Tech Mahindra

Procedures and Functions

Copyright © 2013 Tech Mahindra. All rights reserved.

Sub-Programs

- Subprograms are named PL/SQL blocks that can accept parameters and be invoked from a calling environment.

- PL/SQL has two types of subprograms, ***procedures*** and ***functions***.
  - A procedure that performs an action
  - A function that computes a value

- A Subprogram
  - is based on standard PL/SQL block structure
  - provide modularity, reusability, extensibility and maintainability.
  - provide easy maintenance, improved data security and integrity, improved performance, and improved code clarity

Copyright © 2013 Tech Mahindra. All rights reserved.

Benefits of Sub Programs

- Procedures and functions have many benefits in addition to modularizing application development:
- Easy maintenance
- Improved data security and Integrity
- Improved performance
- Improved code clarity

Copyright © 2013 Tech Mahindra. All rights reserved.

Easy maintenance

- Sub-programs are located in one location and hence it is easy to:
  - Modify routines online without interfering with other users
  - Modify one routine to affect multiple applications
  - Modify one routine to eliminate duplicate testing

Copyright © 2013 Tech Mahindra. All rights reserved.

Improved data security and integrity

- They help controlling indirect access to database objects from non-privileged users with the help of security privileges. As a subprogram is executed with its definer's right by default, it is easy to restrict the access privilege by granting a privilege only to execute the subprogram to a user. And also ensures that related actions are performed together, or not at all.

Copyright © 2013 Tech Mahindra. All rights reserved.

Improved performance

- After a subprogram is compiled, the parsed code is available in the shared SQL area of the server and subsequent calls to the subprogram use this parsed code. This avoids reparsing for multiple users.
- Avoids PL/SQL parsing at run time by parsing at compile time
- Reduces the number of calls to the database and decreases network traffic by bundling commands.

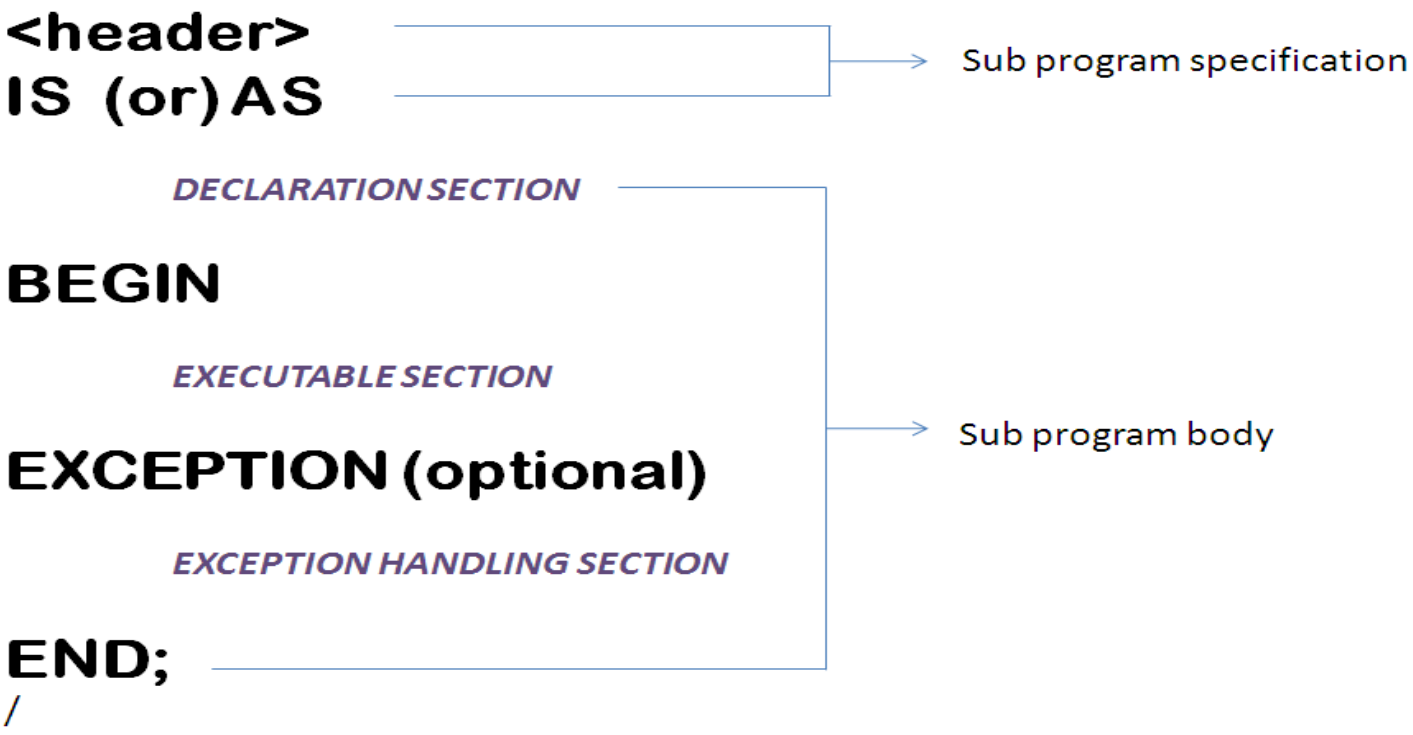
Copyright © 2013 Tech Mahindra. All rights reserved.

Improves code clarity

- Using appropriate identifier names to describe the action of the routines reduces the need for comments and enhances the clarity of the code.

Copyright © 2013 Tech Mahindra. All rights reserved.

Block structure for PL/SQL Subprograms



Copyright © 2013 Tech Mahindra. All rights reserved.

Sub-program Specification

- The header is relevant for named blocks only and determines the way that the program unit is called or invoked.

- The header determines:
  - The PL/SQL subprogram type, that is, either a procedure or a function
  - The name of the subprogram
  - The parameter list, if one exists
  - The RETURN clause, which applies only to functions
  - The IS or AS keyword is mandatory.

Copyright © 2013 Tech Mahindra. All rights reserved.

Sub-program Body

- The declaration section of the block between IS | AS and BEGIN. The keyword, DECLARE that is used to indicate the start of the declaration section in anonymous blocks is not used here.

- The executable section between the BEGIN and END keywords is mandatory, enclosing the body of actions to be performed. There must be at least one statement existing in this section. There should be at least one NULL; statement, which is considered an executable statement.

- The exception section between EXCEPTION and END is optional. This section traps predefined error conditions. In this section, you define actions to take if the specified error condition arises.

Copyright © 2013 Tech Mahindra. All rights reserved.

What is a Procedure?

- A procedure is a type of subprogram that performs an action.
- A procedure can be stored in the database, as a schema object, for repeated execution.
- **Definition of a Procedure**
  - A procedure is a named PL/SQL block that can accept parameters (sometimes referred to as arguments), and be invoked. Generally speaking, you use a procedure to perform an action. A procedure has a header, a declaration section, an executable section, and an optional exception-handling section.

  - A procedure can be compiled and stored in the database as a schema object. Procedures promote reusability and maintainability. When validated, they can be used in any number of applications. If the requirements change, only the procedure needs to be updated.

Copyright © 2013 Tech Mahindra. All rights reserved.

Creating Procedures

- **Syntax**:
  CREATE [OR REPLACE] PROCEDURE procedure_name
  [(parameter1 [mode1] datatype1,
  parameter2 [mode2] datatype2,. . .)]
  IS|AS
  PL/SQL Block;

Copyright © 2013 Tech Mahindra. All rights reserved.

Parameter description

Parameter	Description
Procedure_Name	Name of the procedure
Parameter	Name of PL/SQL variable whose value is passed to or populated by the calling environment, or both, depending on the mode being used
Mode	Type of argument IN (default) OUT IN OUT
Data type	Data type of argument – can be any SQL/ PLSQL data type. Can be of %TYPE, %ROWTYPE, or any scalar or composite data type. You can not specify size of the data type in the parameters.
PL/SQL block	Procedural body that defines the action performed by the procedure.

Copyright © 2013 Tech Mahindra. All rights reserved.

Formal versus Actual Parameters

- ***Formal parameters***: variables declared in the parameter list of a subprogram specification.
- Example:
  - **CREATE PROCEDURE raise_sal(p_id NUMBER, p_amount NUMBER)**
  - **...**
  - **END raise_sal;**
- ***Actual parameters***: variables or expressions referenced in the parameter list of a subprogram call.

- Example:
  - **raise_sal(v_id, 2000)**

Copyright © 2013 Tech Mahindra. All rights reserved.

Creating Procedures with Parameters

IN	OUT	IN OUT
Default Mode	Must be specified	Must be specified
Value is passed into subprogram	Returned to Calling Environment	Passed into subprogram; returned to calling environment.
Formal parameters acts as a constant	Uninitialized variable	Initialized variable
Actual parameter can be a literal, expression, constant, or initialized variable	Must be a variable	Must be a variable
Can be assigned a default value	Cannot be assigned a default value	Cannot be assigned a default value

Copyright © 2013 Tech Mahindra. All rights reserved.

Example

- Write a procedure to get the salary of an employee

```
CREATE OR REPLACE PROCEDURE GETSAL
(P_EMPNO EMP.EMPNO%TYPE, P_SAL OUT NUMBER)
IS
BEGIN
  SELECT SAL INTO P_SAL FROM EMP WHERE EMPNO = P_EMPNO;
EXCEPTION
   WHEN NO_DATA_FOUND THEN
     DBMS_OUTPUT.PUT_LINE('EMPLOYEE NOT FOUND…');
END;
```

```
Execution
VARIABLE G_SAL NUMBER
EXECUTE GETSAL(7521,:G_SAL);
PRINT G_SAL;
```

Copyright © 2013 Tech Mahindra. All rights reserved.

Example

- Procedure to get the name and joindate of an employee

```
CREATE OR REPLACE PROCEDURE GETSAL
(P_EMPNO EMP.EMPNO%TYPE, P_ENAME OUT VARCHAR2, P_HDATE OUT
DATE)
IS
BEGIN
  SELECT ENAME,HIREDATE INTO P_ENAME,P_HDATE FROM EMP WHERE
EMPNO = P_EMPNO;
EXCEPTION
   WHEN NO_DATA_FOUND THEN
     DBMS_OUTPUT.PUT_LINE('EMPLOYEE NOT FOUND…');
END;
```

```
Execution
VARIABLE G_ENAME VARCHAR2(20);
VARIABLE G_HDATE VARCHAR2(12);
EXECUTE GETSAL(7521,:G_ENAME,G_HDATE);
PRINT G_ENAME  G_HDATE;
```

Copyright © 2013 Tech Mahindra. All rights reserved.

Using IN OUT Parameters

- With an IN OUT parameter, you can pass values into a procedure and return a value to the calling environment. The value that is returned is the original, an unchanged value, or a new value set within the procedure.

- An IN OUT parameter acts as an initialized variable.

Copyright © 2013 Tech Mahindra. All rights reserved.

IN OUT Parameters example

- Create a procedure with an IN OUT parameter to accept a character string containing 10 digits and return a phone number formatted as (800) 633-0575.

```
CREATE OR REPLACE PROCEDURE format_phone (p_phone_no IN OUT VARCHAR2)
IS
BEGIN
   p_phone_no := '(' || SUBSTR(p_phone_no,1,3) || ')' || SUBSTR(p_phone_no,4,3) ||
   '-' || SUBSTR(p_phone_no,7);
END format_phone;
/
```

```
SQL> VARIABLE g_phone_no VARCHAR2(15)
SQL> BEGIN
            :g_phone_no := '8006330575';
        END;
/
SQL> PRINT g_phone_no
SQL> EXECUTE format_phone (:g_phone_no)
SQL> PRINT g_phone_no
```

Copyright © 2013 Tech Mahindra. All rights reserved.

Procedure to create a record

- ITEM_MAST table (ITEMNO NUMBER(3) PRIMARY KEY, ITEM_NAME VARCHAR2(20), STOCK NUMBER(3));
- CREATE SEQUENCE SQ_ITEM START WITH 101;

```
CREATE OR REPLACE PROCEDURE ADD_ROW(P_ITEM_NAME VARCHAR2, P_STOCK NUMBER)
 IS
 BEGIN
       INSERT INTO ITEM_MAST VALUES(SQ_ITEM.NEXTVAL, P_ITEM_NAME,P_STOCK);
       COMMIT;
 EXCEPTION
        WHEN DUP_VAL_ON_INDEX THEN
          DBMS_OUTPUT.PUT_LINE(SQLERRM);
  END;
```

```
EXECUTION
EXECUTE ADD_ROW('COKE',40);
EXECUTE ADD_ROW('PEN',50);
```

Copyright © 2013 Tech Mahindra. All rights reserved.

Methods for Passing Parameters

Positional : List actual parameters in the same order as formal parameters.
Named : List actual parameters in arbitrary order by associating each with its corresponding formal parameter.
Combination: List some of the actual parameters as positional and some as named.

Example

```
EXECUTE ADD_ROW('SHAMPOO',60);  -- POSITIONAL
EXECUTE ADD_ROW(P_ITEM_NAME=>'MOUSE',P_STOCK=>70);
                    -- Named
EXECUTE ADD_ROW(P_ITEM_NAME=>'ICE CREAM',30) -- COMBINATION
```

Copyright © 2013 Tech Mahindra. All rights reserved.

Methods for Passing Parameters

```
CREATE OR REPLACE PROCEDURE add_dept
   (p_name IN departments.department_name%TYPE DEFAULT 'unknown',
    p_loc IN departments.location_id%TYPE DEFAULT 1700)
IS
BEGIN
    INSERT INTO departments(department_id, department_name, location_id)
    VALUES (departments_seq.NEXTVAL, p_name, p_loc);
END add_dept;
/
```

```
BEGIN
   add_dept;
   --above is true because the formal parameters have
   --default values associated with them
   add_dept ('TRAINING', 2500);
   --above method call is using positional parameters
   add_dept ( p_loc => 2400, p_name =>'EDUCATION');
   --above method call is using named parameter passing
   add_dept ( p_loc => 1200) ;
   --above method call using named parameter passing
   --it works fine because the second parameter has
   --default value associated with it
END;
```

Copyright © 2013 Tech Mahindra. All rights reserved.

Exercise

- Define a package to maintain passbook
  - Steps to maintain passbook
    - 1. Generate transaction number
      - System generates number and no input is given by user
      - DEFINE procedure
    - 2. Getting the opening balance
      - Based on transaction number generated by Step 1 procedure, get the opening balance. Define procedure
    - 3. Input trntype and amount, handle suitable exceptions
      Calculate closing balance and store the data
      - There are 2 inputs, not getting any output.
      - Define a procedure.
- .

Copyright © 2013 Tech Mahindra. All rights reserved.

Solution

- Generate transaction number

```
CREATE OR REPLACE PROCEDURE GENERATE_TRAN(P_SNO OUT NUMBER)
AS
BEGIN
   SELECT NVL(MAX(SNO),0)+1 INTO P_SNO FROM PASSBOOK;
END;
```

- Getting the opening balance based on above transaction

```
CREATE OR REPLACE PROCEDURE GETOPBAL(P_SNO NUMBER, P_BAL OUT NUMBER)
IS
BEGIN
  IF P_SNO = 1 THEN
     P_BAL := 0;
ELSE
  SELECT BALANCE INTO P_BAL FROM PASSBOOK WHERE SNO = P_SNO-1;
END IF;
END;
```

Copyright © 2013 Tech Mahindra. All rights reserved.

Solution

```
CREATE OR REPLACE PROCEDURE PB_MAINTAIN(P_TRNTYPE CHAR, P_AMOUNT NUMBER)
IS
   V_SNO NUMBER;
   V_BAL  NUMBER;
BEGIN
     GENERATE_TRAN(V_SNO)  -- Calling procedure
     GETOPBAL(V_SNO,V_BAL);  -- Calling procedure
     IF UPPER(P_TRNTYPE) = 'D' THEN
          V_BAL := V_BAL + P_AMOUNT;
     ELSE
          V_BAL := V_BAL – P_AMOUNT;
    END IF;
INSERT INTO PASSBOOK VALUES(V_SNO,SYSDATE,P_TRNTYPE,P_AMOUNT,V_BAL);
COMMIT;
END;
```

Handle suitable exceptions in the above procedure

```
EXECUTE PB_MAINTAIN('D',60000);
EXECUTE PB_MAINTAIN(P_TRNTYPE=>'W',P_AMOUNT=>5000);
EXECUTE PB_MAINTAIN(P_TRNTYPE=>'W',6000);
```

Copyright © 2013 Tech Mahindra. All rights reserved.

Exercise

- Write a procedure to get the salary of an employee
- Write a procedure to get the name and salary of an employee
- Write a procedure to delete particular employee
- Write a procedure to encrypt and decrypt your password
- Write a procedure to get the total salary of all the managers
- Write a procedure to get the name and salary of the highest paid employee

Copyright © 2013 Tech Mahindra. All rights reserved.