CHAPTER-14
CREATING PROCEDURES

Procedures makes the program modular and each module performs a specific function or task.

Modular programming approach makes the program readable,manageable,reusable and reliable.

PL/SQL supports modules of many types such as :

- Anonymous Blocks
- Procedures
- Functions
- Packages

A **Module** or **Procedure** is a logical unit of work ie,a logically grouped set of SQL and PL/SQL statements that together perform a specific task.

Types of PL/SQL Procedures:

PL/SQL supports two types of Procedures:

1. **Local or Anonymous Procedure:**These are unnamed procedures.These procedures are not stored as a database object in an Oracle database.

2. **Stored Procedure:** These are named procedures.These can accept input parameters and pass values to output parameters.

Need of Procedures:

1. Procedures makes a program modular and serve to meet the specific requirement.

2. They make a bigger program broken down into smaller and manageable units.

3. They enhance performance of a program by saving time in network traffic as they do not need recompilation as their compiled form is stored in the database.

4. They enhance reusability as a procedure once written can be used again or reused.

5. They provide a better database security.

6. They use shared memory resources.

PL/SQL structure of a named block:

HEADER

IS

DECLARATION SECTION

BEGIN

EXCEPTION

END;

Therefore, there are four sections in the declaration of a stored procedure.

1. HEADER SECTION: Here the type of block whether procedure or Function or Package and its name and parameters are specified.
2. DECLARATION SECTION: Here local variables to this very block are specified.
3. EXECUTION SECTION: Here all the executable statements appear.
4. EXCEPTION SECTION: Here all the exceptions handling statements are specified.

The declaration and exception sections are optional in any PL/SQL block.

NOTE:THE PROCEDURE specification begins with the keyword PROCEDURE and ends with the procedure name or a parameter list. Parameter declarations are optional. Procedures that take no parameters are written without parentheses.

The PROCEDURE body begins with the keyword IS or AS and ends with the keyword END followed by an optional procedure name. The procedure body has three parts:a declarative part,an executable part,and an optional exception-handling part.

Example : A procedure showing all the four sections.

PARAMETER MODES:

The formal parameters of a procedure have following three major attributes.

1. Name of the Procedure.
2. MODE.(IN ,OUT ,IN OUT)
3. Data type.

The parameter modes define the behaviour of formal parameters.
The three modes are:

a. IN MODE: IN parameter lets user to pass values to the procedure being called inside.Inside the procedure,an IN parameter acts like a constant.Therefore,it can not be assigned a value.
b. OUT MODE: OUT parameter lets user to return the values to the caller of a procedure.ie,to the sub program ,which invokes the procedure.
c. IN OUT MODE: IN OUT parameter lets use pass initial values to the procedure being called nad return updated values to the caller subprogram.
   Inside the procedure,an IN OUT parameter acts like an initialized variable.

DIFFERENCE BETWEEN THE THREE MODES:

IN	OUT	IN OUT
The default.	Must be specified.	Must be specified.
Passes values to a procedure.	Returns values to the caller.	Passes initial values to a procedure;returns updated values to the caller.
Formal parameter acts like a constant.	Formal parameter acts like an un-initialized variable.	Formal parameter acts like an initialized variable.
Formal parameters cannot be assigned a value.	Formal parameter cannot be used in an expression; must be assigned a value.	Formal parameter should be assigned a value.
Actual parameter can be a constant, initialized variable,literal,or expression.	Actual parameter must be a variable.	Actual parameter must be a variable.

PARAMETER DECLARATION CONSTRAINTS:

The data type of a formal parameter can consist of any one of the following type of declarations:

- An unconstrained type name ,such as NUMBER or VARCHAR2.
- A type that is constrained using the %TYPE or %ROWTYPE attributes.

NOTE: Numerically constrained types such as NUMBER(2) or VARCHAR(20) are not allowed in a parameter list.

ASSIGNING DEFAULT VALUE TO PARAMETERS:

Parameters can also take default values.For this the DEFAULT keyword is used.

Ex: PROCEDURE stud_data(roll IN NUMBER DEFAULT 20) IS……

Or

PROCEDURE stud_data(roll IN NUMBER := 20) IS……

THE END LABEL:

Procedure name can be added after the keyword END.

EXAMPLE :

```
CREATE OR REPLACE PROCEDURE get_stud
(stud_id IN student.sid%TYPE,
stud_name OUT student.sname%TYPE,
stud_city OUT student.city%TYPE)
IS
BEGIN
SELECT sname,city INTo stud_name,stud_city FROM student WHERE sid=stud_id;
END get_stud;
```

FUNCTIONS

Functions are modules that carry out one specific job and return a value

Which in the declaration section is shown by keyword RETURN.

SYNTAX:

CREATE FUNCTION <func_name>(list of parameters)RETURN <return type>

AS….

BEGIN

--function body

END ;

NOTE: A function like a procedure receives arguments from the calling program.The difference is that a function is a part of an expression and returns a single value to the calling program for its use.

The return value of a FUNCTION must be assigned to a variable or used in an expression.

EXAMPLE:

```
Create or Replace FUNCTION factorial RETURN NUMBER IS
N NUMBER(5);
F NUMBER(5);
BEGIN
f  :=1;
n :=&n;
for i in 1..n
loop
f := f*i;
end loop;
return f;
end;
```


DROPPING STORED PROCEDURES:

DROP PROCEDURE <procedure_name>;

DROP FUNCTION <function_name>;

* * *