**把过程与函数说透**

**6.1 引言**

过程与函数（另外还有包与触发器）是命名的PL/SQL块（也是用户的方案对象），被编译后存储在数据库中，以备执行。因此，其它PL/SQL块可以按名称来使用他们。所以，可以将商业逻辑、企业规则写成函数或过程保存到数据库中，以便共享。

过程和函数统称为PL/SQL子程序，他们是被命名的PL/SQL块，均存储在数据库中，并通过输入、输出参数或输入/输出参数与其调用者交换信息。过程和函数的唯一区别是函数总向调用者返回数据，而过程则不返回数据。在本节中，主要介绍：

1．   创建存储过程和函数。

2．   正确使用系统级的异常处理和用户定义的异常处理。

3．   建立和管理存储过程和函数。

**6.2 创建函数**

**1. 创建函数**

语法如下：

CREATE [OR REPLACE] FUNCTION function\_name  
 (arg1 [ { IN | OUT | IN OUT }] type1 [DEFAULT value1],  
 [arg2 [ { IN | OUT | IN OUT }] type2 [DEFAULT value1]],  
 ......  
 [argn [ { IN | OUT | IN OUT }] typen [DEFAULT valuen]])  
 [ AUTHID DEFINER | CURRENT\_USER ]  
RETURN return\_type   
 IS | AS  
    <类型.变量的声明部分>   
BEGIN  
    执行部分  
    RETURN expression  
EXCEPTION  
    异常处理部分  
END function\_name;

         IN,OUT,IN OUT是形参的模式。若省略，则为IN模式。IN模式的形参只能将实参传递给形参，进入函数内部，但只能读不能写，函数返回时实参的值不变。OUT模式的形参会忽略调用时的实参值（或说该形参的初始值总是NULL），但在函数内部可以被读或写，函数返回时形参的值会赋予给实参。IN OUT具有前两种模式的特性，即调用时，实参的值总是传递给形参，结束时，形参的值传递给实参。调用时，对于IN模式的实参可以是常量或变量，但对于OUT和IN OUT模式的实参必须是变量。

         一般，只有在确认function\_name函数是新函数或是要更新的函数时，才使用OR REPALCE关键字，否则容易删除有用的函数。

**例1.**           获取某部门的工资总和：

--获取某部门的工资总和  
CREATE OR REPLACE  
FUNCTION get\_salary(  
  Dept\_no NUMBER,  
  Emp\_count OUT NUMBER)  
  RETURN NUMBER   
IS  
  V\_sum NUMBER;  
BEGIN  
  SELECT SUM(SALARY), count(\*) INTO V\_sum, emp\_count  
    FROM EMPLOYEES WHERE DEPARTMENT\_ID=dept\_no;  
  RETURN v\_sum;  
EXCEPTION  
   WHEN NO\_DATA\_FOUND THEN   
      DBMS\_OUTPUT.PUT\_LINE('你需要的数据不存在!');  
   WHEN OTHERS THEN   
      DBMS\_OUTPUT.PUT\_LINE(SQLCODE||'---'||SQLERRM);  
END get\_salary;

**2. 函数的调用**

函数声明时所定义的参数称为形式参数，应用程序调用时为函数传递的参数称为实际参数。应用程序在调用函数时，可以使用以下三种方法向函数传递参数：

**第一种参数传递格式：位置表示法。**

即在调用时按形参的排列顺序，依次写出实参的名称，而将形参与实参关联起来进行传递。用这种方法进行调用，形参与实参的名称是相互独立，没有关系，强调次序才是重要的。

**格式为：**

       argument\_value1[,argument\_value2 …]

**例2：**计算某部门的工资总和：

**DECLARE  
  V\_num NUMBER;  
  V\_sum NUMBER;  
BEGIN  
  V\_sum :=get\_salary(10, v\_num);  
  DBMS\_OUTPUT.PUT\_LINE('部门号为:10的工资总和：'||v\_sum||'，人数为：'||v\_num);  
END;**

**第二种参数传递格式：名称表示法。**

即在调用时按形参的名称与实参的名称，写出实参对应的形参，而将形参与实参关联起来进行传递。这种方法，形参与实参的名称是相互独立的，没有关系，名称的对应关系才是最重要的，次序并不重要。

**格式为：**

       argument => parameter [,…]

其中：argument 为形式参数，它必须与函数定义时所声明的形式参数名称相同parameter 为实际参数。

在这种格式中，形势参数与实际参数成对出现，相互间关系唯一确定，所以参数的顺序可以任意排列。

**例3：**计算某部门的工资总和：

DECLARE  
  V\_num NUMBER;  
    V\_sum NUMBER;  
BEGIN  
    V\_sum :=get\_salary(emp\_count => v\_num, dept\_no => **10**);  
    DBMS\_OUTPUT.PUT\_LINE('部门号为:10的工资总和：'||v\_sum||'，人数为：'||v\_num);  
END;

**第三种参数传递格式：组合传递。**

即在调用一个函数时，同时使用位置表示法和名称表示法为函数传递参数。采用这种参数传递方法时，使用位置表示法所传递的参数必须放在名称表示法所传递的参数前面。也就是说，无论函数具有多少个参数，只要其中有一个参数使用名称表示法，其后所有的参数都必须使用名称表示法。

**例4：**

CREATE OR REPLACE FUNCTION demo\_fun(  
  Name VARCHAR2,--注意VARCHAR2不能给精度，如：VARCHAR2(10)，其它类似  
  Age INTEGER,  
  Sex VARCHAR2)  
  RETURN VARCHAR2   
AS  
  V\_var VARCHAR2(**32**);  
BEGIN  
  V\_var := name||'：'||TO\_CHAR(age)||'岁.'||sex;  
  RETURN v\_var;  
END;  
  
DECLARE   
  Var VARCHAR(**32**);  
BEGIN  
  Var := demo\_fun('user1', **30**, sex => '男');  
  DBMS\_OUTPUT.PUT\_LINE(var);  
  
  Var := demo\_fun('user2', age => **40**, sex => '男');  
  DBMS\_OUTPUT.PUT\_LINE(var);  
  
  Var := demo\_fun('user3', sex => '女', age => **20**);  
  DBMS\_OUTPUT.PUT\_LINE(var);  
END;

无论采用哪一种参数传递方法，实际参数和形式参数之间的数据传递只有两种方法：传址法和传值法。所谓传址法是指在调用函数时，将实际参数的地址指针传递给形式参数，使形式参数和实际参数指向内存中的同一区域，从而实现参数数据的传递。这种方法又称作参照法，即形式参数参照实际参数数据。输入参数均采用传址法传递数据。

       传值法是指将实际参数的数据拷贝到形式参数，而不是传递实际参数的地址。默认时，输出参数和输入/输出参数均采用传值法。在函数调用时，ORACLE将实际参数数据拷贝到输入/输出参数，而当函数正常运行退出时，又将输出形式参数和输入/输出形式参数数据拷贝到实际参数变量中。

**3. 参数默认值**

在CREATE OR REPLACE FUNCTION 语句中声明函数参数时可以使用DEFAULT关键字为输入参数指定默认值。

**例5：**

CREATE OR REPLACE FUNCTION demo\_fun(  
  Name VARCHAR2,  
  Age INTEGER,  
  Sex VARCHAR2 DEFAULT '男')  
  RETURN VARCHAR2   
AS  
  V\_var VARCHAR2(**32**);  
BEGIN  
  V\_var := name||'：'||TO\_CHAR(age)||'岁.'||sex;  
  RETURN v\_var;  
END;

具有默认值的函数创建后，在函数调用时，如果没有为具有默认值的参数提供实际参数值，函数将使用该参数的默认值。但当调用者为默认参数提供实际参数时，函数将使用实际参数值。在创建函数时，只能为输入参数设置默认值，而不能为输入/输出参数设置默认值。

DECLARE

 var VARCHAR(32);

BEGIN

 Var := demo\_fun('user1', 30);

 DBMS\_OUTPUT.PUT\_LINE(var);

 Var := demo\_fun('user2', age => 40);

 DBMS\_OUTPUT.PUT\_LINE(var);

 Var := demo\_fun('user3', sex => '女', age => 20);

 DBMS\_OUTPUT.PUT\_LINE(var);

END;

**6.3 存储过程**

**6.3.1 创建过程**

**建立存储过程**

在 ORACLE SERVER上建立存储过程,可以被多个应用程序调用,可以向存储过程传递参数,也可以向存储过程传回参数.

**创建过程语法:**

CREATE [OR REPLACE] PROCEDURE procedure\_name  
([arg1 [ IN | OUT | IN OUT ]] type1 [DEFAULT value1],  
 [arg2 [ IN | OUT | IN OUT ]] type2 [DEFAULT value1]],  
 ......  
 [argn [ IN | OUT | IN OUT ]] typen [DEFAULT valuen])  
    [ AUTHID DEFINER | CURRENT\_USER ]  
{ IS | AS }  
  <声明部分>   
BEGIN  
  <执行部分>  
EXCEPTION  
  <可选的异常错误处理程序>  
END procedure\_name;

**说明：**相关参数说明参见函数的语法说明。

**例6．**用户连接登记记录；

**CREATE TABLE logtable (userid VARCHAR2(10), logdate date);  
  
CREATE OR REPLACE PROCEDURE logexecution   
IS  
BEGIN  
INSERT INTO logtable (userid, logdate) VALUES (USER, SYSDATE);  
END;**

**例7．**删除指定员工记录；

CREATE OR REPLACE  
PROCEDURE DelEmp  
(v\_empno IN employees.employee\_id%TYPE)   
AS  
No\_result EXCEPTION;  
BEGIN  
   DELETE FROM employees WHERE employee\_id = v\_empno;  
   IF SQL%NOTFOUND THEN  
      RAISE no\_result;  
   END IF;  
   DBMS\_OUTPUT.PUT\_LINE('编码为'||v\_empno||'的员工已被删除!');  
EXCEPTION  
   WHEN no\_result THEN   
      DBMS\_OUTPUT.PUT\_LINE('温馨提示:你需要的数据不存在!');  
   WHEN OTHERS THEN  
      DBMS\_OUTPUT.PUT\_LINE(SQLCODE||'---'||SQLERRM);  
END DelEmp;

**例8．**插入员工记录:

**CREATE OR REPLACE  
PROCEDURE InsertEmp(  
   v\_empno     in employees.employee\_id%TYPE,  
   v\_firstname in employees.first\_name%TYPE,  
   v\_lastname  in employees.last\_name%TYPE,  
   v\_deptno    in employees.department\_id%TYPE  
   )   
AS  
   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(v\_empno, v\_firstname,v\_lastname, sysdate, v\_deptno);  
   DBMS\_OUTPUT.PUT\_LINE('温馨提示:插入数据记录成功!');  
EXCEPTION  
   WHEN empno\_remaining THEN   
      DBMS\_OUTPUT.PUT\_LINE('温馨提示:违反数据完整性约束!');  
   WHEN OTHERS THEN  
      DBMS\_OUTPUT.PUT\_LINE(SQLCODE||'---'||SQLERRM);  
END InsertEmp;**

**例9．**使用存储过程向departments表中插入数据。

CREATE OR REPLACE  
PROCEDURE insert\_dept  
  (v\_dept\_id IN departments.department\_id%TYPE,  
   v\_dept\_name IN departments.department\_name%TYPE,  
   v\_mgr\_id IN departments.manager\_id%TYPE,  
   v\_loc\_id IN departments.location\_id%TYPE)  
IS  
   ept\_null\_error EXCEPTION;  
   PRAGMA EXCEPTION\_INIT(ept\_null\_error, -**1400**);  
   ept\_no\_loc\_id EXCEPTION;  
   PRAGMA EXCEPTION\_INIT(ept\_no\_loc\_id, -**2291**);  
BEGIN  
   INSERT INTO departments  
   (department\_id, department\_name, manager\_id, location\_id)  
   VALUES  
   (v\_dept\_id, v\_dept\_name, v\_mgr\_id, v\_loc\_id);  
   DBMS\_OUTPUT.PUT\_LINE('插入部门'||v\_dept\_id||'成功');  
EXCEPTION  
   WHEN DUP\_VAL\_ON\_INDEX THEN  
      RAISE\_APPLICATION\_ERROR(-**20000**, '部门编码不能重复');  
   WHEN ept\_null\_error THEN  
      RAISE\_APPLICATION\_ERROR(-**20001**, '部门编码、部门名称不能为空');  
   WHEN ept\_no\_loc\_id THEN  
      RAISE\_APPLICATION\_ERROR(-**20002**, '没有该地点');  
END insert\_dept;  
  
/\*调用实例一:  
DECLARE  
   ept\_20000 EXCEPTION;  
   PRAGMA EXCEPTION\_INIT(ept\_20000, -20000);  
   ept\_20001 EXCEPTION;  
   PRAGMA EXCEPTION\_INIT(ept\_20001, -20001);  
   ept\_20002 EXCEPTION;  
   PRAGMA EXCEPTION\_INIT(ept\_20002, -20002);  
BEGIN  
   insert\_dept(300, '部门300', 100, 2400);  
   insert\_dept(310, NULL, 100, 2400);  
   insert\_dept(310, '部门310', 100, 900);  
EXCEPTION  
   WHEN ept\_20000 THEN  
      DBMS\_OUTPUT.PUT\_LINE('ept\_20000部门编码不能重复');  
   WHEN ept\_20001 THEN  
      DBMS\_OUTPUT.PUT\_LINE('ept\_20001部门编码、部门名称不能为空');  
   WHEN ept\_20002 THEN  
      DBMS\_OUTPUT.PUT\_LINE('ept\_20002没有该地点');  
   WHEN OTHERS THEN  
      DBMS\_OUTPUT.PUT\_LINE('others出现了其他异常错误');  
END;  
  
调用实例二：  
DECLARE  
   ept\_20000 EXCEPTION;  
   PRAGMA EXCEPTION\_INIT(ept\_20000, -20000);  
   ept\_20001 EXCEPTION;  
   PRAGMA EXCEPTION\_INIT(ept\_20001, -20001);  
   ept\_20002 EXCEPTION;  
   PRAGMA EXCEPTION\_INIT(ept\_20002, -20002);  
BEGIN  
   insert\_dept(v\_dept\_name => '部门310', v\_dept\_id => 310,   
               v\_mgr\_id => 100, v\_loc\_id => 2400);  
   insert\_dept(320, '部门320', v\_mgr\_id => 100, v\_loc\_id => 900);  
EXCEPTION  
   WHEN ept\_20000 THEN  
      DBMS\_OUTPUT.PUT\_LINE('ept\_20000部门编码不能重复');  
   WHEN ept\_20001 THEN  
      DBMS\_OUTPUT.PUT\_LINE('ept\_20001部门编码、部门名称不能为空');  
   WHEN ept\_20002 THEN  
      DBMS\_OUTPUT.PUT\_LINE('ept\_20002没有该地点');  
   WHEN OTHERS THEN  
      DBMS\_OUTPUT.PUT\_LINE('others出现了其他异常错误');  
END;  
\*/

**6.3.2 调用存储过程**

    存储过程建立完成后，只要通过授权，用户就可以在SQLPLUS 、ORACLE开发工具或第三方开发工具中来调用运行。对于参数的传递也有三种：按位置传递、按名称传递和组合传递，传递方法与函数的一样。ORACLE 使用EXECUTE 语句来实现对存储过程的调用：

EXEC[UTE] procedure\_name( parameter1, parameter2…);

**例10：**

 EXECUTE logexecution;

**例11：**查询指定员工记录；

CREATE OR REPLACE  
PROCEDURE QueryEmp  
(v\_empno IN  employees.employee\_id%TYPE,  
 v\_ename OUT employees.first\_name%TYPE,  
 v\_sal   OUT employees.salary%TYPE)   
AS  
BEGIN  
       SELECT last\_name || last\_name, salary INTO v\_ename, v\_sal   
    FROM employees   
    WHERE employee\_id = v\_empno;   
       DBMS\_OUTPUT.PUT\_LINE('温馨提示:编码为'||v\_empno||'的员工已经查到!');  
EXCEPTION  
       WHEN NO\_DATA\_FOUND THEN   
      DBMS\_OUTPUT.PUT\_LINE('温馨提示:你需要的数据不存在!');  
      WHEN OTHERS THEN   
      DBMS\_OUTPUT.PUT\_LINE(SQLCODE||'---'||SQLERRM);  
END QueryEmp;  
--调用  
 DECLARE  
    v1 employees.first\_name%TYPE;  
    v2 employees.salary%TYPE;  
 BEGIN  
   QueryEmp(**100**, v1, v2);  
   DBMS\_OUTPUT.PUT\_LINE('姓名:'||v1);  
   DBMS\_OUTPUT.PUT\_LINE('工资:'||v2);  
   QueryEmp(**103**, v1, v2);  
   DBMS\_OUTPUT.PUT\_LINE('姓名:'||v1);  
   DBMS\_OUTPUT.PUT\_LINE('工资:'||v2);  
   QueryEmp(**104**, v1, v2);  
   DBMS\_OUTPUT.PUT\_LINE('姓名:'||v1);  
   DBMS\_OUTPUT.PUT\_LINE('工资:'||v2);  
END;

**例12．**计算指定部门的工资总和，并统计其中的职工数量。

**CREATE OR REPLACE  
PROCEDURE proc\_demo  
(  
  dept\_no NUMBER DEFAULT 10,  
    sal\_sum OUT NUMBER,  
    emp\_count OUT NUMBER  
  )  
IS  
BEGIN  
    SELECT SUM(salary), COUNT(\*) INTO sal\_sum, emp\_count  
  FROM employees WHERE department\_id = dept\_no;  
EXCEPTION  
   WHEN NO\_DATA\_FOUND THEN  
      DBMS\_OUTPUT.PUT\_LINE('温馨提示:你需要的数据不存在!');  
   WHEN OTHERS THEN  
      DBMS\_OUTPUT.PUT\_LINE(SQLCODE||'---'||SQLERRM);  
END proc\_demo;  
  
DECLARE  
V\_num NUMBER;  
V\_sum NUMBER(8, 2);  
BEGIN  
  Proc\_demo(30, v\_sum, v\_num);  
DBMS\_OUTPUT.PUT\_LINE('温馨提示:30号部门工资总和：'||v\_sum||',人数：'||v\_num);  
  Proc\_demo(sal\_sum => v\_sum, emp\_count => v\_num);  
DBMS\_OUTPUT.PUT\_LINE('温馨提示:10号部门工资总和：'||v\_sum||',人数：'||v\_num);  
END;**

       在PL/SQL 程序中还可以在块内建立本地函数和过程，这些函数和过程不存储在数据库中，但可以在创建它们的PL/SQL 程序中被重复调用。本地函数和过程在PL/SQL 块的声明部分定义，它们的语法格式与存储函数和过程相同，但不能使用CREATE OR REPLACE 关键字。

**例13：**建立本地过程，用于计算指定部门的工资总和，并统计其中的职工数量；

DECLARE  
V\_num NUMBER;  
V\_sum NUMBER(**8**, **2**);  
PROCEDURE proc\_demo  
  (  
    Dept\_no NUMBER DEFAULT **10**,  
    Sal\_sum OUT NUMBER,  
    Emp\_count OUT NUMBER  
  )  
IS  
BEGIN  
    SELECT SUM(salary), COUNT(\*) INTO sal\_sum, emp\_count   
    FROM employees WHERE department\_id=dept\_no;  
EXCEPTION  
   WHEN NO\_DATA\_FOUND THEN   
      DBMS\_OUTPUT.PUT\_LINE('你需要的数据不存在!');  
   WHEN OTHERS THEN   
      DBMS\_OUTPUT.PUT\_LINE(SQLCODE||'---'||SQLERRM);  
END proc\_demo;  
--调用方法：  
BEGIN  
    Proc\_demo(**30**, v\_sum, v\_num);  
DBMS\_OUTPUT.PUT\_LINE('30号部门工资总和：'||v\_sum||'，人数：'||v\_num);  
    Proc\_demo(sal\_sum => v\_sum, emp\_count => v\_num);  
DBMS\_OUTPUT.PUT\_LINE('10号部门工资总和：'||v\_sum||'，人数：'||v\_num);  
END;

**6.3.3 AUTHID**

过程中的AUTHID 指令可以告诉ORACLE ，这个过程使用谁的权限运行．默任情况下，存储过程会作为调用者的过程运行，但是具有设计者的特权．这称为设计者权利运行．

**例14：**建立过程，使用AUTOID DEFINER；

Connect HR/qaz  
DROP TABLE logtable;  
CREATE table logtable (userid VARCHAR2(**10**), logdate date);  
  
CREATE OR REPLACE PROCEDURE logexecution   
    AUTHID DEFINER  
IS  
BEGIN  
   INSERT INTO logtable (userid, logdate) VALUES (USER, SYSDATE);  
END;  
  
GRANT EXECUTE ON logexecution TO PUBLIC;  
  
CONNECT / AS SYSDBA  
GRANT CONNECT TO testuser1 IDENTIFIED BY userpwd1;  
  
CONNECT testuser1/userpwd1  
INSERT INTO HR.LOGTABLE VALUES (USER, SYSDATE);  
EXECUTE HR.logexecution  
  
CONNECT HR/qaz  
SELECT \* FROM HR.logtable;

**例15：**建立过程，使用AUTOID CURRENT\_USER；

CONNECT HR/qaz  
  
CREATE OR REPLACE PROCEDURE logexecution   
  AUTHID CURRENT\_USER  
IS  
BEGIN  
   INSERT INTO logtable (userid, logdate) VALUES (USER, SYSDATE);  
END;  
  
GRANT EXECUTE ON logexecution TO PUBLIC;  
  
CONNECT testuser1/userpwd1  
INSERT INTO HR.LOGTABLE VALUES (USER, SYSDATE);  
EXECUTE HR.logexecution

**6.3.4 PRAGMA AUTONOMOUS\_TRANSACTION**

ORACLE8i 可以支持事务处理中的事务处理的概念．这种子事务处理可以完成它自己的工作，独立于父事务处理进行提交或者回滚．通过使用这种方法，开发者就能够这样的过程，无论父事务处理是提交还是回滚，它都可以成功执行．

**例16：**建立过程，使用自动事务处理进行日志记录；

DROP TABLE logtable;  
  
CREATE TABLE logtable(  
  Username varchar2(**20**),  
  Dassate\_time date,  
  Mege varchar2(**60**)  
);  
  
CREATE TABLE temp\_table( N number );  
  
CREATE OR REPLACE PROCEDURE log\_message(p\_message varchar2)  
  AS  
  PRAGMA AUTONOMOUS\_TRANSACTION;  
BEGIN  
  INSERT INTO logtable VALUES ( user, sysdate, p\_message );  
  COMMIT;  
END log\_message;  
  
BEGIN  
  Log\_message (‘About to insert into temp\_table‘);  
  INSERT INTO temp\_table VALUES (**1**);  
  Log\_message (‘Rollback to insert into temp\_table‘);  
  ROLLBACK;  
END;  
  
SELECT \* FROM logtable;  
SELECT \* FROM temp\_table;

**例17：**建立过程，没有使用自动事务处理进行日志记录；

CREATE OR REPLACE PROCEDURE log\_message(p\_message varchar2)  
  AS  
BEGIN  
  INSERT INTO logtable VALUES ( user, sysdate, p\_message );  
  COMMIT;  
END log\_message;  
  
BEGIN  
  Log\_message ('About to insert into temp\_table');  
  INSERT INTO temp\_table VALUES (**1**);  
  Log\_message ('Rollback to insert into temp\_table');  
  ROLLBACK;  
END;  
  
SELECT \* FROM logtable;  
SELECT \* FROM temp\_table;

**6.3.5 开发存储过程步骤**

    开发存储过程、函数、包及触发器的步骤如下：

**6.3.5.1 使用文字编辑处理软件编辑存储过程源码**

    使用文字编辑处理软件编辑存储过程源码，要用类似WORD 文字处理软件进行编辑时，要将源码存为文本格式。

**6.3.5.2 在SQLPLUS或用调试工具将存储过程程序进行解释**

    在SQLPLUS或用调试工具将存储过程程序进行解释；

    在SQL>下调试，可用START 或GET 等ORACLE命令来启动解释。如：

SQL>START c:\stat1.sql

    如果使用调式工具，可直接编辑和点击相应的按钮即可生成存储过程。

**6.3.5.3 调试源码直到正确**

    我们不能保证所写的存储过程达到一次就正确。所以这里的调式是每个程序员必须进行的工作之一。在SQLPLUS下来调式主要用的方法是：

         使用 SHOW ERROR命令来提示源码的错误位置；

         使用 user\_errors 数据字典来查看各存储过程的错误位置。

**6.3.5.4 授权执行权给相关的用户或角色**

如果调式正确的存储过程没有进行授权，那就只有建立者本人才可以运行。所以作为应用系统的一部分的存储过程也必须进行授权才能达到要求。在SQL\*PLUS下可以用GRANT命令来进行存储过程的运行授权。

**GRANT语法：**

GRANT system\_privilege | role   
TO user | role | PUBLIC [WITH ADMIN OPTION]  
  
GRANT object\_privilege | ALL ON schema.object   
TO user | role | PUBLIC [WITH GRANT OPTION]  
  
--例子：  
  
CREATE OR REPLACE PUBLIC SYNONYM dbms\_job FOR dbms\_job  
  
GRANT EXECUTE ON dbms\_job TO PUBLIC WITH GRANT OPTION

**6.3.5.5 与过程相关数据字典**

USER\_SOURCE, ALL\_SOURCE, DBA\_SOURCE, USER\_ERRORS,

ALL\_PROCEDURES,USER\_OBJECTS,ALL\_OBJECTS,DBA\_OBJECTS

相关的权限:

CREATE ANY PROCEDURE

DROP ANY PROCEDURE

在SQL\*PLUS 中，可以用DESCRIBE 命令查看过程的名字及其参数表。

DESC[RIBE] Procedure\_name;

**6.3.6 删除过程和函数**

**1．删除过程**

可以使用DROP PROCEDURE命令对不需要的过程进行删除，语法如下：

DROP PROCEDURE [user.]Procudure\_name;

**2．删除函数**

可以使用DROP FUNCTION 命令对不需要的函数进行删除，语法如下：

DROP FUNCTION [user.]Function\_name;  
  
--删除上面实例创建的存储过程与函数  
DROP PROCEDURE logexecution;  
DROP PROCEDURE delemp;  
DROP PROCEDURE insertemp;  
DROP PROCEDURE fireemp;  
DROP PROCEDURE queryemp;  
DROP PROCEDURE proc\_demo;  
DROP PROCEDURE log\_message;  
DROP FUNCTION demo\_fun;  
DROP FUNCTION get\_salary;

**6.3.7**        **过程与函数的比较**

**使用过程与函数具有如下优点：**

1、共同使用的代码可以只需要被编写和测试一次，而被需要该代码的任何应用程序（如：.NET、C++、JAVA、VB程序，也可以是DLL库）调用。

2、这种集中编写、集中维护更新、大家共享（或重用）的方法，简化了应用程序的开发和维护，提高了效率与性能。

3、这种模块化的方法，使得可以将一个复杂的问题、大的程序逐步简化成几个简单的、小的程序部分，进行分别编写、调试。因此使程序的结构清晰、简单，也容易实现。

4、可以在各个开发者之间提供处理数据、控制流程、提示信息等方面的一致性。

5、节省内存空间。它们以一种压缩的形式被存储在外存中，当被调用时才被放入内存进行处理。并且，如果多个用户要执行相同的过程或函数时，就只需要在内存中加载一个该过程或函数。

6、提高数据的安全性与完整性。通过把一些对数据的操作放到过程或函数中，就可以通过是否授予用户有执行该过程或的权限，来限制某些用户对数据进行这些操作。

**过程与函数的相同功能有：**

1、 都使用IN模式的参数传入数据、OUT模式的参数返回数据。

2、 输入参数都可以接受默认值，都可以传值或传引导。

3、 调用时的实际参数都可以使用位置表示法、名称表示法或组合方法。

4、 都有声明部分、执行部分和异常处理部分。

5、 其管理过程都有创建、编译、授权、删除、显示依赖关系等。

**使用过程与函数的原则：**

1、如果需要返回多个值和不返回值，就使用过程；如果只需要返回一个值，就使用函数。

2、过程一般用于执行一个指定的动作，函数一般用于计算和返回一个值。

3、可以SQL语句内部（如表达式）调用函数来完成复杂的计算问题，但不能调用过程。所以这是函数的特色。