SHREE H. N. SHUKLA GROUP OF COLLEGES

CHAPTER-4

Advanced PL/SQL

- Creating & Using Procedure
- Functions
- Package
- Triggers
- Creating Objects
- PL/SQL Tables
- Nested Tables
- Varrays

SHREE H. N. SHUKLA GROUP OF COLLEGES

Q-1 Explain Procedure with Example.

Detail :-

- PL/SQL Procedure or Stored Procedure is a PL/SQL block that performs one or more specific tasks.
- It is just like procedures in other programming languages.
- Procedure is Block or Unit that stores group of data together.
- The procedure contains a **header and body:**
  - **Header :-** The Header contains the name of the procedure or the parameters that passed to procedure.
  - **Body :**- The Body contains a declaration section , Execution section and Exception section.

How to Pass Parameters in Procedure :

- There are three ways to pass parameters in procedure:
  1. **In Parameters :** The In parameter can be referenced by the value of parameter that cannot be overwritten by the procedure.

  2. **OUT Parameters :** The Out parameter cannot be referenced by the value of parameter that can be overwritten by the procedure.

  3. **INOUT Parameters :** The INOUT parameter can be referenced by the procedure or function where value of parameter can be overwritten.

Syntax :-

```
Create  or  replace procedure procedure_name [ Parameters ]
IS
        [declaration section]
BEGIN
        [Executable section]
```

```
EXCEPTION
        [Exception section]
END [procedure_name];
```

Example :-

```
Create  or  replace procedure p1 (id IN NUMBER  name IN VARCHAR2)
IS
Begin
        Insert into user values(id , name);
End;
/
```

Output :-

Procedure Created.

How to Call / Execute Procedure :

- To execute or call any procedure…Execute statement with procedure name can be used like :

Example :-

```
Execute  p1(101, 'snehal')
```

Output :-

PL/SQL Procedure Successfully Completed.

How to Drop Procedure :

- To Drop Procedure , The following Statement can be used:

Example :-

- Drop Procedure p1

Q-2 Explain Function with Example.

Detail :-

- PL/SQL Function is very similar to PL/SQL Procedure.
- The only difference between procedure and function is that , A function must always return a value.
- Function is Block that stores group of data together.
- The function contains a **header and body:**
  - **Header :-** The Header contains the name of the Function or the parameters that passed to Function.
  - **Body :**- The Body contains a declaration section , Execution section and Exception section.

How to Pass Parameters in Function :

- There are three ways to pass parameters in Function:

4. **In Parameters :** The In parameter can be referenced by the value of parameter that cannot be overwritten by the Function.

5. **OUT Parameters :** The Out parameter cannot be referenced by the value of parameter that can be overwritten by the Function.

6. **INOUT Parameters :** The INOUT parameter can be referenced by the function where value of parameter can be overwritten.

Syntax :-

```
Create  or  Replace Function Function _name [ Parameters ]
RETURN return_datatype
{IS | AS}

BEGIN
        <function_body>

END [function_name];
```

Example :-

```
Create  or  replace Function adder (a IN NUMBER , b IN NUMBER)
Return number
IS
N3  number(8);
Begin
        N3:=n1+n2;
Return n3
End;
/
```

Output :-

Function Created.

How to Call / Execute Function :

- To execute or call any Function…Following code can be used:

Example :-

```
Declare
        Ans number(3);
Begin
        Ans := adder(10,20);
        Dbms_output.put_line(Ans);
End;
/
```

Output :-

PL/SQL Procedure Successfully Completed.

How to Drop Function :

- To Drop Function , The following Statement can be used:

Example :-

- Drop Function adder

Q-3 Write note on Varray.

Detail :-

- Varray Stands for Variable – sized array.
- A Varray is single dimensional collection of elements with the same data type.
- A Varray always has a fixed number of elements.
- Varray allows you to store repeating attributes of record in single row.
- To declare Varray type , following syntax can be used:

Syntax :

Type type_name IS VARRAY (max_elements) of element_type [NOT NULL]

- In this declaration:
  - **Type_name** is the type of the Varray.
  - **Max_elements** is the maximum number of elements allowed in the varray.
  - **NOT NULL** specifies that the element of the varray of that type can not have NULL elements.
  - **Element_type** is the type of elements of the varray type's variable.

Example :

Type t _ name IS VARRAY (5) of VARCHAR2(20) NOT NULL;

- Once the varray is declared , it can be used in Table / Abstract Data Type like Following :

Create table master(name varchar2(20) , data t_name);

- Varray support two built- in methods :

1. COUNT :- It returns the number of elements that a varray currently contains , not including null values.
2. LIMIT :- It used for VARRAY to decide the maximum number of values allowed.If LIMIT is used on a nested table it will return a null.

Q-4 Write note on Nested Table.

Detail :-

- Nested table is like one-dimensional array with number of elements.
- But Nested table is differ from array , because the size of nested table can increase dynamically.
- Nested table is table within table.
- Nested tables stored in the database always.
- Nested table must be initialized with a built-in function called constructor.
- Unlike Varray , Nested tables has no limit on the number of entries per row.
- A Nested table is created using the following syntax :-

Syntax :-

```
TYPE   type_name IS TABLE OF element_type [NOT NULL];
Table_name   type_name
```

- A Nested table can be stored in database column.

Example:-

```
TYPE   salary  IS TABLE OF NUMBER NOT NULL;
Salary_list salary;
Name varchar2(20);
```

- While creating a table that includes nested table , you must specify the name of the table that will be used to store the nested table's data.
- Now ,you should create one of the table and you can insert records to that table with the help of Nested table.

Q-5 Write note on Nested Table.

Detail :-

- A PL/SQL table is a one-dimensional , unbounded collection of homogeneous elements , indexed by integers.
- It looks like an array / SQL table but it is not exactly the same.
- There is a difference between Array / SQL table & PL/SQL table.
- PL/SQL tables are composite data structures.
- PL/SQL has two composite datatypes : TABLE and RECORD.
- Objects of type TABLE are known as PL/SQL tables.
- PL/SQL table is not a part of SQL.We can not issue commands like INSERT/UPDATE/DELETE etc. on it.
-
- PL/SQL tables use a primary key to give you array like access to rows.
- The number of rows in a PL/SQL table can increase dynamically.
- The PL/SQL tables grows as new rows are added.
- PL/SQL tables can have one column and a primary key.
- PL/SQL tables can sometimes also referred to as an index by table.
- Rows in a PL/SQL table do not have to be contiguous.

Syntax :-

```
TYPE   type_name IS TABLE OF element_type index by <type>;
```

Example :-

```
TYPE   emp_table  IS  TABLE OF varchar2(10) index by binary_integer;
Var_of_table  emp_table;
Var_of_table(1) := “hello world”;
Var _of _table(2) := “good day”;
```

Row #	Emp_Name
100	Meet
225	Snehal
226	Vandana
300	JAY
340	SRP
220	KSP

Q-6 Write note on Package.

Detail :-

- A Package is an object , which holds other objects like procedure , functions , cursor etc. within it.
- It is a container object and allows related objects to be stored together.
- Package support mainly following components :
  - Package Specification (Package Header)
  - Package Body

Package Specification (Package Header):-

- It contains name of the package.
- It Contains declaration of the procedure ,function,variables cursors etc.
- It does not contains any code for procedure /functions.

Syntax :-

```
CREATE OR REPLACE PACKAGE Package_name
{AS|IS}

    Private _variable_declaration |
    Private _cursor _declaration |
    Function_specification
    Procedure_specification

END <Package Name>;
```

Package Body :-

- It Contains the definition of public objects that are declared in the specification.
- Package body can also have other objects , which are private to the package.
- If package header does not contain an procedure / function then package body is optional.

Syntax :-

CREATE OR REPLACE PACKAGE BODY Package_name
{AS|IS}

Private _variable_declaration |
Private _cursor _declaration |
Function_specification
Procedure_specification

END <Package Name>;

- The variables /constants declared in the package specification can be accessed by any procedure /function within the package.

How to Execute / Call Package :-

- To Execute / Call the package , we can use Execute statement like following:

Syntax :-

Execute <Package_name> . <Object name>

Example :-

Execute Package1.function1

Example :-

```
Create Or Replace Package Body Pkgemp IS
    Procedure updaterecord(no stdent.rno%type) IS
        BEGIN
            UPDATE student set age=23 where rno = no;
            IF SQL%FOUND THEN
                Dbms_output.put_line('updated');
            ELSE
                Dbms_output.put_line('Not updated');
            ENDIF
    END updaterecord;
```

Example :- (Calling)

Execute Pkgemp.updaterecord

Q-7 Write note on Trigger.

Detail :-

- Triggers are the programs that are executed automatically in response to a change in the database.
- Oracle allows special type of procedures that are automatically executed when the events like INSERT/UPDATE/DELETE occurs.
- These event procedures are called "DATABASE TRIGGERS".
- The events that cause triggers firing are:
  - DML Events
  - DDL Events
  - Database Events
- The DML event triggers can be statement or row triggers.
- The DML statement trigger gets fired before or after the triggering statement.
- You can define multiple triggers for single event and type.

TRIGGER PARTS :

- The Trigger can be divide into following parts:
  - **Triggering Event :-** The statement like INSERT UPDATE or DELETE that cause trigger to be fired is called triggering event.
  - **Trigger Restrictions :**- It is an option specified using WHEN clause.This option is available for triggers that are fired for each row.
  - **Trigger Code :**- It is the PL/SQL code.

TRIGGER TYPES :

- The Trigger have following types :
  - **Row Triggers :-**
    - This trigger is fired each time a row in the table is affected.
    - This type of trigger should be used when some action is required when any row of the table is affected.
  - **Statement Triggers :-**
    - This is default type of triggers.
    - This trigger will be fired once and it is independent of the no. of affected rows in table.

- Even if the none of the row is affected , statement trigger will be fired.

BEFORE V/S AFTER TRIGGER :

- Before trigger execute the trigger action before the triggering statements.
- After trigger executes the trigger action after the triggering statement is executed.
- It is possible to have both BEFORE & AFTER trigger for the same triggering statement.

Syntax :-

```
CREATE OR REPLACE TRIGGER <trigger name>
[BEFORE  / AFTER ]
[DELETE / INSERT / UPDATE | OF <Col1> , <Col2>…]
ON <Table / View name>

[FOR EACH ROW  [WHEN  <condition> ] ]
DECLARE
    <variable / constant Declaration>
BEGIN
    <PL/SQL statement body>;
EXCEPTION
    <Exception PL/SQL statement body>
END;
```

Example :-

```
CREATE OR REPLACE TRIGGER tri1 After delete on college
Declare
    X number;
Begin
    Select  count(*) into x from college;
End;
/

Trigger Created.
```

SQL > Delete from college where rno > 1;
Error at line 1:
Can not delete
Error during execution of trigger tri1

Q-8 How to create object in PL/SQL.

Detail :-

- In PL/SQL , the programming is based on object types.
- An object type can represent any real world entity.
- The object type can not be created at sub program level.
- Once the object type is defined , the same can be used in subprograms.
- The object type can be created using 'CREATE TYPE'.
- The type body can be created only after creating its object type.

Syntax :-

```
Create type <object _ type_name> AS OBJECT
(
    <Attribute_1> <data type>,
    .
    .
);
/
```

- Once the object type is created , it can be used in sub program declarative section to declare variable of that object type.
- Whenever any variable is declared in the subprogram as object type , at run time a new instance of the object type will be created.
- By the way , a single object type can store multiple values under different instances.
- The constructors are the implicit method of an object that can be referred with the same name as that of the object type.
- Whenever the object is referred for the first time , this constructor will be called implicitly.
- We can also initialize the objects using these constructor.

Example :-

```
Create type emp_object AS
OBJECT
(emp_no number,
 Emp_name varchar2(20),
Salary number)

Type Created.
```

Q-9 Give difference between Procedure and Function.

	Procedure	Function
1	A Procedure is a sub program that perform a specific task.	A Function is a subprogram that computes a value.
2	Procedure does and does not return the value.	Function must return atleast a single value.
3	Procedure do not need any RETURN statement.	Function must contains atleast one RETURN statement.
4	Procedure can execute / invoke as a PL/SQL statement.	Function can execute/invoke as a part of an expression.
5	Transactions are possible.	Transactions are not possible.