) 对数据库的商品表按商品号分组求平均价格，再把商品号和平均价格存入新表中。

```
INSERT INTO AvgGoods (t_ID, a_avg)
```

```
SELECT t_ID, AVG(g_Price)
```

```
FROM Goods GROUP BY t_ID;
```

( 3 ) 查看表 AvgGoods 表中的记录。

```
SELECT * FROM AvgGoods;
```



## 5.4 子查询

### UPDATE 子查询

- 在 **UPDATE** 语句中使用子查询，可以通过构造复杂的更新操作条件来实现更新数据记录的操作。

- **【例 6-8】** 将商品中类别名称为 “家用电器” 的商品折扣修改为 0.8

。

**UPDATE Goods**

**SET g\_Discount=0.8**

**WHERE '家用电器' =**

**(SELECT t\_Name FROM Types WHERE Goods.t\_ID=Types.t\_ID);**

再使用查询语句：

**SELECT g\_ID,g\_Name,t\_ID,g\_Discount FROM GOODS;**





## 5.4 子查询

### DELETE 子查询

- 在 **DELETE FROM** 语句中使用子查询，可以通过构造复杂删除操作条件来实现删除数据记录的操作。
- 【例 6-10】删除类别名称为“家用电器”的商品的基本信息。

**DELETE**

**FROM Goods**

**WHERE '家用电器' =**

**(SELECT t\_Name FROM Types WHERE Goods.t\_ID=Types.t\_ID)**

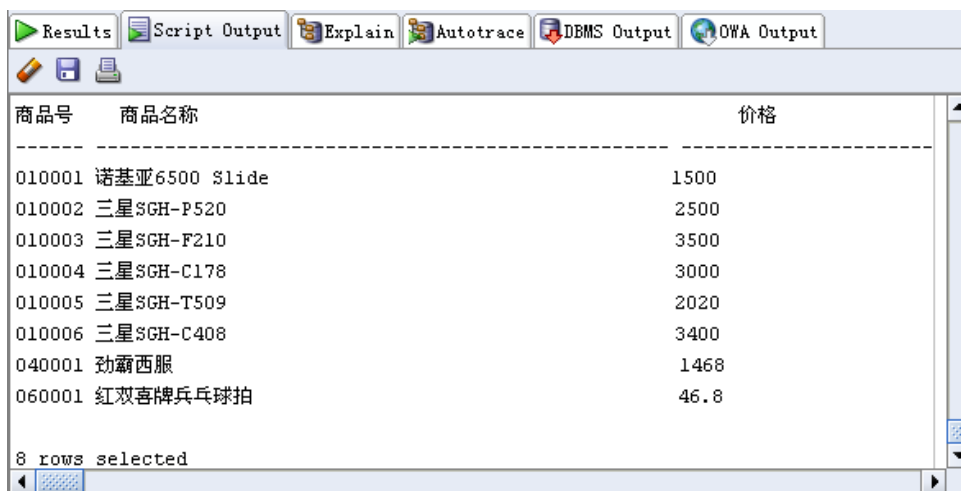


## 5.5 联合查询

### 联合查询

需要了解“三星”的商品以及价格不高于 2000 的商品。完成语句如下所示。

```
SELECT g_ID 商品号 ,g_Name 商品名称 ,g_Price 价格
FROM Goods
WHERE SUBSTR(g_Name,1,2)='三星 '
UNION
SELECT g_ID 商品号 ,g_Name 商品名称 ,g_Price 价格
FROM Goods
WHERE g_Price<2000 ;
```



| 商品号    | 商品名称          | 价格   |
|--------|---------------|------|
| 010001 | 诺基亚6500 Slide | 1500 |
| 010002 | 三星SGH-P520    | 2500 |
| 010003 | 三星SGH-F210    | 3500 |
| 010004 | 三星SGH-C178    | 3000 |
| 010005 | 三星SGH-T509    | 2020 |
| 010006 | 三星SGH-C408    | 3400 |
| 040001 | 劲霸西服          | 1468 |
| 060001 | 红双喜牌乒乓球拍      | 46.8 |

8 rows selected



## 任务 1

在已经创建好的 **BookData** 数据库基础上，完成下列查询

- (1) 查询书名中包含有“程序设计”字样的图书详细信息。
- (2) 查询书名中包含有“程序设计”字样，出版社编号为“003”的图书详细信息。
- (3) 查询出版社编号为“005”，价格在 15 ~ 25 元之间的图书详细信息。
- (4) 查询编者信息中包含有“刘志成”，出版时间在 2006 年 1 月 1 日到 2009 年 10 月 1 日之间的图书详细信息。
- (5) 查询书名中包含有“程序设计”字样，出版社名称为“清华大学出版社”的图书详细信息。
- (6) 查询读者“王周应”借阅的图书的存放位置。
- (7) 查询到当前日期为止未还的图书名称和借书人。





课外实践

## 任务 2

根据 china-pub 网站提供的查询功能，尝试各种查询操作，并编写对应的查询语句。

