**游标的使用**

    在 PL/SQL 程序中，对于处理多行记录的事务经常使用游标来实现。

**4.1 游标概念**

  在PL/SQL块中执行SELECT、INSERT、DELETE和UPDATE语句时，ORACLE会在内存中为其分配上下文区（Context Area），即缓冲区。游标是指向该区的一个指针，或是命名一个工作区（Work Area），或是一种结构化数据类型。它为应用等量齐观提供了一种对具有多行数据查询结果集中的每一行数据分别进行单独处理的方法，是设计嵌入式SQL语句的应用程序的常用编程方式。

 在每个用户会话中，可以同时打开多个游标，其数量由数据库初始化参数文件中的OPEN\_CURSORS参数定义。

对于不同的SQL语句，游标的使用情况不同：

|  |  |
| --- | --- |
| SQL语句 | 游标 |
| 非查询语句 | 隐式的 |
| 结果是单行的查询语句 | 隐式的或显示的 |
| 结果是多行的查询语句 | 显示的 |

**4.1.1 处理显式游标**

**1. 显式游标处理**

显式游标处理需四个 PL/SQL步骤:

**定义/声明游标：**就是定义一个游标名，以及与其相对应的SELECT 语句。

格式：

    CURSOR cursor\_name[(parameter[, parameter]…)]   
           [RETURN datatype]  
    IS   
        select\_statement;

游标参数只能为输入参数，其格式为：

parameter\_name [IN] datatype [{:= | DEFAULT} expression]

在指定数据类型时，不能使用长度约束。如NUMBER(4),CHAR(10) 等都是错误的。

[RETURN datatype]是可选的，表示游标返回数据的数据。如果选择，则应该严格与select\_statement中的选择列表在次序和数据类型上匹配。一般是记录数据类型或带“%ROWTYPE”的数据。

**打开游标：**就是执行游标所对应的SELECT 语句，将其查询结果放入工作区，并且指针指向工作区的首部，标识游标结果集合。如果游标查询语句中带有FOR UPDATE选项，OPEN 语句还将锁定数据库表中游标结果集合对应的数据行。

格式：

OPEN cursor\_name[([parameter =>] value[, [parameter =>] value]…)];

在向游标传递参数时，可以使用与函数参数相同的传值方法，即位置表示法和名称表示法。PL/SQL 程序不能用OPEN 语句重复打开一个游标。

**提取游标数据：**就是检索结果集合中的数据行，放入指定的输出变量中。

格式：

FETCH cursor\_name INTO {variable\_list | record\_variable };

执行FETCH语句时，每次返回一个数据行，然后自动将游标移动指向下一个数据行。当检索到最后一行数据时，如果再次执行FETCH语句，将操作失败，并将游标属性%NOTFOUND置为TRUE。所以每次执行完FETCH语句后，检查游标属性%NOTFOUND就可以判断FETCH语句是否执行成功并返回一个数据行，以便确定是否给对应的变量赋了值。

对该记录进行处理；

继续处理，直到活动集合中没有记录；

**关闭游标：**当提取和处理完游标结果集合数据后，应及时关闭游标，以释放该游标所占用的系统资源，并使该游标的工作区变成无效，不能再使用FETCH 语句取其中数据。关闭后的游标可以使用OPEN 语句重新打开。

格式：

CLOSE cursor\_name;

**注：定义的游标不能有INTO 子句。**

**例1.**查询前10名员工的信息。

DECLARE  
   CURSOR c\_cursor   
   IS SELECT first\_name || last\_name, Salary   
   FROM EMPLOYEES   
   WHERE rownum<**11**;     
   v\_ename  EMPLOYEES.first\_name%TYPE;  
   v\_sal    EMPLOYEES.Salary%TYPE;     
BEGIN  
  OPEN c\_cursor;  
  FETCH c\_cursor INTO v\_ename, v\_sal;  
  WHILE c\_cursor%FOUND LOOP  
     DBMS\_OUTPUT.PUT\_LINE(v\_ename||'---'||to\_char(v\_sal) );  
     FETCH c\_cursor INTO v\_ename, v\_sal;  
  END LOOP;  
  CLOSE c\_cursor;  
END;

**例2.**游标参数的传递方法。

DECLARE  
  DeptRec    DEPARTMENTS%ROWTYPE;  
  Dept\_name  DEPARTMENTS.DEPARTMENT\_NAME%TYPE;  
  Dept\_loc   DEPARTMENTS.LOCATION\_ID%TYPE;  
  CURSOR c1 IS   
  SELECT DEPARTMENT\_NAME, LOCATION\_ID FROM DEPARTMENTS   
  WHERE DEPARTMENT\_ID <= **30**;  
    
  CURSOR c2(dept\_no NUMBER DEFAULT **10**) IS  
    SELECT DEPARTMENT\_NAME, LOCATION\_ID FROM DEPARTMENTS   
    WHERE DEPARTMENT\_ID <= dept\_no;  
  CURSOR c3(dept\_no NUMBER DEFAULT **10**) IS   
    SELECT \* FROM DEPARTMENTS   
    WHERE DEPARTMENTS.DEPARTMENT\_ID <=dept\_no;  
BEGIN  
  OPEN c1;  
  LOOP  
    FETCH c1 INTO dept\_name, dept\_loc;  
    EXIT WHEN c1%NOTFOUND;  
        DBMS\_OUTPUT.PUT\_LINE(dept\_name||'---'||dept\_loc);  
    END LOOP;  
    CLOSE c1;  
  
    OPEN c2;  
    LOOP  
        FETCH c2 INTO dept\_name, dept\_loc;  
        EXIT WHEN c2%NOTFOUND;  
        DBMS\_OUTPUT.PUT\_LINE(dept\_name||'---'||dept\_loc);  
    END LOOP;  
    CLOSE c2;  
  
    OPEN c3(dept\_no =>**20**);  
    LOOP  
        FETCH c3 INTO deptrec;  
        EXIT WHEN c3%NOTFOUND;  
        DBMS\_OUTPUT.PUT\_LINE(deptrec.DEPARTMENT\_ID||'---'||deptrec.DEPARTMENT\_NAME||'---'||deptrec.LOCATION\_ID);  
    END LOOP;  
    CLOSE c3;  
END;

**2.游标属性**

 Cursor\_name%FOUND     布尔型属性，当最近一次提取游标操作FETCH成功则为 TRUE,否则为FALSE；

 Cursor\_name%NOTFOUND   布尔型属性，与%FOUND相反；

 Cursor\_name%ISOPEN     布尔型属性，当游标已打开时返回 TRUE；

 Cursor\_name%ROWCOUNT   数字型属性，返回已从游标中读取的记录数。

**例3：**给工资低于1200 的员工增加工资50。

DECLARE  
   v\_empno  EMPLOYEES.EMPLOYEE\_ID%TYPE;  
   v\_sal      EMPLOYEES.Salary%TYPE;  
   CURSOR c\_cursor IS SELECT EMPLOYEE\_ID, Salary FROM EMPLOYEES;   
BEGIN  
   OPEN c\_cursor;  
   LOOP  
      FETCH c\_cursor INTO v\_empno, v\_sal;  
      EXIT WHEN c\_cursor%NOTFOUND;   
      IF v\_sal<=**1200** THEN  
            UPDATE EMPLOYEES SET Salary=Salary+**50** WHERE EMPLOYEE\_ID=v\_empno;  
            DBMS\_OUTPUT.PUT\_LINE('编码为'||v\_empno||'工资已更新!');  
      END IF;  
   DBMS\_OUTPUT.PUT\_LINE('记录数:'|| c\_cursor %ROWCOUNT);  
   END LOOP;  
   CLOSE c\_cursor;  
END;

**例4：**没有参数且没有返回值的游标。

DECLARE  
   v\_f\_name employees.first\_name%TYPE;  
   v\_j\_id   employees.job\_id%TYPE;  
   CURSOR c1       --声明游标,没有参数没有返回值  
   IS  
      SELECT first\_name, job\_id FROM employees   
      WHERE department\_id = **20**;  
BEGIN  
   OPEN c1;        --打开游标  
   LOOP  
      FETCH c1 INTO v\_f\_name, v\_j\_id;    --提取游标  
      IF c1%FOUND THEN  
         DBMS\_OUTPUT.PUT\_LINE(v\_f\_name||'的岗位是'||v\_j\_id);  
      ELSE  
         DBMS\_OUTPUT.PUT\_LINE('已经处理完结果集了');  
         EXIT;  
      END IF;  
   END LOOP;  
   CLOSE c1;   --关闭游标  
END;

**例5：**有参数且没有返回值的游标。

DECLARE  
   v\_f\_name employees.first\_name%TYPE;  
   v\_h\_date employees.hire\_date%TYPE;  
   CURSOR c2(dept\_id NUMBER, j\_id VARCHAR2) --声明游标,有参数没有返回值  
   IS  
      SELECT first\_name, hire\_date FROM employees  
      WHERE department\_id = dept\_id AND job\_id = j\_id;  
BEGIN  
   OPEN c2(**90**, 'AD\_VP');  --打开游标,传递参数值  
   LOOP  
      FETCH c2 INTO v\_f\_name, v\_h\_date;    --提取游标  
      IF c2%FOUND THEN  
         DBMS\_OUTPUT.PUT\_LINE(v\_f\_name||'的雇佣日期是'||v\_h\_date);  
      ELSE  
         DBMS\_OUTPUT.PUT\_LINE('已经处理完结果集了');  
         EXIT;  
      END IF;  
   END LOOP;  
   CLOSE c2;   --关闭游标  
END;

**例6：**有参数且有返回值的游标。

DECLARE  
   TYPE emp\_record\_type IS RECORD(  
        f\_name   employees.first\_name%TYPE,  
        h\_date   employees.hire\_date%TYPE);  
   v\_emp\_record EMP\_RECORD\_TYPE;  
  
   CURSOR c3(dept\_id NUMBER, j\_id VARCHAR2) --声明游标,有参数有返回值  
          RETURN EMP\_RECORD\_TYPE  
   IS  
      SELECT first\_name, hire\_date FROM employees  
      WHERE department\_id = dept\_id AND job\_id = j\_id;  
BEGIN  
   OPEN c3(j\_id => 'AD\_VP', dept\_id => **90**);  --打开游标,传递参数值  
   LOOP  
      FETCH c3 INTO v\_emp\_record;    --提取游标  
      IF c3%FOUND THEN  
         DBMS\_OUTPUT.PUT\_LINE(v\_emp\_record.f\_name||'的雇佣日期是'  
                            ||v\_emp\_record.h\_date);  
      ELSE  
         DBMS\_OUTPUT.PUT\_LINE('已经处理完结果集了');  
         EXIT;  
      END IF;  
   END LOOP;  
   CLOSE c3;   --关闭游标  
END;

**例7：**基于游标定义记录变量。

DECLARE  
   CURSOR c4(dept\_id NUMBER, j\_id VARCHAR2) --声明游标,有参数没有返回值  
   IS  
      SELECT first\_name f\_name, hire\_date FROM employees  
      WHERE department\_id = dept\_id AND job\_id = j\_id;  
    --基于游标定义记录变量，比声明记录类型变量要方便，不容易出错  
    v\_emp\_record c4%ROWTYPE;  
BEGIN  
   OPEN c4(**90**, 'AD\_VP');  --打开游标,传递参数值  
   LOOP  
      FETCH c4 INTO v\_emp\_record;    --提取游标  
      IF c4%FOUND THEN  
         DBMS\_OUTPUT.PUT\_LINE(v\_emp\_record.f\_name||'的雇佣日期是'  
                            ||v\_emp\_record.hire\_date);  
      ELSE  
         DBMS\_OUTPUT.PUT\_LINE('已经处理完结果集了');  
         EXIT;  
      END IF;  
   END LOOP;  
   CLOSE c4;   --关闭游标  
END;

**3. 游标的FOR循环**

    PL/SQL语言提供了游标FOR循环语句，自动执行游标的OPEN、FETCH、CLOSE语句和循环语句的功能；当进入循环时，游标FOR循环语句自动打开游标，并提取第一行游标数据，当程序处理完当前所提取的数据而进入下一次循环时，游标FOR循环语句自动提取下一行数据供程序处理，当提取完结果集合中的所有数据行后结束循环，并自动关闭游标。

格式：

   FOR index\_variable IN cursor\_name[(value[, value]…)] LOOP  
    -- 游标数据处理代码  
  END LOOP;

其中：

index\_variable为游标FOR 循环语句隐含声明的索引变量，该变量为记录变量，其结构与游标查询语句返回的结构集合的结构相同。在程序中可以通过引用该索引记录变量元素来读取所提取的游标数据，index\_variable中各元素的名称与游标查询语句选择列表中所制定的列名相同。如果在游标查询语句的选择列表中存在计算列，则必须为这些计算列指定别名后才能通过游标FOR 循环语句中的索引变量来访问这些列数据。

***注：不要在程序中对游标进行人工操作；不要在程序中定义用于控制FOR循环的记录。***

**例8：**

DECLARE  
   CURSOR c\_sal IS SELECT employee\_id, first\_name || last\_name ename, salary  
   FROM employees ;  
BEGIN  
   --隐含打开游标  
   FOR v\_sal IN c\_sal LOOP  
   --隐含执行一个FETCH语句  
      DBMS\_OUTPUT.PUT\_LINE(to\_char(v\_sal.employee\_id)||'---'|| v\_sal.ename||'---'||to\_char(v\_sal.salary)) ;  
   --隐含监测c\_sal%NOTFOUND  
   END LOOP;  
--隐含关闭游标  
END;

**例9：**当所声明的游标带有参数时，通过游标FOR 循环语句为游标传递参数。

DECLARE  
  CURSOR c\_cursor(dept\_no NUMBER DEFAULT **10**)   
  IS  
    SELECT department\_name, location\_id FROM departments WHERE department\_id <= dept\_no;  
BEGIN  
    DBMS\_OUTPUT.PUT\_LINE('当dept\_no参数值为30：');  
    FOR c1\_rec IN c\_cursor(**30**) LOOP        DBMS\_OUTPUT.PUT\_LINE(c1\_rec.department\_name||'---'||c1\_rec.location\_id);  
    END LOOP;  
    DBMS\_OUTPUT.PUT\_LINE(CHR(**10**)||'使用默认的dept\_no参数值10：');  
    FOR c1\_rec IN c\_cursor LOOP        DBMS\_OUTPUT.PUT\_LINE(c1\_rec.department\_name||'---'||c1\_rec.location\_id);  
    END LOOP;  
END;

**例10：**PL/SQL还允许在游标FOR循环语句中使用子查询来实现游标的功能。

BEGIN  
    FOR c1\_rec IN(SELECT department\_name, location\_id FROM departments) LOOP        DBMS\_OUTPUT.PUT\_LINE(c1\_rec.department\_name||'---'||c1\_rec.location\_id);  
    END LOOP;  
END;

**4.1.2 处理隐式游标**

显式游标主要是用于对查询语句的处理，尤其是在查询结果为多条记录的情况下；而对于非查询语句，如修改、删除操作，则由ORACLE 系统自动地为这些操作设置游标并创建其工作区，这些由系统隐含创建的游标称为隐式游标，隐式游标的名字为SQL，这是由ORACLE 系统定义的。对于隐式游标的操作，如定义、打开、取值及关闭操作，都由ORACLE 系统自动地完成，无需用户进行处理。用户只能通过隐式游标的相关属性，来完成相应的操作。在隐式游标的工作区中，所存放的数据是与用户自定义的显示游标无关的、最新处理的一条SQL 语句所包含的数据。

格式调用为： SQL%

**注：INSERT, UPDATE, DELETE, SELECT 语句中不必明确定义游标。**

**隐式游标属性**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **属性** | **值** | **SELECT** | **INSERT** | **UPDATE** | **DELETE** |
| SQL%ISOPEN |  | FALSE | FALSE | FALSE | FALSE |
| SQL%FOUND | TRUE | 有结果 |  | 成功 | 成功 |
| SQL%FOUND | FALSE | 没结果 |  | 失败 | 失败 |
| SQL%NOTFUOND | TRUE | 没结果 |  | 失败 | 失败 |
| SQL%NOTFOUND | FALSE | 有结果 |  | 成功 | 失败 |
| SQL%ROWCOUNT |  | 返回行数，只为1 | 插入的行数 | 修改的行数 | 删除的行数 |

**例11:**删除EMPLOYEES表中某部门的所有员工，如果该部门中已没有员工，则在DEPARTMENT表中删除该部门。

 DECLARE  
    V\_deptno department\_id%TYPE :=&p\_deptno;  
BEGIN  
    DELETE FROM employees WHERE department\_id=v\_deptno;  
    IF SQL%NOTFOUND THEN  
        DELETE FROM departments WHERE department\_id=v\_deptno;  
    END IF;  
END;

**例12:**通过隐式游标SQL的%ROWCOUNT属性来了解修改了多少行。

DECLARE  
   v\_rows NUMBER;  
BEGIN  
--更新数据  
   UPDATE employees SET salary = **30000**  
   WHERE department\_id = **90** AND job\_id = 'AD\_VP';  
--获取默认游标的属性值  
   v\_rows := SQL%ROWCOUNT;  
   DBMS\_OUTPUT.PUT\_LINE('更新了'||v\_rows||'个雇员的工资');  
--回退更新，以便使数据库的数据保持原样  
   ROLLBACK;  
END;

**4.1.3 关于 NO\_DATA\_FOUND 和 %NOTFOUND的区别**

SELECT … INTO 语句触发 NO\_DATA\_FOUND；

当一个显式游标的WHERE子句未找到时触发%NOTFOUND；

当UPDATE或DELETE 语句的WHERE 子句未找到时触发 SQL%NOTFOUND；在提取循环中要用 %NOTFOUND 或%FOUND 来确定循环的退出条件，不要用 NO\_DATA\_FOUND.**4.1.4  使用游标更新和删除数据**

游标修改和删除操作是指在游标定位下，修改或删除表中指定的数据行。这时，要求游标查询语句中必须使用FOR UPDATE选项，以便在打开游标时锁定游标结果集合在表中对应数据行的所有列和部分列。

为了对正在处理(查询)的行不被另外的用户改动，ORACLE 提供一个 FOR UPDATE 子句来对所选择的行进行锁住。该需求迫使ORACLE锁定游标结果集合的行，可以防止其他事务处理更新或删除相同的行，直到您的事务处理提交或回退为止。

**语法：**

 SELECT column\_list FROM table\_list FOR UPDATE [OF column[, column]…] [NOWAIT]

    如果另一个会话已对活动集中的行加了锁，那么SELECT FOR UPDATE操作一直等待到其它的会话释放这些锁后才继续自己的操作，对于这种情况，当加上**NOWAIT**子句时，如果这些行真的被另一个会话锁定，则OPEN立即返回并给出：

ORA-0054 ：resource busy  and  acquire with nowait specified.

如果使用 FOR UPDATE 声明游标，则可在DELETE和UPDATE 语句中使用

WHERE CURRENT OF cursor\_name子句，修改或删除游标结果集合当前行对应的数据库表中的数据行。

**例13：**从EMPLOYEES表中查询某部门的员工情况，将其工资最低定为 1500；

DECLARE   
    V\_deptno employees.department\_id%TYPE :=&p\_deptno;  
    CURSOR emp\_cursor   
  IS   
  SELECT employees.employee\_id, employees.salary   
    FROM employees WHERE employees.department\_id=v\_deptno  
  FOR UPDATE NOWAIT;  
BEGIN  
    FOR emp\_record IN emp\_cursor LOOP  
    IF emp\_record.salary < **1500** THEN  
        UPDATE employees SET salary=**1500**  
    WHERE CURRENT OF emp\_cursor;  
    END IF;  
    END LOOP;  
--    COMMIT;  
END;

**例14：**将EMPLOYEES表中部门编码为90、岗位为AD\_VP的雇员的工资都更新为2000元；

DECLARE  
   v\_emp\_record employees%ROWTYPE;  
   CURSOR c1  
   IS  
      SELECT \* FROM employees FOR UPDATE;  
BEGIN  
   OPEN c1;  
   LOOP  
      FETCH c1 INTO v\_emp\_record;  
      EXIT WHEN c1%NOTFOUND;  
      IF v\_emp\_record.department\_id = **90** AND  
         v\_emp\_record.job\_id = 'AD\_VP'  
      THEN  
         UPDATE employees SET salary = **20000**  
         WHERE CURRENT OF c1;  --更新当前游标行对应的数据行  
      END IF;  
   END LOOP;  
   COMMIT;   --提交已经修改的数据  
   CLOSE c1;  
END;

**4.2 游标变量**

与游标一样，游标变量也是一个指向多行查询结果集合中当前数据行的指针。但与游标不同的是，游标变量是动态的，而游标是静态的。游标只能与指定的查询相连，即固定指向一个查询的内存处理区域，而游标变量则可与不同的查询语句相连，它可以指向不同查询语句的内存处理区域（但不能同时指向多个内存处理区域，在某一时刻只能与一个查询语句相连），只要这些查询语句的返回类型兼容即可。

**4.2.1  声明游标变量**

游标变量为一个指针，它属于参照类型，所以在声明游标变量类型之前必须先定义游标变量类型。在PL/SQL中，可以在块、子程序和包的声明区域内定义游标变量类型。

语法格式为：

TYPE ref\_type\_name IS REF CURSOR  
 [ RETURN return\_type];

其中：ref\_type\_name为新定义的游标变量类型名称；

  return\_type 为游标变量的返回值类型，它必须为记录变量。

在定义游标变量类型时，可以采用强类型定义和弱类型定义两种。强类型定义必须指定游标变量的返回值类型，而弱类型定义则不说明返回值类型。

声明一个游标变量的两个步骤：

步骤一：定义一个REF CURSOU数据类型，如：

TYPE ref\_cursor\_type IS REF CURSOR;

步骤二：声明一个该数据类型的游标变量，如：

cv\_ref REF\_CURSOR\_TYPE;

例：创建两个强类型定义游标变量和一个弱类型游标变量：

DECLARE  
    TYPE deptrecord IS RECORD(  
        Deptno departments.department\_id%TYPE,  
        Dname departments.department\_name%TYPE,  
        Loc departments.location\_id%TYPE  
    );  
    TYPE deptcurtype IS REF CURSOR RETURN departments%ROWTYPE;  
    TYPE deptcurtyp1 IS REF CURSOR RETURN deptrecord;  
    TYPE curtype IS REF CURSOR;  
    Dept\_c1 deptcurtype;  
    Dept\_c2 deptcurtyp1;  
    Cv curtype;

**4.2.2  游标变量操作**

与游标一样，游标变量操作也包括打开、提取和关闭三个步骤。

**1． 打开游标变量**

打开游标变量时使用的是OPEN…FOR 语句。格式为：

OPEN {cursor\_variable\_name | :host\_cursor\_variable\_name}  
FOR select\_statement;

其中：cursor\_variable\_name为游标变量，host\_cursor\_variable\_name为PL/SQL主机环境（如OCI: ORACLE Call Interface，Pro\*c 程序等）中声明的游标变量。

OPEN…FOR 语句可以在关闭当前的游标变量之前重新打开游标变量，而不会导致CURSOR\_ALREAD\_OPEN异常错误。新打开游标变量时，前一个查询的内存处理区将被释放。

**2． 提取游标变量数据**

使用FETCH语句提取游标变量结果集合中的数据。格式为：

FETCH {cursor\_variable\_name | :host\_cursor\_variable\_name}  
INTO {variable [, variable]…| record\_variable};

其中：cursor\_variable\_name和host\_cursor\_variable\_name分别为游标变量和宿主游标变量名称；variable和record\_variable分别为普通变量和记录变量名称。

**3． 关闭游标变量**

CLOSE语句关闭游标变量，格式为：

CLOSE {cursor\_variable\_name | :host\_cursor\_variable\_name}

其中：cursor\_variable\_name和host\_cursor\_variable\_name分别为游标变量和宿主游标变量名称，如果应用程序试图关闭一个未打开的游标变量，则将导致INVALID\_CURSOR异常错误。

**例15：**强类型参照游标变量类型

DECLARE  
    TYPE emp\_job\_rec IS RECORD(  
        Employee\_id employees.employee\_id%TYPE,  
        Employee\_name employees.first\_name%TYPE,  
        Job\_title employees.job\_id%TYPE  
    );  
    TYPE emp\_job\_refcur\_type IS REF CURSOR RETURN emp\_job\_rec;  
    Emp\_refcur emp\_job\_refcur\_type ;  
    Emp\_job emp\_job\_rec;  
BEGIN  
    OPEN emp\_refcur FOR   
    SELECT employees.employee\_id, employees.first\_name||employees.last\_name, employees.job\_id   
  FROM employees   
  ORDER BY employees.department\_id;  
    
    FETCH emp\_refcur INTO emp\_job;  
    WHILE emp\_refcur%FOUND LOOP  
       DBMS\_OUTPUT.PUT\_LINE(emp\_job.employee\_id||': '||emp\_job.employee\_name||' is a '||emp\_job.job\_title);  
    FETCH emp\_refcur INTO emp\_job;  
    END LOOP;  
END;

**例16：**弱类型参照游标变量类型

PROMPT  
PROMPT 'What table would you like to see?'  
ACCEPT tab PROMPT '(D)epartment, or (E)mployees:'  
  
DECLARE  
    Type refcur\_t IS REF CURSOR;  
    Refcur refcur\_t;  
    TYPE sample\_rec\_type IS RECORD (  
        Id number,  
        Description VARCHAR2 (**30**)  
    );  
    sample sample\_rec\_type;  
    selection varchar2(**1**) := UPPER (SUBSTR ('&tab', **1**, **1**));  
BEGIN  
    IF selection='D' THEN  
        OPEN refcur FOR   
    SELECT departments.department\_id, departments.department\_name FROM departments;  
        DBMS\_OUTPUT.PUT\_LINE('Department data');  
    ELSIF selection='E' THEN  
        OPEN refcur FOR   
    SELECT employees.employee\_id, employees.first\_name||' is a '||employees.job\_id FROM employees;  
        DBMS\_OUTPUT.PUT\_LINE('Employee data');  
    ELSE  
        DBMS\_OUTPUT.PUT\_LINE('Please enter ''D'' or ''E''');  
        RETURN;  
    END IF;  
    DBMS\_OUTPUT.PUT\_LINE('----------------------');  
    FETCH refcur INTO sample;  
    WHILE refcur%FOUND LOOP  
        DBMS\_OUTPUT.PUT\_LINE(sample.id||': '||sample.description);  
        FETCH refcur INTO sample;  
    END LOOP;  
    CLOSE refcur;  
END;

**例17：**使用游标变量（没有RETURN子句）

DECLARE  
--定义一个游标数据类型  
   TYPE emp\_cursor\_type IS REF CURSOR;  
--声明一个游标变量  
   c1 EMP\_CURSOR\_TYPE;  
--声明两个记录变量  
   v\_emp\_record employees%ROWTYPE;  
   v\_reg\_record regions%ROWTYPE;  
  
BEGIN  
   OPEN c1 FOR SELECT \* FROM employees WHERE department\_id = **20**;  
   LOOP  
      FETCH c1 INTO v\_emp\_record;  
      EXIT WHEN c1%NOTFOUND;  
      DBMS\_OUTPUT.PUT\_LINE(v\_emp\_record.first\_name||'的雇佣日期是'  
                            ||v\_emp\_record.hire\_date);  
   END LOOP;  
--将同一个游标变量对应到另一个SELECT语句  
   OPEN c1 FOR SELECT \* FROM regions WHERE region\_id IN（**1**，**2**）;  
   LOOP  
      FETCH c1 INTO v\_reg\_record;  
      EXIT WHEN c1%NOTFOUND;  
      DBMS\_OUTPUT.PUT\_LINE(v\_reg\_record.region\_id||'表示'  
                            ||v\_reg\_record.region\_name);  
   END LOOP;  
   CLOSE c1;  
END;

**例18：**使用游标变量（有RETURN子句）

DECLARE  
--定义一个与employees表中的这几个列相同的记录数据类型  
   TYPE emp\_record\_type IS RECORD(  
        f\_name   employees.first\_name%TYPE,  
        h\_date   employees.hire\_date%TYPE,  
        j\_id     employees.job\_id%TYPE);  
--声明一个该记录数据类型的记录变量  
   v\_emp\_record EMP\_RECORD\_TYPE;  
--定义一个游标数据类型  
   TYPE emp\_cursor\_type IS REF CURSOR  
        RETURN EMP\_RECORD\_TYPE;  
--声明一个游标变量  
   c1 EMP\_CURSOR\_TYPE;  
BEGIN  
   OPEN c1 FOR SELECT first\_name, hire\_date, job\_id  
               FROM employees WHERE department\_id = **20**;  
   LOOP  
      FETCH c1 INTO v\_emp\_record;  
      EXIT WHEN c1%NOTFOUND;  
      DBMS\_OUTPUT.PUT\_LINE('雇员名称：'||v\_emp\_record.f\_name  
                ||'  雇佣日期：'||v\_emp\_record.h\_date  
                ||'  岗位：'||v\_emp\_record.j\_id);  
   END LOOP;  
   CLOSE c1;  
END;