**异常错误处理**

ORACLE 提供异常情况(EXCEPTION)和异常处理(EXCEPTION HANDLER)来实现错误处理。

**5.1 异常处理概念**

异常情况处理(EXCEPTION)是用来处理正常执行过程中未预料的事件,程序块的异常处理预定义的错误和自定义错误,由于PL/SQL程序块一旦产生异常而没有指出如何处理时,程序就会自动终止整个程序运行.

有三种类型的异常错误：

    1． 预定义 ( Predefined )错误

  ORACLE预定义的异常情况大约有24个。对这种异常情况的处理，无需在程序中定义，由ORACLE自动将其引发。

    2． 非预定义 ( Predefined )错误

   即其他标准的ORACLE错误。对这种异常情况的处理，需要用户在程序中定义，然后由ORACLE自动将其引发。

    3． 用户定义(User\_define) 错误

 程序执行过程中，出现编程人员认为的非正常情况。对这种异常情况的处理，需要用户在程序中定义，然后显式地在程序中将其引发。

异常处理部分一般放在 PL/SQL 程序体的后半部,结构为:

EXCEPTION  
   WHEN first\_exception THEN  <code to handle first exception >  
   WHEN second\_exception THEN  <code to handle second exception >  
   WHEN OTHERS THEN  <code to handle others exception >  
END;

异常处理可以按任意次序排列,但 OTHERS 必须放在最后.

**5.1.1 预定义的异常处理**

   预定义说明的部分 ORACLE 异常错误

|  |  |  |
| --- | --- | --- |
| 错误号 | 异常错误信息名称 | 说明 |
| ORA-0001 | Dup\_val\_on\_index | 违反了唯一性限制 |
| ORA-0051 | Timeout-on-resource | 在等待资源时发生超时 |
| ORA-0061 | Transaction-backed-out | 由于发生死锁事务被撤消 |
| ORA-1001 | Invalid-CURSOR | 试图使用一个无效的游标 |
| ORA-1012 | Not-logged-on | 没有连接到ORACLE |
| ORA-1017 | Login-denied | 无效的用户名/口令 |
| ORA-1403 | No\_data\_found | SELECT INTO没有找到数据 |
| ORA-1422 | Too\_many\_rows | SELECT INTO 返回多行 |
| ORA-1476 | Zero-divide | 试图被零除 |
| ORA-1722 | Invalid-NUMBER | 转换一个数字失败 |
| ORA-6500 | Storage-error | 内存不够引发的内部错误 |
| ORA-6501 | Program-error | 内部错误 |
| ORA-6502 | Value-error | 转换或截断错误 |
| ORA-6504 | Rowtype-mismatch | 宿主游标变量与 PL/SQL变量有不兼容行类型 |
| ORA-6511 | CURSOR-already-OPEN | 试图打开一个已处于打开状态的游标 |
| ORA-6530 | Access-INTO-null | 试图为null 对象的属性赋值 |
| ORA-6531 | Collection-is-null | 试图将Exists 以外的集合( collection)方法应用于一个null pl/sql 表上或varray上 |
| ORA-6532 | Subscript-outside-limit | 对嵌套或varray索引得引用超出声明范围以外 |
| ORA-6533 | Subscript-beyond-count | 对嵌套或varray 索引得引用大于集合中元素的个数. |

对这种异常情况的处理，只需在PL/SQL块的异常处理部分，直接引用相应的异常情况名，并对其完成相应的异常错误处理即可。

**例1：**更新指定员工工资，如工资小于1500，则加100；

**DECLARE  
   v\_empno employees.employee\_id%TYPE := &empno;  
   v\_sal   employees.salary%TYPE;  
BEGIN  
   SELECT salary INTO v\_sal FROM employees WHERE employee\_id = v\_empno;  
   IF v\_sal<=1500 THEN   
        UPDATE employees SET salary = salary + 100 WHERE employee\_id=v\_empno;   
        DBMS\_OUTPUT.PUT\_LINE('编码为'||v\_empno||'员工工资已更新!');       
   ELSE  
        DBMS\_OUTPUT.PUT\_LINE('编码为'||v\_empno||'员工工资已经超过规定值!');  
   END IF;  
EXCEPTION  
   WHEN NO\_DATA\_FOUND THEN    
      DBMS\_OUTPUT.PUT\_LINE('数据库中没有编码为'||v\_empno||'的员工');  
   WHEN TOO\_MANY\_ROWS THEN  
      DBMS\_OUTPUT.PUT\_LINE('程序运行错误!请使用游标');  
   WHEN OTHERS THEN  
      DBMS\_OUTPUT.PUT\_LINE(SQLCODE||'---'||SQLERRM);  
END;**

**5.1.2 非预定义的异常处理**

对于这类异常情况的处理，首先必须对非定义的ORACLE错误进行定义。步骤如下：

1. 在PL/SQL 块的定义部分定义异常情况：

<异常情况>  EXCEPTION;

 2. 将其定义好的异常情况，与标准的ORACLE错误联系起来，使用EXCEPTION\_INIT语句：

PRAGMA EXCEPTION\_INIT(<异常情况>, <错误代码>);

 3. 在PL/SQL 块的异常情况处理部分对异常情况做出相应的处理。

**例2：**删除指定部门的记录信息，以确保该部门没有员工。

INSERT INTO departments VALUES(**50**, 'FINANCE', 'CHICAGO');  
  
DECLARE  
   v\_deptno departments.department\_id%TYPE := &deptno;  
   deptno\_remaining EXCEPTION;  
   PRAGMA EXCEPTION\_INIT(deptno\_remaining, -**2292**);  
   /\* -2292 是违反一致性约束的错误代码 \*/  
BEGIN  
   DELETE FROM departments WHERE department\_id = v\_deptno;  
EXCEPTION  
   WHEN deptno\_remaining THEN   
      DBMS\_OUTPUT.PUT\_LINE('违反数据完整性约束!');  
   WHEN OTHERS THEN  
      DBMS\_OUTPUT.PUT\_LINE(SQLCODE||'---'||SQLERRM);  
END;

**5.1.3 用户自定义的异常处理**

当与一个异常错误相关的错误出现时，就会隐含触发该异常错误。用户定义的异常错误是通过显式使用 RAISE 语句来触发。当引发一个异常错误时，控制就转向到 EXCEPTION块异常错误部分，执行错误处理代码。

对于这类异常情况的处理，步骤如下：

1． 在PL/SQL 块的定义部分定义异常情况：

<异常情况>  EXCEPTION;

2． RAISE <异常情况>；

3． 在PL/SQL 块的异常情况处理部分对异常情况做出相应的处理。

**例3：**更新指定员工工资，增加100；

DECLARE  
   v\_empno employees.employee\_id%TYPE :=&empno;  
   no\_result  EXCEPTION;  
BEGIN  
   UPDATE employees SET salary = salary+**100** WHERE employee\_id = v\_empno;  
   IF SQL%NOTFOUND THEN  
      RAISE no\_result;  
   END IF;  
EXCEPTION  
   WHEN no\_result THEN   
      DBMS\_OUTPUT.PUT\_LINE('你的数据更新语句失败了!');  
   WHEN OTHERS THEN  
      DBMS\_OUTPUT.PUT\_LINE(SQLCODE||'---'||SQLERRM);  
END;

**5.1.4  用户定义的异常处理**

调用DBMS\_STANDARD(ORACLE提供的包)包所定义的RAISE\_APPLICATION\_ERROR过程，可以重新定义异常错误消息，它为应用程序提供了一种与ORACLE交互的方法。

RAISE\_APPLICATION\_ERROR 的语法如下：

    RAISE\_APPLICATION\_ERROR(error\_number,error\_message,[keep\_errors] );

    这里的error\_number 是从 –20,000 到 –20,999 之间的参数，

    error\_message 是相应的提示信息(< 2048 字节)，

  keep\_errors 为可选，如果keep\_errors =TRUE ,则新错误将被添加到已经引发的错误列表中。如果keep\_errors=FALSE(缺省),则新错误将替换当前的错误列表。

**例4：**创建一个函数get\_salary, 该函数检索指定部门的工资总和，其中定义了-20991和-20992号错误，分别处理参数为空和非法部门代码两种错误：

CREATE TABLE errlog(  
  Errcode NUMBER,  
  Errtext CHAR(**40**));  
  
CREATE OR REPLACE FUNCTION get\_salary(p\_deptno NUMBER)  
RETURN NUMBER   
AS  
  v\_sal NUMBER;  
BEGIN  
  IF p\_deptno IS NULL THEN  
    RAISE\_APPLICATION\_ERROR(-**20991**, ’部门代码为空’);  
  ELSIF p\_deptno<**0** THEN  
    RAISE\_APPLICATION\_ERROR(-**20992**, ’无效的部门代码’);  
  ELSE  
    SELECT SUM(employees.salary) INTO v\_sal FROM employees   
    WHERE employees.department\_id=p\_deptno;  
    RETURN v\_sal;  
  END IF;  
END;  
  
DECLARE   
  V\_salary NUMBER(**7**,**2**);  
  V\_sqlcode NUMBER;  
  V\_sqlerr VARCHAR2(**512**);  
  Null\_deptno EXCEPTION;  
  Invalid\_deptno EXCEPTION;  
  PRAGMA EXCEPTION\_INIT(null\_deptno,-**20991**);  
  PRAGMA EXCEPTION\_INIT(invalid\_deptno, -**20992**);  
BEGIN  
  V\_salary :=get\_salary(**10**);  
  DBMS\_OUTPUT.PUT\_LINE('10号部门工资：' || TO\_CHAR(V\_salary));  
  
  BEGIN  
    V\_salary :=get\_salary(-**10**);  
  EXCEPTION  
    WHEN invalid\_deptno THEN  
      V\_sqlcode :=SQLCODE;  
      V\_sqlerr  :=SQLERRM;  
      INSERT INTO errlog(errcode, errtext)   
      VALUES(v\_sqlcode, v\_sqlerr);  
      COMMIT;  
  END inner1;  
  
  V\_salary :=get\_salary(**20**);  
  DBMS\_OUTPUT.PUT\_LINE('部门号为20的工资为：'||TO\_CHAR(V\_salary));  
  
  BEGIN  
    V\_salary :=get\_salary(NULL);  
  END inner2;  
  
  V\_salary := get\_salary(**30**);  
  DBMS\_OUTPUT.PUT\_LINE('部门号为30的工资为：'||TO\_CHAR(V\_salary));  
  
  EXCEPTION  
    WHEN null\_deptno THEN  
      V\_sqlcode :=SQLCODE;  
      V\_sqlerr  :=SQLERRM;  
      INSERT INTO errlog(errcode, errtext) VALUES(v\_sqlcode, v\_sqlerr);  
      COMMIT;  
    WHEN OTHERS THEN  
         DBMS\_OUTPUT.PUT\_LINE(SQLCODE||'---'||SQLERRM);  
END outer;

**例5：**定义触发器，使用RAISE\_APPLICATION\_ERROR阻止没有员工姓名的新员式记录插入：

CREATE OR REPLACE TRIGGER tr\_insert\_emp  
BEFORE INSERT ON employees  
FOR EACH ROW  
BEGIN  
  IF :new.first\_name IS NULL OR :new.last\_name is null THEN  
    RAISE\_APPLICATION\_ERROR(-**20000**,'Employee must have a name.');  
  END IF;  
END;

**5.2 异常错误传播**

    由于异常错误可以在声明部分和执行部分以及异常错误部分出现，因而在不同部分引发的异常错误也不一样。

**5.2.1 在执行部分引发异常错误**

    当一个异常错误在执行部分引发时，有下列情况：

如果当前块对该异常错误设置了处理，则执行它并成功完成该块的执行，然后控制转给包含块。

如果没有对当前块异常错误设置定义处理器，则通过在包含块中引发它来传播异常错误。然后对该包含块执行步骤1)。

**5.2.2 在声明部分引发异常错误**

    如果在声明部分引起异常情况，即在声明部分出现错误，那么该错误就能影响到其它的块。比如在有如下的PL/SQL程序：

DECLARE  
    name varchar2(**12**):='EricHu';  
    其它语句  
BEGIN  
    其它语句  
EXCEPTION  
    WHEN OTHERS THEN   
    其它语句  
END;

     例子中，由于Abc number(3)=’abc’; 出错，尽管在EXCEPTION中说明了WHEN OTHERS THEN语句，但WHEN OTHERS THEN也不会被执行。 但是如果在该错误语句块的外部有一个异常错误，则该错误能被抓住，如：

BEGIN  
    DECLARE  
    name varchar2(**12**):='EricHu';  
    其它语句  
   BEGIN  
    其它语句  
   EXCEPTION  
    WHEN OTHERS THEN   
    其它语句  
    END;  
EXCEPTION  
WHEN OTHERS THEN   
    其它语句  
END;

**5.3 异常错误处理编程**

    在一般的应用处理中，建议程序人员要用异常处理，因为如果程序中不声明任何异常处理，则在程序运行出错时，程序就被终止，并且也不提示任何信息。下面是使用系统提供的异常来编程的例子。

**5.4  在 PL/SQL 中使用 SQLCODE, SQLERRM异常处理函数**

    由于ORACLE 的错信息最大长度是512字节，为了得到完整的错误提示信息，我们可用 SQLERRM和 SUBSTR 函数一起得到错误提示信息，方便进行错误，特别是如果WHEN OTHERS异常处理器时更为方便。

SQLCODE  返回遇到的Oracle错误号,

SQLERRM  返回遇到的Oracle错误信息.

**如:**  SQLCODE=-100    SQLERRM=’no\_data\_found ‘

 SQLCODE=0       SQLERRM=’normal, successfual completion’

**例6.**将ORACLE错误代码及其信息存入错误代码表

CREATE TABLE errors (errnum NUMBER(**4**), errmsg VARCHAR2(**100**));  
  
DECLARE  
   err\_msg  VARCHAR2(**100**);  
BEGIN  
   /\*  得到所有 ORACLE 错误信息  \*/  
   FOR err\_num IN -**100** .. **0** LOOP  
      err\_msg := SQLERRM(err\_num);  
      INSERT INTO errors VALUES(err\_num, err\_msg);  
   END LOOP;  
END;  
DROP TABLE errors;

**例7.**查询ORACLE错误代码；

BEGIN  
   INSERT INTO employees(employee\_id, first\_name,last\_name,hire\_date,department\_id)  
   VALUES(**2222**, 'Eric','Hu', SYSDATE, **20**);  
   DBMS\_OUTPUT.PUT\_LINE('插入数据记录成功!');  
     
   INSERT INTO employees(employee\_id, first\_name,last\_name,hire\_date,department\_id)  
   VALUES(**2222**, '胡','勇', SYSDATE, **20**);  
   DBMS\_OUTPUT.PUT\_LINE('插入数据记录成功!');  
EXCEPTION  
   WHEN OTHERS THEN  
      DBMS\_OUTPUT.PUT\_LINE(SQLCODE||'---'||SQLERRM);  
END;

**例8.**利用ORACLE错误代码，编写异常错误处理代码；

DECLARE  
   empno\_remaining EXCEPTION;  
   PRAGMA EXCEPTION\_INIT(empno\_remaining, -**1**);  
   /\* -1 是违反唯一约束条件的错误代码 \*/  
BEGIN  
   INSERT INTO employees(employee\_id, first\_name,last\_name,hire\_date,department\_id)  
   VALUES(**3333**, 'Eric','Hu', SYSDATE, **20**);  
   DBMS\_OUTPUT.PUT\_LINE('插入数据记录成功!');  
     
   INSERT INTO employees(employee\_id, first\_name,last\_name,hire\_date,department\_id)  
   VALUES(**3333**, '胡','勇',SYSDATE, **20**);  
   DBMS\_OUTPUT.PUT\_LINE('插入数据记录成功!');  
EXCEPTION  
   WHEN empno\_remaining THEN   
      DBMS\_OUTPUT.PUT\_LINE('违反数据完整性约束!');  
   WHEN OTHERS THEN  
      DBMS\_OUTPUT.PUT\_LINE(SQLCODE||'---'||SQLERRM);  
END;