**PL/SQL流程控制语句**

**本篇主要内容如下：**

[3.1  条件语句](http://www.cnblogs.com/huyong/archive/2011/05/13/2045407.html#_Toc32227)

[3.2  CASE 表达式](http://www.cnblogs.com/huyong/archive/2011/05/13/2045407.html#_Toc29808)

[3.3  循环](http://www.cnblogs.com/huyong/archive/2011/05/13/2045407.html#_Toc10255)

[3.4  标号和GOTO](http://www.cnblogs.com/huyong/archive/2011/05/13/2045407.html#_Toc28430)

[3.5  NULL 语句](http://www.cnblogs.com/huyong/archive/2011/05/13/2045407.html#_Toc15721)

**介绍PL/SQL的流程控制语句, 包括如下三类:**

控制语句: IF 语句

循环语句: LOOP语句, EXIT语句

顺序语句: GOTO语句, NULL语句

**3.1  条件语句**

IF <布尔表达式> THEN  
  PL/SQL 和 SQL语句  
END IF;  
-----------------------  
IF <布尔表达式> THEN  
  PL/SQL 和 SQL语句  
ELSE  
  其它语句  
END IF;  
-----------------------  
IF <布尔表达式> THEN  
  PL/SQL 和 SQL语句  
ELSIF < 其它布尔表达式> THEN  
  其它语句  
ELSIF < 其它布尔表达式> THEN  
  其它语句  
ELSE  
  其它语句  
END IF;

**提示: ELSIF 不能写成 ELSEIF**

**例1:**

DECLARE  
    v\_empno  employees.employee\_id%TYPE :=&empno;  
    V\_salary employees.salary%TYPE;  
    V\_comment VARCHAR2(**35**);  
BEGIN  
   SELECT salary INTO v\_salary FROM employees   
   WHERE employee\_id = v\_empno;  
   IF v\_salary < **1500** THEN  
       V\_comment:= '太少了,加点吧~!';  
   ELSIF v\_salary <**3000** THEN  
      V\_comment:= '多了点,少点吧~!';  
   ELSE  
      V\_comment:= '没有薪水~!';  
   END IF;  
   DBMS\_OUTPUT.PUT\_LINE(V\_comment);  
   exception  
     when no\_data\_found then  
        DBMS\_OUTPUT.PUT\_LINE('没有数据~!');  
     when others then  
        DBMS\_OUTPUT.PUT\_LINE(sqlcode || '---' || sqlerrm);          
END;

**例2:**

DECLARE  
   v\_first\_name  VARCHAR2(**20**);  
   v\_salary NUMBER(**7**,**2**);  
BEGIN  
   SELECT first\_name, salary INTO v\_first\_name, v\_salary FROM employees  
   WHERE employee\_id = &emp\_id;  
   DBMS\_OUTPUT.PUT\_LINE(v\_first\_name||'雇员的工资是'||v\_salary);  
   IF v\_salary < **10000** THEN  
      DBMS\_OUTPUT.PUT\_LINE('工资低于10000');  
   ELSE  
      IF **10000** <= v\_salary AND v\_salary < **20000** THEN  
         DBMS\_OUTPUT.PUT\_LINE('工资在10000到20000之间');  
      ELSE  
         DBMS\_OUTPUT.PUT\_LINE('工资高于20000');  
      END IF;  
   END IF;  
END;

**例3:**

**DECLARE  
   v\_first\_name  VARCHAR2(20);  
   v\_hire\_date DATE;  
   v\_bonus NUMBER(6,2);  
BEGIN  
   SELECT first\_name, hire\_date INTO v\_first\_name, v\_hire\_date FROM employees  
   WHERE employee\_id = &emp\_id;  
   IF v\_hire\_date > TO\_DATE('01-1月-90') THEN  
      v\_bonus := 800;  
   ELSIF v\_hire\_date > TO\_DATE('01-1月-88') THEN  
      v\_bonus := 1600;  
   ELSE  
      v\_bonus := 2400;  
   END IF;  
   DBMS\_OUTPUT.PUT\_LINE(v\_first\_name||'雇员的雇佣日期是'||v\_hire\_date  
                                    ||'、奖金是'||v\_bonus);  
END;**

**3.2  CASE 表达式**

---------格式一---------  
CASE 条件表达式  
  WHEN 条件表达式结果1 THEN   
     语句段1  
  WHEN 条件表达式结果2 THEN  
     语句段2  
  ......  
  WHEN 条件表达式结果n THEN  
     语句段n  
  [ELSE 条件表达式结果]  
END;  
---------格式二---------  
CASE   
  WHEN 条件表达式1 THEN  
     语句段1  
  WHEN 条件表达式2 THEN  
     语句段2  
  ......  
  WHEN 条件表达式n THEN   
     语句段n  
  [ELSE 语句段]  
END;

**例4:**

DECLARE  
  V\_grade char(**1**) := UPPER('&p\_grade');  
  V\_appraisal VARCHAR2(**20**);  
BEGIN  
  V\_appraisal :=  
  CASE v\_grade  
    WHEN 'A' THEN 'Excellent'  
    WHEN 'B' THEN 'Very Good'  
    WHEN 'C' THEN 'Good'  
    ELSE 'No such grade'  
  END;  
  DBMS\_OUTPUT.PUT\_LINE('Grade:'||v\_grade||'  Appraisal: '|| v\_appraisal);  
END;

**例5:**

DECLARE  
   v\_first\_name employees.first\_name%TYPE;  
   v\_job\_id employees.job\_id%TYPE;  
   v\_salary employees.salary%TYPE;  
   v\_sal\_raise NUMBER(**3**,**2**);  
BEGIN  
   SELECT first\_name,   job\_id,   salary INTO  
          v\_first\_name, v\_job\_id, v\_salary  
   FROM employees WHERE employee\_id = &emp\_id;  
   CASE  
      WHEN v\_job\_id = 'PU\_CLERK' THEN  
         IF v\_salary < **3000** THEN v\_sal\_raise := .**08**;  
         ELSE v\_sal\_raise := .**07**;  
         END IF;  
      WHEN v\_job\_id = 'SH\_CLERK' THEN  
         IF v\_salary < **4000** THEN v\_sal\_raise := .**06**;  
         ELSE v\_sal\_raise := .**05**;  
         END IF;  
      WHEN v\_job\_id = 'ST\_CLERK' THEN  
         IF v\_salary < **3500** THEN v\_sal\_raise := .**04**;  
         ELSE v\_sal\_raise := .**03**;  
         END IF;  
      ELSE  
         DBMS\_OUTPUT.PUT\_LINE('该岗位不涨工资: '||v\_job\_id);  
   END CASE;  
   DBMS\_OUTPUT.PUT\_LINE(v\_first\_name||'的岗位是'||v\_job\_id  
                                    ||'、的工资是'||v\_salary  
                                    ||'、工资涨幅是'||v\_sal\_raise);  
END;

**3.3  循环**

**1.  简单循环**

  LOOP  
      要执行的语句;  
      EXIT WHEN <条件语句> --条件满足，退出循环语句  
  END LOOP;

**例 6.**

DECLARE  
    int NUMBER(**2**) :=**0**;  
BEGIN  
   LOOP  
      int := int + **1**;  
      DBMS\_OUTPUT.PUT\_LINE('int 的当前值为:'||int);  
      EXIT WHEN int =**10**;  
   END LOOP;  
END;

**2.  WHILE 循环**

WHILE <布尔表达式> LOOP  
    要执行的语句;  
END LOOP;

**例7.**

DECLARE   
  x NUMBER :=**1**;  
BEGIN  
   WHILE x<=**10** LOOP  
      DBMS\_OUTPUT.PUT\_LINE('X的当前值为:'||x);  
       x:= x+**1**;  
   END LOOP;  
END;

**3.  数字式循环**

[<<循环标签>>]  
FOR 循环计数器 IN [ REVERSE ] 下限 .. 上限 LOOP  
  要执行的语句;  
END LOOP [循环标签];

每循环一次，循环变量自动加1；使用关键字REVERSE，循环变量自动减1。跟在IN REVERSE 后面的数字必须是从小到大的顺序，而且必须是整数，不能是变量或表达式。可以使用EXIT 退出循环。

**例8.**

BEGIN  
   FOR int  in **1**..**10** LOOP  
       DBMS\_OUTPUT.PUT\_LINE('int 的当前值为: '||int);  
   END LOOP;  
END;

**例 9.**

CREATE TABLE temp\_table(num\_col NUMBER);  
  
DECLARE  
    V\_counter NUMBER := **10**;  
BEGIN  
   INSERT INTO temp\_table(num\_col) VALUES (v\_counter );  
   FOR v\_counter IN **20** .. **25** LOOP  
      INSERT INTO temp\_table (num\_col ) VALUES ( v\_counter );  
   END LOOP;  
   INSERT INTO temp\_table(num\_col) VALUES (v\_counter );  
   FOR v\_counter IN REVERSE **20** .. **25** LOOP  
      INSERT INTO temp\_table (num\_col ) VALUES ( v\_counter );  
   END LOOP;  
END ;  
  
DROP TABLE temp\_table;

**例10:**

DECLARE  
   TYPE jobids\_varray IS VARRAY(**12**) OF VARCHAR2(**10**); --定义一个VARRAY数据类型  
   v\_jobids JOBIDS\_VARRAY; --声明一个具有JOBIDS\_VARRAY数据类型的变量  
   v\_howmany NUMBER; --声明一个变量来保存雇员的数量  
  
BEGIN  
   --用某些job\_id值初始化数组  
   v\_jobids := jobids\_varray('FI\_ACCOUNT', 'FI\_MGR', 'ST\_CLERK', 'ST\_MAN');  
  
   --用FOR...LOOP...END LOOP循环使用每个数组成员的值  
   FOR i IN v\_jobids.FIRST..v\_jobids.LAST LOOP  
  
   --针对数组中的每个岗位，决定该岗位的雇员的数量  
      SELECT count(\*) INTO v\_howmany FROM employees WHERE job\_id = v\_jobids(i);  
      DBMS\_OUTPUT.PUT\_LINE ( '岗位'||v\_jobids(i)||  
                       '总共有'|| TO\_CHAR(v\_howmany) || '个雇员');  
   END LOOP;  
END;

**例11**在While循环中嵌套loop循环

**/\*求100至110之间的素数\*/  
DECLARE  
   v\_m NUMBER := 101;  
   v\_i NUMBER;  
   v\_n NUMBER := 0;  
BEGIN  
   WHILE v\_m < 110 LOOP  
      v\_i := 2;  
      LOOP  
         IF mod(v\_m, v\_i) = 0 THEN  
            v\_i := 0;  
            EXIT;  
         END IF;  
      
         v\_i := v\_i + 1;  
         EXIT WHEN v\_i > v\_m - 1;   
      END LOOP;  
        
      IF v\_i > 0 THEN  
         v\_n := v\_n + 1;  
         DBMS\_OUTPUT.PUT\_LINE('第'|| v\_n || '个素数是' || v\_m);  
      END IF;  
  
      v\_m := v\_m + 2;  
   END LOOP;  
END;**

**3.4  标号和GOTO**

PL/SQL中GOTO语句是无条件跳转到指定的标号去的意思。语法如下：

GOTO label;  
......  
<<label>>　/\*标号是用<<　>>括起来的标识符 \*/

**注意**，在以下地方使用是不合法的，编译时会出错误。

跳转到非执行语句前面。

跳转到子块中。

跳转到循环语句中。

跳转到条件语句中。

从异常处理部分跳转到执行。

从条件语句的一部分跳转到另一部分。

**例12:**

DECLARE  
   V\_counter NUMBER := **1**;  
BEGIN  
   LOOP   
     DBMS\_OUTPUT.PUT\_LINE('V\_counter的当前值为:'||V\_counter);  
     V\_counter := v\_counter + **1**;  
   IF v\_counter > **10** THEN  
       GOTO labelOffLOOP;  
   END IF;  
   END LOOP;  
   <<labelOffLOOP>>  
     DBMS\_OUTPUT.PUT\_LINE('V\_counter的当前值为:'||V\_counter);  
END;

**例13:**

**DECLARE  
   v\_i NUMBER := 0;  
   v\_s NUMBER := 0;  
BEGIN  
   <<label\_1>>  
   v\_i := v\_i + 1;  
   IF v\_i <= 1000 THEN  
      v\_s := v\_s + v\_i;  
      GOTO label\_1;  
   END IF;  
   DBMS\_OUTPUT.PUT\_LINE(v\_s);  
END;**

**3.5  NULL 语句**

在PL/SQL 程序中，NULL语句是一个可执行语句，可以用 null 语句来说明“不用做任何事情”的意思，相当于一个占位符或不执行任何操作的空语句，可以使某些语句变得有意义，提高程序的可读性，保证其他语句结构的完整性和正确性。如：

**例14:**

DECLARE  
    ...  
BEGIN  
    ...  
    IF v\_num IS NULL THEN  
    GOTO labelPrint;  
    END IF;  
  …  
  <<labelPrint>>  
  NULL; --不需要处理任何数据。  
END;

**例15:**

DECLARE  
   v\_emp\_id employees.employee\_id%TYPE;  
   v\_first\_name employees.first\_name%TYPE;  
   v\_salary employees.salary%TYPE;  
   v\_sal\_raise NUMBER(**3**,**2**);  
BEGIN  
   v\_emp\_id := &emp\_id;  
   SELECT first\_name, salary INTO v\_first\_name, v\_salary  
   FROM employees WHERE employee\_id = v\_emp\_id;  
   IF v\_salary <= **3000** THEN  
      v\_sal\_raise := .**10**;  
      DBMS\_OUTPUT.PUT\_LINE(v\_first\_name||'的工资是'||v\_salary  
                                       ||'、工资涨幅是'||v\_sal\_raise);  
   ELSE  
      NULL;