PL/SQL

PL/SQL

- Introduction to PL/SQL
- Data types in PL/SQL
- Control Structures in PL/SQL
- Functions in PL/SQL

PL/SQL

- Procedure
- Cursor
- Trigger
- Exception In PL/SQL

PL/SQL

Introduction to PL/SQL

- **PL/SQL is a combination of SQL along with the procedural features of programming languages.**

- **Properties of programming as well as the great interaction with database.**

Points to Ponder:-

PLSQL IS NOT A CASE SENSITIVE LANGUAGE.

PL/SQL

Comments in PL/SQL

The PL/SQL compiler ignores comments but you should not.

Single-line comments begin with a double hyphen (--)

Multiline comments begin with a slashasterisk (/*), end with an asterisk-

PL/SQL

Important PL/SQL Concepts

- **DECLARE :- if you want to declare a variable in plsql program then it takes place in declare section.**
- **BEGIN:- is used to start the working of program.**
- **END:- is used to terminate the begin.**

Points to Ponder:-

DELIMITER IS USED TO RUN (/)

What to do Previously

- SET SERVEROUTPUT ON :- is run after every time when you login in a session.

PL/SQL

To show OUTPUT on Your Screen

DBMS_OUTPUT.PUT_LINE command
for e.g. if sal=10 and you want to print it.

Then it looks like

```
dbms_output.put_line('the salary is ' ||sal);
```

PL/SQL

Value assign in Variable

```
 Declare

Num number(11);

Begin

Num:=5;

End;
/
```

PL/SQL

User’s Input for a Variable

```
 DECLARE
N NUMBER(11);

BEGIN
DBMS_OUTPUT.PUT_LINE('ENTER A NUMBER:-');
N:=&N;
DBMS_OUTPUT.PUT_LINE('THE VALUE IS'||N);

END;
/
```

PL/SQL

Sample program to print your ‘Hello World’

```
 BEGIN

DBMS_OUTPUT.PUT_LINE(‘Hello World');

END;

/
```

PL/SQL

Control Structures in PL/SQL

IF STATEMENT

Syntax:-

```
IF condition THEN
statement 1;
ELSE
statement 2;
END IF;
```

Points to Ponder:-

IF STATEMENT WORKS AS SIMILAR AS C OR C++

PL/SQL

Control Structures in PL/SQL

IF STATEMENT EXAMPLE

```
DECLARE
Age number(11);
Begin
Age:=&age;
If age>18 then
Dbms_output.put_line('You can vote');
Else
Dbms_output.put_line('You cannot vote');
End if;
End;
/
```

PL/SQL

Control Structures in PL/SQL

IF WITH SQL TABLE

```
Declare
A number(11);
Begin
Select sal into a from empcp where
    ename='MARTIN';
If a>1000 then
Update empcp set comm=comm+1000 where
    ename='MARTIN';
Else
Update empcp set comm=comm+500 where
    ename='MARTIN';End if;End;/
```

PL/SQL

Control Structures in PL/SQL

LOOPS IN PL/SQL

1)SIMPLE LOOP

2)WHILE LOOP

3)FOR LOOP

PL/SQL

Control Structures in PL/SQL

FOR LOOP

Print number from 1 to 10 using FOR loop

```
BEGIN
FOR i in [REVERSE] 1 ..10 Loop
Dbms_output.put_line(i);
END Loop;
End;
/
```

Points to Ponder:-

(For FOR LOOP has NO need to initialize variable i explicitly but it need in while)

PL/SQL

Control Structures in PL/SQL

WHILE LOOP

Print number from 1 to 10 using WHILE loop

```
Declare
i number(3):=0;
Begin
While i<10 loop
i:=i+1;
Dbms_output.put_line(i);
End loop;
End;
/
```

Control Structures in PL/SQL

SYNTAX OF SIMPLE LOOP

```
LOOP
Statement 1;
Statement 2;
Exit condition
End loop;
```

PL/SQL

Control Structures in PL/SQL

SIMPLE LOOP EXAMPLE

```
Declare
I number(2):=0;
Begin
Loop
dbms_output.put_line('Before Increment value of
    variable I:   ' || I);
I:=I+1;
dbms_output.put_line('After Increment value of
    variable I:  ' || I);
EXIT when (I>0);END loop;END;/
```

PL/SQL

Control Structures in PL/SQL

SYNTAX OF "CASE"

```
CASE (expression)
 WHEN <value1> THEN action_blockl;
 WHEN <value2> THEN action_block2;
 WHEN <value3> THEN action_block3;
 ELSE action_block_default;
END CASE;
```

PL/SQL

Control Structures in PL/SQL

EXAMPLE OF “CASE”

```
Declare
a NUMBER :=55;
b NUMBER :=5;
arth_operation VARCHAR2(20) :='MULTIPLY';
BEGIN
dbms_output.put_line('Program started.' );
CASE (arth_operation)
WHEN 'ADD' THEN dbms_output.put_line('Addition of the numbers are:'  || a+b );
WHEN 'SUBTRACT' THEN dbms_output.put_line('Subtraction of the numbers are: '||a-b );
WHEN 'MULTIPLY' THEN dbms_output.put_line('Multiplication of the numbers are: '|| a*b
);
WHEN 'DIVIDE' THEN dbms_output.put_line('Division of the numbers are:'|| a/b);
ELSE dbms_output.put_line('No operation action defined. Invalid operation');
END CASE;
dbms_output.put_line('Program completed.' );
END;
/
```

PL/SQL

Functions in PL/SQL

"IN" PARAMETER

- **This is similar to passing parameters in programming languages.**
- **We can pass values to the stored procedure and function through these parameters or variables.**
- **This type of parameter is a read only parameter.**

Points to Ponder:-

We can assign the value of IN type parameter to a variable or use it in a query, but we cannot change its value inside the procedure.

PL/SQL

Functions in PL/SQL

"IN" PARAMETER SYNTAX

CREATE [OR REPLACE] FUNCTION function_name (param_name1 IN data_type, param_name2 IN data_type..)

Here:-

(i)param_name1, param_name2... are unique parameter names.

(ii) data_type - defines the DataType of the variable.

(iii) IN - is optional, by default it is a IN type parameter.

Functions in PL/SQL

"OUT" PARAMETER

- The OUT parameters are used to send the OUTPUT from a procedure or a function.

- This is a write-only parameter i.e. we cannot pass values to OUT parameter while executing the stored procedure, but we can assign values to OUT parameter inside the stored procedure and the calling program can receive this output value.

Functions in PL/SQL

"OUT" PARAMETER SYNTAX

```
CREATE [OR REPLACE] PROCEDURE
    procedure_name
(param_name1 IN data_type, param_name2 IN
    data_type , param_name3 OUT data_type)
```

Here:-

(i) param_name1, param_name2 are unique parameter Names and can not be modified with in procedure.

(ii) param_name3 can be modified with in procedure.

(iii) data_type - defines the DataType of the variable.

Functions in PL/SQL

EXAMPLE ON “IN” AND “OUT”

```
CREATE OR REPLACE PROCEDURE procname_outparam (inparam1 in VARCHAR2,outParam2
      OUT VARCHAR2)
IS
BEGIN
outParam2 := 'Hello World OUT parameter I am changeable parameter' || inparam1;
END;
/
```

Run it:-

```
DECLARE
outParam2 VARCHAR2(100);
inparam1  VARCHAR2(100);
BEGIN
Procname_outparam ('i am IN parameter',outParam2);
DBMS_OUTPUT.PUT_LINE(outParam2);
END;
/
```

Functions in PL/SQL

“INOUT” PARAMETER

- The IN OUT parameter allows us to pass values into a procedure and get output values from the procedure.
- This parameter is used if the value of the IN parameter can be changed in the calling program.
- By using IN OUT parameter we can pass values into a parameter and return a value to the calling program using the same parameter. But this is possible only if the value passed to the procedure and output value have a same data type.

PL/SQL

Functions in PL/SQL

"IN OUT" PARAMETER SYNTAX

CREATE [OR REPLACE] PROCEDURE procedure_name

(param_name IN OUT data_type)

Here:-

(i) param_name are unique parameter Names and can not be modified with in procedure.

(ii) param_name3 can be modified with in procedure.

(iii) data_type - defines the DataType of the variable.

PL/SQL

Functions in PL/SQL

DEFINITION OF FUNCTION

Functions is a standalone PL/SQL subprogram. Like PL/SQL procedure, functions have a unique name by which it can be referred. These are stored as PL/SQL database objects. Below are some of the characteristics of functions.

- **Functions are a standalone block that is mainly used for calculation purpose.**
- **Function use RETURN keyword to return the value, and the data type of this is defined at the time of creation.**

PL/SQL

Functions in PL/SQL

What is Function

- **A Function should either return a value or raise the exception, i.e. return is mandatory in functions.**

Points to Ponder:-

* **Function with no DML statements can be directly called in SELECT query whereas the function with DML operation can only be called from other PL/SQL blocks.**

* **Function can also return the value through OUT parameters other than using RETURN.**

PL/SQL

Functions in PL/SQL

SYNTAX OF FUNCTION

```
CREATE OR REPLACE FUNCTION
<function_name>
(
< variable_name parameter IN/OUT <datatype>
)
RETURN <datatype>
[ IS | AS ]
<declaration_part>
BEGIN
<execution part>
RETURN (value/var);
END<function_name>;
```

सी-डैक
CDAC

PL/SQL

Functions in PL/SQL

EXAMPLE ON FUNCTION

```
create or replace function f_mul( x in int,y in int)
return int
as
z int;
begin
z:= x*y;
return(z);
End f_mul;
/
```

Points to Ponder:-

 * calling function from select query

```
                                 select  f_mul(12,10) from dual;
```

 * calling a function from a program

```
                                  Declare
                                  a int;b int;c int;
                                  Begin
                                  a:= &a;b:=&b;
                                  c:=f_mul(a,b);
                                  dbms_output.put_line('Result:' || c);End;/
```

Functions in PL/SQL

EXAMPLE ON FUNCTION

Points to Ponder:-

*** calling a function from procedure**

```
(I) create or replace procedure proc_mul(x int,y int,z out int)
 as
begin
z:=f_mul(x,y);
End;
 /
 (II) Declare
            a int;
            b int;
            c int;
      Begin
             a:=&a;
             b:=&b;
             proc_mul(a,b,c);
             dbms_output.put_line('value of a:' ||a || 'value of b:' ||b);
             dbms_output.put_line(‘Multiplication of  aXb  is :-'  || c);
             End;
              /
```

PL/SQL

Procedure in PL/SQL

DEFINITION OF PROCEDURE

A Procedure is a subprogram unit that consists of a group of PL/SQL statements. Each procedure in Oracle has its own unique name by which it can be referred. This subprogram unit is stored as a database object. Below are the characteristics of this subprogram unit.

- **Procedures are standalone blocks of a program that can be stored in the database.**
- **Call to these procedures can be made by referring to their name, to execute the PL/SQL statements.**

PL/SQL

Procedure in PL/SQL

DEFINITION OF PROCEDURE

- **It is mainly used to execute a process in PL/SQL.**
- **It can have nested blocks, or it can be defined and nested inside the other blocks or packages.**
- **The values can be passed into the procedure or fetched from the procedure through parameters.**
- **Procedure can have a RETURN statement to return the control to the calling block, but it cannot return any values through the RETURN statement.**
- **Procedures cannot be called directly from SELECT statements. They can be called from another block or through EXEC keyword.**

PL/SQL

Procedure in PL/SQL

SYNTAX OF PROCEDURE

```
CREATE [OR REPLACE ] PROCEDURE <name>[(argvarDatatype,
    argvarDatatype)]
IS/AS
<declaration>
BEGIN
Statement1
Statement2
-----------
-----------
EXCEPTION
Statement1
Statement2
----------
----------
END;/Note:-[] i.e OPTIONAL
```

PL/SQL

Procedure in PL/SQL

EXAMPLE ON PROCEDURE

```
CREATE OR REPLACE PROCEDURE proc_empcntno(vdesg emp.job%type)
AS
      empcntno int;
BEGIN
       select count(*) into empcntno from emp
       where job=vdesg;
       dbms_output.put_line('Entered Designation is:'||vdesg);
       dbms_output.put_line('NO. of Employees:'||empcntno);
  END ;
  /
```

PL/SQL

Procedure in PL/SQL

IMPORTANT POINT ABOUT PROCEDURE

* Executing Multiple Queries .
* Reduces no. of hits to DB.
* Improve DB performance and N/W performance.
* Enhancebility.
* Reusability.
* Modularity.

PL/SQL

Exception in PL/SQL

1) Exception

2)Exception Handling

3) Structure of Exception Handling.

4) Types of Exception Handling.

PL/SQL

Exception in PL/SQL

WHAT IS EXCEPTION

An error occurs during the program execution is called Exception in PL/SQL.

PL/SQL facilitates programmers to catch such conditions using exception block in the program and an appropriate action is taken against the error condition.

PL/SQL

Exception in PL/SQL

EXCEPTION HANDLING

PL/SQL provides a feature to handle the Exceptions which occur in a PL/SQL Block known as exception Handling. Using Exception Handling we can test the code and avoid it from exiting abruptly.

When an exception occurs a messages which explains its cause is received.

PL/SQL Exception message consists of three parts.

1) Type of Exception(Exception Handler)

2) An Error Code

3) A message

PL/SQL

Exception in PL/SQL

STRUCTURE OF HANDLING EXCEPTION

```
DECLARE
   Declaration section
 BEGIN
   Exception section
 EXCEPTION
 WHEN ex_name1 THEN
    -Error handling statements
 WHEN ex_name2 THEN
    -Error handling statements
 WHEN Others THEN
   -Error handling statements
END
```

Exception in PL/SQL

TYPES OF EXCEPTION

There are 3 types of Exceptions.

a) Named System Exceptions
b) Unnamed System Exceptions
c) User-defined Exceptions

PL/SQL

Exception in PL/SQL

NAMED SYSTEM EXCEPTIONS

System exceptions are automatically raised by Oracle, when a program violates a RDBMS rule. There are some system exceptions which are raised frequently, so they are pre-defined and given a name in Oracle which are known as Named System Exceptions.

सी डैक
CDAC

PL/SQL

Exception in PL/SQL

NAMED SYSTEM EXCEPTIONS

For example: NO_DATA_FOUND and ZERO_DIVIDE are called Named System exceptions.

Named system exceptions are:

1) Not Declared explicitly,

2) Raised implicitly when a predefined Oracle error occurs,

3) caught by referencing the standard name within an exception-handling routine.

PL/SQL

Exception in PL/SQL

NAMED SYSTEM EXCEPTIONS

Exception Name	Reason(Message)	Error Number
CURSOR_ALREADY_OPEN	When you open a cursor that is already open.	ORA-06511

PL/SQL

Exception in PL/SQL

NAMED SYSTEM EXCEPTIONS

Exception Name	Reason(Message)	Error Number
INVALID_CURSOR	When you perform an invalid operation on a cursor like closing a cursor, fetch data from a cursor that is not opened.	ORA-01001
NO_DATA_FOUND	When a SELECT...INTO clause does not return any row from a table.	ORA-01403

PL/SQL

Exception in PL/SQL

NAMED SYSTEM EXCEPTIONS

Exception Name	Reason(Message)	Error Number
TOO_MANY_ROWS	When you SELECT or fetch more than one row into a record or variable.	ORA-01422
ZERO_DIVIDE	When you attempt to divide a number by zero.	ORA-01476.

PL/SQL

Exception in PL/SQL

PROGRAM ON NAMED SYSTEM EXCEPTION

```
Create or Replace Procedure proc_excep(vdesg emp.job%type)
        is
        vename varchar2(20);
        Begin
            select ename into vename from emp
            where job =vdesg;
         dbms_output.put_line('Name:-' || vename);
        Exception
           when NO_DATA_FOUND then
             dbms_output.put_line('No Employee existed with given job');
           when TOO_MANY_ROWS then
             dbms_output.put_line('Multiple employees existed ,use explicit
    cursor');
        End proc_excep;
        /
```

PL/SQL

Exception in PL/SQL

USER DEFINED EXCEPTIONS

Apart from system exceptions we can explicitly define exceptions based on business rules. These are known as user-defined exceptions.

Steps to be followed to use user-defined exceptions:

- **They should be explicitly declared in the declaration section.**
- **They should be explicitly raised in the Execution Section.**
- **They should be handled by referencing the user-defined exception name in the exception section.**

PL/SQL

Exception in PL/SQL

SYNTAX OF USER DEFINED EXCEPTION

```
Declare
var_name Exception
Begin
Statement1
Statement2
Raise <var_name>;
Exception
 When <var_name> then
 Statement1
 Statement2
End;
/
```

PL/SQL

Exception in PL/SQL

EXAMPLE ON USER DEFINED EXCEPTION

```
Create or Replace Procedure proc_comm (veid int)
               is
               vcomm int;
               c_miss Exception;
               Begin
                   select comm into vcomm from emp where empno=veid;
                 if vcomm IS NOT NULL then
                 dbms_output.put_line('Comm of' || veid || 'is' || vcomm);
                 else
                 RAISE c_miss;
                 end if;
                 Exception
                 when NO_DATA_FOUND then
                 dbms_output.put_line('EmpId Not Existed');
                 when TOO_MANY_ROWS then
                 dbms_output.put_line('Duplicate EmpId');
                 when c_miss then
                 dbms_output.put_line('Comm is Not Existed');
                 End proc_comm;
                 /
```

PL/SQL

Exception in PL/SQL

UNNAMED EXCEPTIONS

- **It is also known as PRAGMA EXCEPTION_INIT.**
- **It is the way to associate user defined exception with oracle predefined error.**
- **If the oracle predefined error is not having name then we can assign name to that error using PRAGMA EXCEPTION_INIT.**

PL/SQL

Exception in PL/SQL

SYNTAX OF UNNAMED EXCEPTION

```
DECLARE
        user_define_exception_name EXCEPTION;
        PRAGMA EXCEPTION_INIT(user_define_exception_name,-
  error_number);
      BEGIN
        statement(s);
        IF condition THEN
        RAISE user_define_exception_name;
        END IF;
      EXCEPTION
        WHEN user_define_exception_name THEN
        User defined statement (action) will be taken;
      END;
      /
```

Points to Ponder:-

* **exception_name and error_number define on yourself, where exception_name is character string up to 2048 bytes support and error_number is a negative integer range from -20000 to -20999.**

PL/SQL

Exception in PL/SQL

PROGRAM ON UNNAMED EXCEPTION

```
DECLARE
              myex EXCEPTION;
              PRAGMA EXCEPTION_INIT(myex,-20015);
              n NUMBER := &n;
           BEGIN
              FOR i IN 1..n LOOP
              dbms_output.put_line(i);
              IF i=n THEN
              RAISE myex;
              END IF;
              END LOOP;
           EXCEPTION
              WHEN myex THEN
              dbms_output.put_line('loop finish');
           END;
           /
```

PL/SQL

Cursor in PL/SQL

DEFINITION OF CURSOR

- **A cursor is a temporary work area created in the system memory when a SQL statement is executed. A cursor contains information on a select statement and the rows of data accessed by it.**
- **This temporary work area is used to store the data retrieved from the database, and manipulate this data. A cursor can hold more than one row, but can process only one row at a time. The set of rows the cursor holds is called the active set.**

PL/SQL

Cursor in PL/SQL

TYPES OF CURSOR

There are two types of cursors in PL/SQL:

- **Implicit cursors**
- **Explicit cursors**

PL/SQL

Cursor in PL/SQL

IMPLICIT CURSOR

- **Managed by Oracle Engine**
- **Allocation and deallocation is done by Oracle Engine.**
- **Used by DML,DQL.**

PL/SQL

Cursor in PL/SQL

PROPERTIES OF IMPLICIT CURSOR

- **SQL%ISOPEN**
- **SQL%FOUND**
- **SQL%NOTFOUND**
- **SQL%ROWCOUNT**

PL/SQL

Cursor in PL/SQL

PROGRAM ON IMPLICIT CURSOR

```
Create or Replace procedure proc_sal_update(vjob in emp.job%type,incr in int)
          is
          Begin
              update empcp
              set sal=sal+((sal*incr)/100)
              where job=vjob;
              if (sql%found) then
              dbms_output.put_line('Updation is Successful');
              dbms_output.put_line('Number of '||vjob||'s updated:'
              ||SQL%ROWCOUNT);
           else
               dbms_output.put_line('Updation Failed');
               dbms_output.put_line('Number of '||vjob||'s updated:'
               ||SQL%ROWCOUNT);
           end if;
           End proc_sal_update;
           /
```

PL/SQL

Cursor in PL/SQL

EXPLICIT CURSOR

- **Managed by User**
- **Allocation and deallocation is done by User.**
- **To display multiple records using subprogram/procedure.**

Its has four steps to define it;-

I)Declare Cursor

II)Open Cursor

III)Fetch data from cursor

IV)Close Cursor

PL/SQL

Cursor in PL/SQL

SYNTAX OF EXPLICIT CURSOR

I)Declare Cursor

```
  CURSOR <cursor_name> IS select
   column_name...... from <table_name>
        where condition;
```

II)Open Cursor

```
   OPEN <cursor_name>;
```

III)fetch data from Cursor

```
   FETCH <cursor_name> INTO <var>
```

IV)Close Cursor

```
CLOSE <cursor_name>;
```

PL/SQL

Cursor in PL/SQL

PROPERTIES OF EXPLICIT CURSOR

- **<cursor_name>%ISOPEN**
- **<cursor_name>%FOUND**
- **<cursor_name>%NOTFOUND**
- **<cursor_name>%ROWCOUNT**

Cursor in PL/SQL

PROGRAM ON EXPLICIT CURSOR

```
 DECLARE
   CURSOR my_cursor IS SELECT sal + NVL(comm, 0) wages, ename
   FROM emp;
   my_rec  my_cursor%ROWTYPE;
BEGIN
   OPEN my_cursor;
   LOOP
      FETCH my_cursor INTO my_rec;
      EXIT WHEN my_cursor%NOTFOUND;
      IF my_rec.wages > 2000 THEN
     dbms_output.put_line(my_rec.wages||'  '||my_rec.ename);
      END IF;
   END LOOP;
   CLOSE my_cursor;
END;
/
```

PL/SQL

Trigger in PL/SQL

DEFINITION

Triggers are stored programs, which are automatically executed or fired when some events occur. Triggers are, in fact, written to be executed in response to any of the following events –

***A database manipulation (DML) statement (DELETE, INSERT, or UPDATE)**

***A database definition (DDL) statement (CREATE, ALTER, or DROP).**

***A database operation (SERVERERROR, LOGON, LOGOFF, STARTUP, or SHUTDOWN).**

PL/SQL

Trigger in PL/SQL

SYNTAX OF TRIGGER

```
CREATE [OR REPLACE ] TRIGGER trigger_name
{BEFORE | AFTER | }
{INSERT [OR] | UPDATE [OR] | DELETE}
[OF col_name]
ON table_name
[REFERENCING OLD AS o NEW AS n]
[FOR EACH ROW]
WHEN (condition)
DECLARE
   Declaration-statements
BEGIN
   Executable-statements
EXCEPTION
   Exception-handling-statements
END
```

सी डैक
CDAC

PL/SQL

Trigger in PL/SQL

PROGRAM ON DML CURSOR

```
CREATE OR REPLACE TRIGGER display_salary_changes1
BEFORE DELETE OR INSERT OR UPDATE ON empcp1
FOR EACH ROW
WHEN (old.deptno > 0)
DECLARE
   sal_diff number;
BEGIN
   sal_diff := :NEW.sal  - :OLD.sal;
   dbms_output.put_line('Old salary: ' || :OLD.sal);
   dbms_output.put_line('New salary: ' || :NEW.sal);
   dbms_output.put_line('Salary difference: ' || sal_diff);
END;
/
```

Thank you!