第8章 存储过程、函数和包

8.1 存储过程和函数

8.1.1 认识存储过程和函数

存储过程和函数也是一种PL/SQL块，是存入数据库的PL/SQL块。但存储过程和函数不同于已经介绍过的PL/SQL程序，我们通常把PL/SQL程序称为无名块，而存储过程和函数是以命名的方式存储于数据库中的。和PL/SQL程序相比，存储过程有很多优点，具体归纳如下：

* 存储过程和函数以命名的数据库对象形式存储于数据库当中。存储在数据库中的优点是很明显的，因为代码不保存在本地，用户可以在任何客户机上登录到数据库，并调用或修改代码。

* 存储过程和函数可由数据库提供安全保证，要想使用存储过程和函数，需要有存储过程和函数的所有者的授权，只有被授权的用户或创建者本身才能执行存储过程或调用函数。

* 存储过程和函数的信息是写入数据字典的，所以存储过程可以看作是一个公用模块，用户编写的PL/SQL程序或其他存储过程都可以调用它(但存储过程和函数不能调用PL/SQL程序)。一个重复使用的功能，可以设计成为存储过程，比如：显示一张工资统计表，可以设计成为存储过程；一个经常调用的计算，可以设计成为存储函数；根据雇员编号返回雇员的姓名，可以设计成存储函数。

* 像其他高级语言的过程和函数一样，可以传递参数给存储过程或函数，参数的传递也有多种方式。存储过程可以有返回值，也可以没有返回值，存储过程的返回值必须通过参数带回；函数有一定的数据类型，像其他的标准函数一样，我们可以通过对函数名的调用返回函数值。

存储过程和函数需要进行编译，以排除语法错误，只有编译通过才能调用。

8.1.2 创建和删除存储过程

创建存储过程，需要有CREATE PROCEDURE或CREATE ANY PROCEDURE的系统权限。该权限可由系统管理员授予。创建一个存储过程的基本语句如下：

CREATE [OR REPLACE] PROCEDURE 存储过程名[(参数 [IN|OUT|IN OUT] 数据类型...)]

{AS|IS}

[说明部分]

BEGIN

可执行部分

[EXCEPTION

错误处理部分]

END [过程名];

其中：

可选关键字OR REPLACE 表示如果存储过程已经存在，则用新的存储过程覆盖，通常用于存储过程的重建。

参数部分用于定义多个参数(如果没有参数，就可以省略)。参数有三种形式：IN、OUT和IN OUT。如果没有指明参数的形式，则默认为IN。

关键字AS也可以写成IS，后跟过程的说明部分，可以在此定义过程的局部变量。

编写存储过程可以使用任何文本编辑器或直接在SQL*Plus环境下进行，编写好的存储过程必须要在SQL*Plus环境下进行编译，生成编译代码，原代码和编译代码在编译过程中都会被存入数据库。编译成功的存储过程就可以在Oracle环境下进行调用了。

一个存储过程在不需要时可以删除。删除存储过程的人是过程的创建者或者拥有DROP ANY PROCEDURE系统权限的人。删除存储过程的语法如下：

DROP PROCEDURE 存储过程名；

如果要重新编译一个存储过程，则只能是过程的创建者或者拥有ALTER ANY PROCEDURE系统权限的人。语法如下：

ALTER PROCEDURE 存储过程名 COMPILE；

执行(或调用)存储过程的人是过程的创建者或是拥有EXECUTE ANY PROCEDURE系统权限的人或是被拥有者授予EXECUTE权限的人。执行的方法如下：

方法1：

```
EXECUTE 模式名.存储过程名[(参数...)];
```

方法2：

```
BEGIN
模式名.存储过程名[(参数...)];
END;
```

传递的参数必须与定义的参数类型、个数和顺序一致(如果参数定义了默认值，则调用时可以省略参数)。参数可以是变量、常量或表达式，用法参见下一节。

如果是调用本账户下的存储过程，则模式名可以省略。要调用其他账户编写的存储过程，则模式名必须要添加。

以下是一个生成和调用简单存储过程的训练。注意要事先授予创建存储过程的权限。

【训练1】 创建一个显示雇员总人数的存储过程。

步骤1：登录SCOTT账户(或学生个人账户)。

步骤2：在SQL*Plus输入区中，输入以下存储过程：

```
CREATE OR REPLACE PROCEDURE EMP_COUNT
AS
V_TOTAL NUMBER(10);
BEGIN
 SELECT COUNT(*) INTO V_TOTAL FROM EMP;
 DBMS_OUTPUT.PUT_LINE('雇员总人数为：'||V_TOTAL);
END;
```

步骤3：按“执行”按钮进行编译。

如果存在错误，就会显示:

警告: 创建的过程带有编译错误。

如果存在错误，对脚本进行修改，直到没有错误产生。

如果编译结果正确，将显示：

过程已创建。

步骤4：调用存储过程，在输入区中输入以下语句并执行：

```
EXECUTE EMP_COUNT;
```

显示结果为：

雇员总人数为：14

PL/SQL 过程已成功完成。

说明：在该训练中，V_TOTAL变量是存储过程定义的局部变量，用于接收查询到的雇员总人数。

注意：在SQL*Plus中输入存储过程，按“执行”按钮是进行编译，不是执行存储过程。

如果在存储过程中引用了其他用户的对象，比如表，则必须有其他用户授予的对象访问权限。一个存储过程一旦编译成功，就可以由其他用户或程序来引用。但存储过程或函数的所有者必须授予其他用户执行该过程的权限。

存储过程没有参数，在调用时，直接写过程名即可。

【训练2】 在PL/SQL程序中调用存储过程。

步骤1：登录SCOTT账户。

步骤2：授权STUDENT账户使用该存储过程，即在SQL*Plus输入区中，输入以下的命令：

```
GRANT EXECUTE ON EMP_COUNT TO STUDENT
```

授权成功。

步骤3：登录STUDENT账户，在SQL*Plus输入区中输入以下程序：

```
SET SERVEROUTPUT ON
BEGIN
SCOTT.EMP_COUNT;
END;
```

步骤4：执行以上程序，结果为：

雇员总人数为：14

PL/SQL 过程已成功完成。

说明：在本例中，存储过程是由SCOTT账户创建的，STUDEN账户获得SCOTT账户的授权后，才能调用该存储过程。

注意：在程序中调用存储过程，使用了第二种语法。

【训练3】 编写显示雇员信息的存储过程EMP_LIST，并引用EMP_COUNT存储过程。

步骤1：在SQL*Plus输入区中输入并编译以下存储过程：

```
CREATE OR REPLACE PROCEDURE EMP_LIST
AS
 CURSOR emp_cursor IS
SELECT empno,ename FROM emp;
BEGIN
```

```
FOR Emp_record IN emp_cursor LOOP

DBMS_OUTPUT.PUT_LINE(Emp_record.empno||Emp_record.
ename);
END LOOP;
EMP_COUNT;
END;
```

执行结果：

过程已创建。

步骤2：调用存储过程，在输入区中输入以下语句并执行：

EXECUTE EMP_LIST

显示结果为：

7369SMITH

7499ALLEN

7521WARD

7566JONES

执行结果：

雇员总人数为：14

PL/SQL 过程已成功完成。

说明：以上的EMP_LIST存储过程中定义并使用了游标，用来循环显示所有雇员的信息。然后调用已经成功编译的存储过程EMP_COUNT，用来附加显示雇员总人数。通过EXECUTE命令来执行EMP_LIST存储过程。

【练习1】编写显示部门信息的存储过程DEPT_LIST，要求统计出部门个数。

8.1.3 参数传递

参数的作用是向存储过程传递数据，或从存储过程获得返回结果。正确的使用参数可以大大增加存储过程的灵活性和通用性。

参数的类型有三种，如表8-1所示。

表8-1 参数的类型

参数类型	说　　明
IN	定义一个输入参数变量，用于传递参数给存储过程
OUT	定义一个输出参数变量，用于从存储过程获取数据
IN OUT	定义一个输入、输出参数变量，兼有以上两者的功能

参数的定义形式和作用如下：

参数名 IN 数据类型 DEFAULT 值；

定义一个输入参数变量，用于传递参数给存储过程。在调用存储过程时，主程序的实际参数可以是常量、有值变量或表达式等。DEFAULT 关键字为可选项，用来设定参数的默认值。如果在调用存储过程时不指明参数，则参数变量取默认值。在存储过程中，输入变量接收主程序传递的值，但不能对其进行赋值。

参数名 OUT 数据类型；

定义一个输出参数变量，用于从存储过程获取数据，即变量从存储过程中返回值给主程序。

在调用存储过程时，主程序的实际参数只能是一个变量，而不能是常量或表达式。在存储过程中，参数变量只能被赋值而不能将其用于赋值，在存储过程中必须给输出变量至少赋值一次。

参数名 IN OUT 数据类型 DEFAULT 值;

定义一个输入、输出参数变量，兼有以上两者的功能。在调用存储过程时，主程序的实际参数只能是一个变量，而不能是常量或表达式。DEFAULT 关键字为可选项，用来设定参数的默认值。在存储过程中，变量接收主程序传递的值，同时可以参加赋值运算，也可以对其进行赋值。在存储过程中必须给变量至少赋值一次。

如果省略IN、OUT或IN OUT，则默认模式是IN。

【训练1】 编写给雇员增加工资的存储过程CHANGE_SALARY，通过IN类型的参数传递要增加工资的雇员编号和增加的工资额。

步骤1：登录SCOTT账户。

步骤2：在SQL*Plus输入区中输入以下存储过程并执行：

```
CREATE OR REPLACE PROCEDURE CHANGE_SALARY(P_EMPNO IN NUMBER DEFAULT 7788,P_RAISE NUMBER DEFAULT 10)
AS
 V_ENAME VARCHAR2(10);
```

```
V_SAL NUMBER(5);
BEGIN
SELECT ENAME,SAL INTO V_ENAME,V_SAL FROM EMP WHERE EMPNO=P_EMPNO;
UPDATE EMP SET SAL=SAL+P_RAISE WHERE EMPNO=P_EMPNO;
DBMS_OUTPUT.PUT_LINE('雇员'||V_ENAME||'的工资被改为'||TO_CHAR(V_SAL+P_RAISE));
```

```
COMMIT;
EXCEPTION
 WHEN OTHERS THEN
DBMS_OUTPUT.PUT_LINE('发生错误，修改失败！');
ROLLBACK;
END;
```

执行结果为：

过程已创建。

步骤3：调用存储过程，在输入区中输入以下语句并执行：

EXECUTE CHANGE_SALARY(7788,80)

显示结果为：

雇员SCOTT的工资被改为3080

说明：从执行结果可以看到，雇员SCOTT的工资已由原来的3000改为3080。

参数的值由调用者传递，传递的参数的个数、类型和顺序应该和定义的一致。如果顺序不一致，可以采用以下调用方法。如上例，执行语句可以改为：

```
EXECUTE CHANGE_SALARY(P_RAISE=>80,P_EMPNO=>7788);
```

可以看出传递参数的顺序发生了变化，并且明确指出了参数名和要传递的值，=>运算符左侧是参数名，右侧是参数表达式，这种赋值方法的意义较清楚。

【练习1】创建插入雇员的存储过程INSERT_EMP，并将雇员编号等作为参数。

在设计存储过程的时候，也可以为参数设定默认值，这样调用者就可以不传递或少传递参数了。

【训练2】 调用存储过程CHANGE_SALARY，不传递参数，使用默认参数值。

在SQL*Plus输入区中输入以下命令并执行：

EXECUTE CHANGE_SALARY

显示结果为：

雇员SCOTT的工资被改为3090

说明：在存储过程的调用中没有传递参数，而是采用了默认值7788和10，即默认雇员号为7788，增加的工资为10。

【训练3】 使用OUT类型的参数返回存储过程的结果。

步骤1：登录SCOTT账户。

步骤2：在SQL*Plus输入区中输入并编译以下存储过程：

```
CREATE OR REPLACE PROCEDURE EMP_COUNT(P_TOTAL
OUT NUMBER)
AS
BEGIN
SELECT COUNT(*) INTO P_TOTAL FROM EMP;
END;
```

执行结果为：

过程已创建。

步骤3：输入以下程序并执行：

```
DECLARE
V_EMPCOUNT NUMBER;
BEGIN
EMP_COUNT(V_EMPCOUNT);
DBMS_OUTPUT.PUT_LINE('雇员总人数为：'||V_EMPCOUNT);
END;
```

显示结果为：

雇员总人数为：14

PL/SQL 过程已成功完成。

说明：在存储过程中定义了OUT类型的参数P_TOTAL，在主程序调用该存储过程时，传递了参数V_EMPCOUNT。在存储过程中的SELECT...INTO... 语句中对P_TOTAL进行赋值，赋值结果由V_EMPCOUNT变量带回给主程序并显示。

以上程序要覆盖同名的EMP_COUNT存储过程，如果不使用OR REPLACE选项，就会出现以下错误：

ERROR 位于第 1 行:

ORA-00955: 名称已由现有对象使用。

【练习2】创建存储过程，使用OUT类型参数获得雇员经理名。

【训练4】 使用IN OUT类型的参数，给电话号码增加区码。

步骤1：登录SCOTT账户。

步骤2：在SQL*Plus输入区中输入并编译以下存储过程：

```
CREATE OR REPLACE PROCEDURE ADD_REGION(P_HPONE_NUM IN OUT VARCHAR2)
AS
BEGIN
 P_HPONE_NUM:='024-'||P_HPONE_NUM;
END;
```

执行结果为：

过程已创建。

步骤3：输入以下程序并执行：

```
SET SERVEROUTPUT ON
DECLARE
V_PHONE_NUM VARCHAR2(15);
BEGIN
V_PHONE_NUM:='26731092';
ADD_REGION(V_PHONE_NUM);
DBMS_OUTPUT.PUT_LINE('新的电话号码：'||V_PHONE_NUM);
END;
```

显示结果为：

新的电话号码：024-26731092

PL/SQL 过程已成功完成。

说明：变量V_HPONE_NUM既用来向存储过程传递旧电话号码，也用来向主程序返回新号码。新的号码在原来基础上增加了区号024和-。

8.1.4 创建和删除存储函数

创建函数，需要有CREATE PROCEDURE或CREATE ANY PROCEDURE的系统权限。该权限可由系统管理员授予。创建存储函数的语法和创建存储过程的类似，即

CREATE [OR REPLACE] FUNCTION 函数名[(参数[IN] 数据类型...)]

RETURN 数据类型

{AS|IS}

[说明部分]

BEGIN

可执行部分

RETURN (表达式)

[EXCEPTION

错误处理部分]

END [函数名];

其中，参数是可选的，但只能是IN类型(IN关键字可以省略)。

在定义部分的RETURN 数据类型，用来表示函数的数据类型，也就是返回值的类型，此部分不可省略。

在可执行部分的RETURN(表达式)，用来生成函数的返回值，其表达式的类型应该和定义部分说明的函数返回值的数据类型一致。在函数的执行部分可以有多个RETURN语句，但只有一个RETURN语句会被执行，一旦执行了RETURN语句，则函数结束并返回调用环境。

一个存储函数在不需要时可以删除，但删除的人应是函数的创建者或者是拥有DROP ANY PROCEDURE系统权限的人。其语法如下：

DROP FUNCTION 函数名;

重新编译一个存储函数时，编译的人应是函数的创建者或者拥有ALTER ANY PROCEDURE系统权限的人。重新编译一个存储函数的语法如下：

ALTER PROCEDURE 函数名 COMPILE;

函数的调用者应是函数的创建者或拥有EXECUTE ANY PROCEDURE系统权限的人，或是被函数的拥有者授予了函数执行权限的账户。函数的引用和存储过程不同，函数要出现在程序体中，可以参加表达式的运算或单独出现在表达式中，其形式如下：

变量名:=函数名(...)

【训练1】 创建一个通过雇员编号返回雇员名称的函数GET_EMP_NAME。

步骤1：登录SCOTT账户。

步骤2：在SQL*Plus输入区中输入以下存储函数并编译：

```
CREATE OR REPLACE FUNCTION GET_EMP_NAME(P_EMPNO
NUMBER DEFAULT 7788)
RETURN VARCHAR2
AS
 V_ENAME VARCHAR2(10);
BEGIN
SELECT ENAME INTO V_ENAME FROM EMP WHERE
EMPNO=P_EMPNO;
```

```
 RETURN(V_ENAME);
EXCEPTION
 WHEN NO_DATA_FOUND THEN
  DBMS_OUTPUT.PUT_LINE('没有该编号雇员！ ');
  RETURN (NULL);
 WHEN TOO_MANY_ROWS THEN
  DBMS_OUTPUT.PUT_LINE('有重复雇员编号！ ');
  RETURN (NULL);
 WHEN OTHERS THEN
  DBMS_OUTPUT.PUT_LINE('发生其他错误！ ');
  RETURN (NULL);
END;
```

步骤3：调用该存储函数，输入并执行以下程序：

```
BEGIN
DBMS_OUTPUT.PUT_LINE('雇员7369的名称是：'||GET_EMP_NAME(7369));
 DBMS_OUTPUT.PUT_LINE('雇员7839的名称是：'||GET_EMP_NAME(7839));
END;
```

显示结果为：

雇员7369的名称是：SMITH

雇员7839的名称是：KING

PL/SQL 过程已成功完成。

说明：函数的调用直接出现在程序的DBMS_OUTPUT.PUT_LINE语句中，作为字符串表达式的一部分。如果输入了错误的雇员编号，就会在函数的错误处理部分输出错误信息。试修改雇员编号，重新运行调用部分。

【练习1】创建一个通过部门编号返回部门名称的存储函数GET_DEPT_NAME。

【练习2】将函数的执行权限授予STUDENT账户，然后登录STUDENT账户调用。

8.1.5 存储过程和函数的查看

可以通过对数据字典的访问来查询存储过程或函数的有关信息，如果要查询当前用户的存储过程或函数的源代码，可以通过对USER_SOURCE数据字典视图的查询得到。USER_SOURCE的结构如下：

```
DESCRIBE USER_SOURCE
```

结果为：

名称 是否为空? 类型

NAME	VARCHAR2(30)
TYPE	VARCHAR2(12)
LINE	NUMBER
TEXT	VARCHAR2(4000)

说明：里面按行存放着过程或函数的脚本，NAME是过程或函数名，TYPE 代表类型(PROCEDURE或FUNCTION)，LINE是行号，TEXT 为脚本。

【训练1】 查询过程EMP_COUNT的脚本。

在SQL*Plus中输入并执行如下查询：

```
select TEXT  from user_source WHERE NAME='EMP_COUNT';
```

结果为：

```
TEXT
------------------------------------------------------------------------
PROCEDURE EMP_COUNT(P_TOTAL OUT NUMBER)
AS
BEGIN
 SELECT COUNT(*) INTO P_TOTAL FROM EMP;
END;
```

【训练2】 查询过程GET_EMP_NAME的参数。

在SQL*Plus中输入并执行如下查询：

DESCRIBE GET_EMP_NAME

结果为：

FUNCTION GET_EMP_NAME RETURNS VARCHAR2

参数名称 类型 输入/输出默认值？

------------------------------------ ------------------------------ -------

--------- ------------

P_EMPNO NUMBER(4) IN DEFAULT

【训练3】 在发生编译错误时，显示错误。

SHOW ERRORS

以下是一段编译错误显示：

LINE/COL ERROR

---------- --

4/2 PL/SQL: SQL Statement ignored

4/36 PLS-00201: 必须说明标识符 'EMPP'

说明：查询一个存储过程或函数是否是有效状态(即编译成功)，可以使用数据字典USER_OBJECTS的STATUS列。

【训练4】 查询EMP_LIST存储过程是否可用：

```
SELECT STATUS FROM USER_OBJECTS WHERE OBJECT_NAME='EMP_LIST';
```

结果为：

```
STATUS
-----------
VALID
```

说明：VALID表示该存储过程有效(即通过编译)，INVALID表示存储过程无效或需要重新编译。当Oracle调用一个无效的存储过程或函数时，首先试图对其进行编译，如果编译成功则将状态置成VALID并执行，否则给出错误信息。

当一个存储过程编译成功，状态变为VALID，会不会在某些情况下变成INVALID。结论是完全可能的。比如一个存储过程中包含对表的查询，如果表被修改或删除，存储过程就会变成无效INVALID。所以要注意存储过程和函数对其他对象的依赖关系。

如果要检查存储过程或函数的依赖性，可以通过查询数据字典USER_DENPENDENCIES来确定，该表结构如下：

DESCRIBE USER_DEPENDENCIES;

结果：

名称 是否为空? 类型

NAME NOT NULL VARCHAR2(30)

TYPE VARCHAR2(12)

REFERENCED_OWNER VARCHAR2(30)

REFERENCED_NAME VARCHAR2(64)

REFERENCED_TYPE VARCHAR2(12)

REFERENCED_LINK_NAME VARCHAR2(128)

SCHEMAID NUMBER

DEPENDENCY_TYPE VARCHAR2(4)

说明：NAME为实体名，TYPE为实体类型，REFERENCED_OWNER为涉及到的实体拥有者账户，REFERENCED_NAME为涉及到的实体名，REFERENCED_TYPE为涉及到的实体类型。

【训练5】 查询EMP_LIST存储过程的依赖性。

```
SELECT REFERENCED_NAME,REFERENCED_TYPE
FROM USER_DEPENDENCIES WHERE NAME='EMP_LIST';
```

执行结果：

```
REFERENCED_NAME
REFERENCED_TYPE
--------------------------------------------------------------------------------
---------- -------------------------
```

STANDARD	PACKAGE
SYS_STUB_FOR_PURITY_ANALYSIS	PACKAGE
DBMS_OUTPUT	PACKAGE
DBMS_OUTPUT	SYNONYM

DBMS_OUTPUT NON-EXISTENT

EMP TABLE

EMP_COUNT

PROCEDURE

说明：可以看出存储过程EMP_LIST依赖一些系统包、EMP表和EMP_COUNT存储过程。如果删除了EMP表或EMP_COUNT存储过程，EMP_LIST将变成无效。

还有一种情况需要我们注意：如果一个用户A被授予执行属于用户B的一个存储过程的权限，在用户B的存储过程中，访问到用户C的表，用户B被授予访问用户C的表的权限，但用户A没有被授予访问用户C表的权限，那么用户A调用用户B的存储过程是失败的还是成功的呢？答案是成功的。如果读者有兴趣，不妨进行一下实际测试。

8.2 包

8.2.1 包的概念和组成

包是用来存储相关程序结构的对象，它存储于数据字典中。包由两个分离的部分组成：包头(PACKAGE)和包体(PACKAGE BODY)。包头是包的说明部分，是对外的操作接口，对应用是可见的；包体是包的代码和实现部分，对应用来说是不可见的黑盒。

包中可以包含的程序结构如表8-2所示。

表8-2 包中包含的程序结构

程序结构	说　　明
过程(PROCUDURE)	带参数的命名的程序模块
函数(FUNCTION)	带参数、具有返回值的命名的程序模块
变量(VARIABLE)	存储变化的量的存储单元
常量(CONSTANT)	存储不变的量的存储单元
游标(CURSOR)	用户定义的数据操作缓存区，在可执行部分使用
类型(TYPE)	用户定义的新的结构类型
异常(EXCEPTION)	在标准包中定义或由用户自定义，用于处理程序错误

说明部分可以出现在包的三个不同的部分：出现在包头中的称为公有元素，出现在包体中的称为私有元素，出现在包体的过程(或函数)中的称为局部变量。它们的性质有所不同，如表8-3所示。

表8-3 包中元素的性质

元　素	说　　明	有效范围
公有元素(PUBLIC)	在包头中说明，在包体中具体定义	在包外可见并可以访问，对整个应用的全过程有效
私有元素(PRIVATE)	在包体的说明部分说明	只能被包内部的其他部分访问
局部变量(LOCAL)	在过程或函数的说明部分说明	只能在定义变量的过程或函数中使用

在包体中出现的过程或函数，如果需要对外公用，就必须在包头中说明，包头中的说明应该和包体中的说明一致。

包有以下优点：

* 包可以方便地将存储过程和函数组织到一起，每个包又是相互独立的。在不同的包中，过程、函数都可以重名，这解决了在同一个用户环境中命名的冲突问题。

* 包增强了对存储过程和函数的安全管理，对整个包的访问权只需一次授予。

* 在同一个会话中，公用变量的值将被保留，直到会话结束。

* 区分了公有过程和私有过程，包体的私有过程增加了过程和函数的保密性。

* 包在被首次调用时，就作为一个整体被全部调入内存，减少了多次访问过程或函数的I/O次数。

8.2.2 创建包和包体

包由包头和包体两部分组成，包的创建应该先创建包头部分，然后创建包体部分。创建、删除和编译包的权限同创建、删除和编译存储过程的权限相同。

创建包头的简要语句如下：

```
CREATE [OR REPLACE] PACKAGE 包名
{IS|AS}
```

公有变量定义

公有类型定义

公有游标定义

公有异常定义

函数说明

过程说明

END;

创建包体的简要语法如下：

CREATE [OR REPLACE] PACKAGE BODY 包名 {IS|AS}

私有变量定义

私有类型定义

私有游标定义

私有异常定义

函数定义

过程定义

END;

包的其他操作命令包括：

删除包头：

DROP PACKAGE 包头名

删除包体：

DROP PACKAGE BODY 包体名

重新编译包头：

ALTER PACKAGE 包名 COMPILE PACKAGE

重新编译包体：

ALTER PACKAGE 包名 COMPILE PACKAGE BODY

在包头中说明的对象可以在包外调用，调用的方法和调用单独的过程或函数的方法基本相同，惟一的区别就是要在调用的过程或函数名前加上包的名字(中间用“.”分隔)。但要注意，不同的会话将单独对包的公用变量进行初始化，所以不同的会话对包的调用属于不同的应用。

8.2.3 系统包

Oracle预定义了很多标准的系统包，这些包可以在应用中直接使用，比如在训练中我们使用的DBMS_OUTPUT包，就是系统包。PUT_LINE是该包的一个函数。常用系统包如表8-4所示。

表8-4 常 用 系 统 包

系统包	说　　明
DBMS_OUTPUT	在 SQL*Plus 环境下输出信息
DBMS_DDL	编译过程函数和包
DBMS_SESSION	改变用户的会话，初始化包等
DBMS_TRANSACTION	控制数据库事务
DBMS_MAIL	连接 Oracle*Mail
DBMS_LOCK	进行复杂的锁机制管理
DBMS_ALERT	识别数据库事件告警
DBMS_PIPE	通过管道在会话间传递信息
DBMS_JOB	管理 Oracle 的作业
DBMS_LOB	操纵大对象
DBMS_SQL	执行动态 SQL 语句

8.2.4 包的应用

在SQL*Plus环境下，包和包体可以分别编译，也可以一起编译。如果分别编译，则要先编译包头，后编译包体。如果在一起编译，则包头写在前，包体在后，中间用“/”分隔。

可以将已经存在的存储过程或函数添加到包中，方法是去掉过程或函数创建语句的CREATE OR REPLACE部分，将存储过程或函数复制到包体中，然后重新编译即可。

如果需要将私有过程或函数变成共有过程或函数的话，将过程或函数说明部分复制到包头说明部分，然后重新编译就可以了。

【训练1】 创建管理雇员信息的包EMPLOYE，它具有从EMP表获得雇员信息，修改雇员名称，修改雇员工资和写回EMP表的功能。

步骤1：登录SCOTT账户，输入以下代码并编译：

```
CREATE OR REPLACE PACKAGE EMPLOYE --包头部分
IS
 PROCEDURE SHOW_DETAIL;
 PROCEDURE GET_EMPLOYE(P_EMPNO NUMBER);
 PROCEDURE SAVE_EMPLOYE;
 PROCEDURE CHANGE_NAME(P_NEWNAME VARCHAR2);
```

```
PROCEDURE CHANGE_SAL(P_NEWSAL NUMBER);
END EMPLOYE;
/
CREATE OR REPLACE PACKAGE BODY EMPLOYE --包体部分
IS
 EMPLOYE EMP%ROWTYPE;
---------- 显示雇员信息 ----------
PROCEDURE SHOW_DETAIL
AS
BEGIN
```

```
DBMS_OUTPUT.PUT_LINE(‘----- 雇员信息 -----’);
DBMS_OUTPUT.PUT_LINE('雇员编号：'||EMPLOYE.EMPNO);
DBMS_OUTPUT.PUT_LINE('雇员名称：'||EMPLOYE.ENAME);
 DBMS_OUTPUT.PUT_LINE('雇员职务：'||EMPLOYE.JOB);
DBMS_OUTPUT.PUT_LINE('雇员工资：'||EMPLOYE.SAL);
DBMS_OUTPUT.PUT_LINE('部门编号：'||EMPLOYE.DEPTNO);
END SHOW_DETAIL;
```

```
---------------- 从EMP表取得一个雇员 -------------------
 PROCEDURE GET_EMPLOYE(P_EMPNO NUMBER)
AS
BEGIN
SELECT * INTO EMPLOYE FROM EMP WHERE EMPNO=P_EMPNO;
DBMS_OUTPUT.PUT_LINE('获取雇员'||EMPLOYE.ENAME||'信息成功');
 EXCEPTION
 WHEN OTHERS THEN
   DBMS_OUTPUT.PUT_LINE('获取雇员信息发生错误！');
END GET_EMPLOYE;
```

```
---------------------- 保存雇员到EMP表 -------------------------
PROCEDURE SAVE_EMPLOYE
AS
BEGIN
UPDATE EMP SET ENAME=EMPLOYE.ENAME,
SAL=EMPLOYE.SAL WHERE EMPNO=
EMPLOYE.EMPNO;
 DBMS_OUTPUT.PUT_LINE('雇员信息保存完成！ ');
END SAVE_EMPLOYE;
```

```
---------------------------- 修改雇员名称 ----------------------------
PROCEDURE CHANGE_NAME(P_NEWNAME VARCHAR2)
 AS
BEGIN
 EMPLOYE.ENAME:=P_NEWNAME;
 DBMS_OUTPUT.PUT_LINE('修改名称完成！ ');
END CHANGE_NAME;
```

```
---------------------------- 修改雇员工资 ------------------------
PROCEDURE CHANGE_SAL(P_NEWSAL NUMBER)
AS
BEGIN
 EMPLOYE.SAL:=P_NEWSAL;
 DBMS_OUTPUT.PUT_LINE('修改工资完成！ ');
END CHANGE_SAL;
END EMPLOYE;
```

步骤2：获取雇员7788的信息:

```
SET SERVEROUTPUT ON
EXECUTE EMPLOYE.GET_EMPLOYE(7788);
```

结果为：

```
获取雇员SCOTT信息成功
PL/SQL 过程已成功完成。
```

步骤3：显示雇员信息：

```
EXECUTE EMPLOYE.SHOW_DETAIL;
```

结果为：

```
---------- 雇员信息 ----------
雇员编号：7788
雇员名称：SCOTT
雇员职务：ANALYST
雇员工资：3000
部门编号：20
PL/SQL 过程已成功完成。
```

步骤4：修改雇员工资：

EXECUTE EMPLOYE.CHANGE_SAL(3800);

结果为：

修改工资完成！

PL/SQL 过程已成功完成。

步骤5：将修改的雇员信息存入EMP表

EXECUTE EMPLOYE.SAVE_EMPLOYE;

结果为：

雇员信息保存完成！

PL/SQL 过程已成功完成。

说明：该包完成将EMP表中的某个雇员的信息取入内存记录变量，在记录变量中进行修改编辑，在确认显示信息正确后写回EMP表的功能。记录变量EMPLOYE用来存储取得的雇员信息，定义为私有变量，只能被包的内部模块访问。

【练习1】为包增加修改雇员职务和部门编号的功能。

8.3 阶段训练

下面的训练通过定义和创建完整的包EMP_PK并综合运用本章的知识，完成对雇员表的插入、删除等功能，包中的主要元素解释如表8-5所示。

表8-5 完整的雇员包EMP_PK的成员

程序结构	类 型	说 明
V_EMP_COUNT	公有变量	跟踪雇员的总人数变化，插入、删除雇员的同时修改该变量的值
INIT	公有过程	对包进行初始化，初始化雇员人数和工资修改的上、下限
LIST_EMP	公有过程	显示雇员列表
INSERT_EMP	公有过程	通过编号插入新雇员
DELETE_EMP	公有过程	通过编号删除雇员
CHANGE_EMP_SAL	公有过程	通过编号修改雇员工资
V_MESSAGE	私有变量	存放准备输出的信息
C_MAX_SAL	私有变量	对工资修改的上限
C_MIN_SAL	私有变量	对工资修改的下限
SHOW_MESSAGE	私有过程	显示私有变量 V_MESSAGE 中的信息
EXIST_EMP	私有函数	判断某个编号的雇员是否存在，该函数被 INSERT_EMP、DELETE_EMP 和 CHANGE_EMP_SAL 等过程调用

【训练1】 完整的雇员包EMP_PK的创建和应用。

步骤1：在SQL*Plus中登录SCOTT账户，输入以下包头和包体部分，按“执行”按钮编译：

```
CREATE OR REPLACE PACKAGE EMP_PK
--包头部分
IS
V_EMP_COUNT NUMBER(5);
--雇员人数
PROCEDURE INIT(P_MAX NUMBER,P_MIN NUMBER); --初始化
PROCEDURE LIST_EMP;
--显示雇员列表
```

```
PROCEDURE INSERT_EMP(P_EMPNO
NUMBER,P_ENAMEVARCHAR2,P_JOB VARCHAR2,
P_SAL NUMBER);
--插入雇员
PROCEDURE DELETE_EMP(P_EMPNO NUMBER);          --删除雇员
 PROCEDURE  CHANGE_EMP_SAL(P_EMPNO  NUMBER,P_SAL NUMBER);
--修改雇员工资
END EMP_PK;
/CREATE OR REPLACE PACKAGE BODY EMP_PK
 --包体部分
IS
V_MESSAGE VARCHAR2(50); --显示信息
```

```
 V_MAX_SAL NUMBER(7); --工资上限
V_MIN_SAL NUMBER(7); --工资下限
FUNCTION EXIST_EMP(P_EMPNO NUMBER)    RETURN
BOOLEAN; --判断雇员是否存在函数
PROCEDURE SHOW_MESSAGE; --显示信息过程
------------------------------ 初始化过程 --------------------------
PROCEDURE INIT(P_MAX NUMBER,P_MIN NUMBER)
IS
BEGIN
SELECT COUNT(*) INTO V_EMP_COUNT FROM EMP;
```

```
    V_MAX_SAL:=P_MAX;
    V_MIN_SAL:=P_MIN;
    V_MESSAGE:='初始化过程已经完成！';
    SHOW_MESSAGE;
  END INIT;
---------------------------- 显示雇员列表过程 --------------------
  PROCEDURE LIST_EMP
   IS
  BEGIN
```

```
 DBMS_OUTPUT.PUT_LINE('姓名      职务      工资');
FOR emp_rec IN (SELECT * FROM EMP)
LOOP

DBMS_OUTPUT.PUT_LINE(RPAD(emp_rec.ename,10,'')||RPAD(emp
_rec.job,10,' ')||TO_CHAR(emp_rec.sal));
 END LOOP;
 DBMS_OUTPUT.PUT_LINE('雇员总人数'||V_EMP_COUNT);
END LIST_EMP;
```

```
---------------------------- 插入雇员过程 ----------------------------
        PROCEDUREINSERT_EMP(P_EMPNO
        NUMBER,P_ENAMEVARCHAR2,P_JOB VARCHAR2,P_SAL
    NUMBER)
         IS
        BEGIN
        IF NOT EXIST_EMP(P_EMPNO) THEN
        INSERT INTO EMP(EMPNO,ENAME,JOB,SAL)
        VALUES(P_EMPNO,P_ENAME,P_JOB,P_SAL);
        COMMIT;
        V_EMP_COUNT:=V_EMP_COUNT+1;
        V_MESSAGE:='雇员'||P_EMPNO||'已插入!';
        ELSE
```

```
V_MESSAGE:='雇员'||P_EMPNO||'已存在,不能插入!';
 END IF;
 SHOW_MESSAGE;
 EXCEPTION
WHEN OTHERS THEN
 V_MESSAGE:='雇员'||P_EMPNO||'插入失败!';
 SHOW_MESSAGE;
 END INSERT_EMP;
```

```
---------------------- 删除雇员过程 -------------------
 PROCEDURE DELETE_EMP(P_EMPNO NUMBER)
IS
BEGIN
IF EXIST_EMP(P_EMPNO) THEN
DELETE FROM EMP WHERE EMPNO=P_EMPNO;
COMMIT;
 V_EMP_COUNT:=V_EMP_COUNT-1;
 V_MESSAGE:='雇员'||P_EMPNO||'已删除!';
 ELSE
```

```
V_MESSAGE:='雇员'||P_EMPNO||'不存在，不能删除!';
END IF;
SHOW_MESSAGE;
 EXCEPTION
 WHEN OTHERS THEN
 V_MESSAGE:='雇员'||P_EMPNO||'删除失败!';
 SHOW_MESSAGE;
END DELETE_EMP;
```

```
---------------------------------- 修改雇员工资过程 ------------------------------
----
    PROCEDURE  CHANGE_EMP_SAL(P_EMPNO  NUMBER,P_SAL
NUMBER)
     IS
     BEGIN
     IF (P_SAL>V_MAX_SAL OR P_SAL<V_MIN_SAL) THEN
     V_MESSAGE:='工资超出修改范围!';
    ELSIF NOT EXIST_EMP(P_EMPNO) THEN
    V_MESSAGE:='雇员'||P_EMPNO||'不存在，不能修改工资!';
```

```
        ELSE
         UPDATE EMP SET SAL=P_SAL WHERE
EMPNO=P_EMPNO;
        COMMIT;
        V_MESSAGE:='雇员'||P_EMPNO||'工资已经修改!';
        END IF;
        SHOW_MESSAGE;
        EXCEPTION
         WHEN OTHERS THEN
         V_MESSAGE:='雇员'||P_EMPNO||'工资修改失败!';
         SHOW_MESSAGE;
         END CHANGE_EMP_SAL;
```

---------- 显示信息过程 ----------

```
 PROCEDURE SHOW_MESSAGE
IS
BEGIN
 DBMS_OUTPUT.PUT_LINE('提示信息：'||V_MESSAGE);
END SHOW_MESSAGE;
```

```
---------------------- 判断雇员是否存在函数 ------------------
    FUNCTION EXIST_EMP(P_EMPNO NUMBER)
    RETURN BOOLEAN
    IS
   V_NUM NUMBER; --局部变量
   BEGIN
   SELECT COUNT(*) INTO V_NUM FROM EMP WHERE
EMPNO=P_EMPNO;
```

```
IF V_NUM=1 THEN
   RETURN TRUE;
 ELSE
 RETURN FALSE;
END IF;
END EXIST_EMP;
----------------------------
END EMP_PK;
```

结果为：

程序包已创建。

程序包主体已创建。

步骤2：初始化包：

```
SET SERVEROUTPUT ON
EXECUTE EMP_PK.INIT(6000,600);
```

显示为：

提示信息：初始化过程已经完成！

步骤3：显示雇员列表：

```
EXECUTE EMP_PK.LIST_EMP;
```

显示为：

```
姓名                职务      工资
SMITH               CLERK     1560
ALLEN               SALESMAN  1936
WARD                SALESMAN  1830
JONES               MANAGER   2975
...
雇员总人数：14
PL/SQL 过程已成功完成。
```

步骤4：插入一个新记录：

```
EXECUTE EMP_PK.INSERT_EMP(8001,'小王','CLERK',1000);
```

显示结果为：

提示信息：雇员8001已插入!

PL/SQL 过程已成功完成。

步骤5：通过全局变量V_EMP_COUNT查看雇员人数：

```
BEGIN
DBMS_OUTPUT.PUT_LINE(EMP_PK.V_EMP_COUNT);
END;
```

显示结果为：

15

PL/SQL 过程已成功完成。

步骤6：删除新插入记录：

EXECUTE EMP_PK.DELETE_EMP(8001);

显示结果为：

提示信息：雇员8001已删除!

PL/SQL 过程已成功完成。

再次删除该雇员：

EXECUTE EMP_PK.DELETE_EMP(8001);

结果为：

提示信息：雇员8001不存在，不能删除!

步骤7：修改雇员工资：

EXECUTE EMP_PK.CHANGE_EMP_SAL(7788,8000);

显示结果为：

提示信息：工资超出修改范围!

PL/SQL 过程已成功完成。

步骤8：授权其他用户调用包：

如果是另外一个用户要使用该包，必须由包的所有者授权，下面授予STUDEN账户对该包的使用权：

GRANT EXECUTE ON EMP_PK TO STUDENT;

每一个新的会话要为包中的公用变量开辟新的存储空间，所以需要重新执行初始化过程。两个会话的进程互不影响。

步骤9：其他用户调用包。

启动另外一个SQL*Plus，登录STUDENT账户，执行以下过程：

```
SET SERVEROUTPUT ON
EXECUTE SCOTT.EMP_PK. EMP_PK.INIT(5000,700);
```

结果为：

```
提示信息：初始化过程已经完成！
PL/SQL 过程已成功完成。
```

说明：在初始化中设置雇员的总人数和修改工资的上、下限，初始化后V_EMP_COUNT为14人，插入雇员后V_EMP_COUNT为15人。V_EMP_COUNT为公有变量，所以可以在外部程序中使用DBMS_OUTPUT.PUT_LINE输出，引用时用EMP_PK.V_EMP_COUNT的形式，说明所属的包。而私有变量V_MAX_SAL和V_MIN_SAL不能被外部访问，只能通过内部过程来修改。同样，EXIST_EMP和SHOW_MESSAGE也是私有过程，也只能在过程体内被其他模块引用。

注意：在最后一个步骤中，因为STUDENT模式调用了SCOTT模式的包，所以包名前要增加模式名SCOTT。不同的会话对包的调用属于不同的应用，所以需要重新进行初始化。

8.4 练习

1．如果存储过程的参数类型为OUT，那么调用时传递的参数应该为：

A．常量　　B．表达式

C．变量　　D．都可以

2．下列有关存储过程的特点说法错误的是：

A．存储过程不能将值传回调用的主程序

B．存储过程是一个命名的模块

C．编译的存储过程存放在数据库中

D．一个存储过程可以调用另一个存储过程

3．下列有关函数的特点说法错误的是：

A．函数必须定义返回类型

B．函数参数的类型只能是IN

C．在函数体内可以多次使用RETURN语句

D．函数的调用应使用EXECUTE命令

4．包中不能包含的元素为:

A．存储过程　　B．存储函数

C．游标　　D．表

5．下列有关包的使用说法错误的是：

A．在不同的包内模块可以重名

B．包的私有过程不能被外部程序调用

C．包体中的过程和函数必须在包头部分说明

D．必须先创建包头，然后创建包体