VINSYS

DELiVERiNGSKiLLS
DRiViNGSUCCESS

www.VINSYS.COM

VINSYS

- Introduction To PL/SQL

About PL/SQL

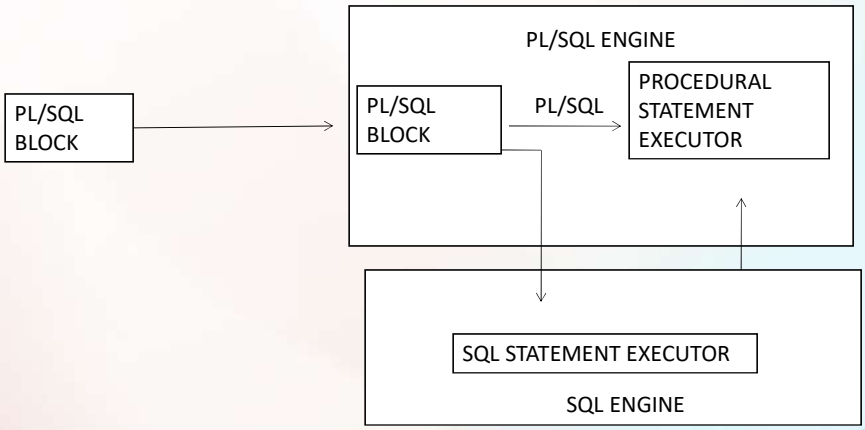
PL/SQL is the procedural extension to SQL with design features of programming languages.

PL/SQL is a combination of SQL along with the procedural features of programming languages. It is Oracle corporation Product

Need for PL/SQL

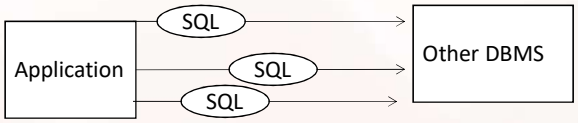
- Processing bulk of data
- Writing program modules
- Storing the data temporarily
- Provision of conditional logic
- Looping the data
- Handling errors arising in execution of query
- Complex Business Logic can not be handled(Trigger)
- User defined functions are possible within PL/SQL

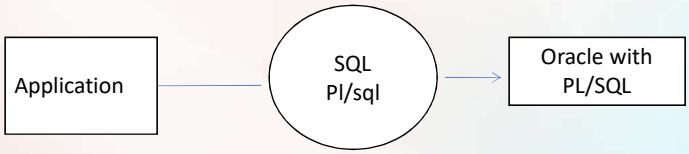
PL/SQL Environment



Benefits of PL/SQL

- Improved performance





Benefits of PL/SQL

- Modularize program development
  - Every unit of pl/sql comprises of one or more blocks.
  - Modularization help keep logically related data under inside one block
  - Nest subblocks inside larger blocks .
  - Break-down a complex problem into a set of manageable modules
  - Named blocks stored in Database can be accessible to any application that interact with an oracle server.

Benefits of PL/SQL

- PL/SQL is portable
  - PL/SQL programs can run anywhere oracle server runs
- You can Declare variables.
- You use Control Structures
- You can handle errors
  - Oracle server errors
  - User Defined Error

Benefits of PL/SQL

- Improved data security
- Improved performance
  - No need of parsing a code for different users
  - No need of parsing a code multiple times-each time the module is executed
  - Reduced number of calls to database and decrease network traffic by bundling code.
- Improved code clarity

PL/SQL block structure

DECLARE(Optional)

Variables,cursors,user-defined exceptions,PL/SQL records,pl/Sql Tables etc.

BEGIN(Mandatory)

SQL statements

PL/SQL statements

EXCEPTION(Optional)

Actions to perform when any error takes place

END;(Mandatory)

Block Types

Anonymous	Named Block
[DECLARE]	Procedure
BEGIN	Function
--statements	
END;	

Features Of Anonymous Block

- Session Specific
- Compiled before execution
- Anonymous blocks can't accept parameters
- Decentralized Approach - If stored, they are stored as 'script files' on the client machine
- You can not invoke this block from other blocks/subprograms
- Real life application-Testing-Debugging

Use of variables

- Variables can be used for:
  - Temporary storage of data
  - Manipulation of stored values
  - Reusability

- PL/SQL does not have input or output capability of its own.
- You can reference substitution variables within a PL/SQL block with a preceding ampersand.
- *iSQL*Plus host (or "bind") variables can be used* to pass run time values out of the PL/SQL block back to the *iSQL*Plus environment.*

Declaring PL/SQL Variables

Syntax-

identifier [CONSTANT] datatype [NOT NULL]
[:= | DEFAULT expr];

Examples –

```
DECLARE
    v_hiredate DATE;
    v_deptno NUMBER(2) NOT NULL := 10;
    v_location VARCHAR2(13) := 'Atlanta';
    c_comm CONSTANT NUMBER := 1400;
```

Guidelines for Using Variables

- Follow naming conventions.
- Initialize variables designated as NOT NULL and CONSTANT.
- Declare one identifier per line.
- Initialize identifiers by using the assignment operator (:=) or the DEFAULT reserved word.

  identifier := expr;
- Variables are case 'insensitive'.

```
BEGIN
   DBMS_OUTPUT.PUT_LINE('HELLO WORLD');
END;
```

Example of writing PL/SQL Block

```
Declare
 x varchar2(25);
Begin
 x:='Hello';
 dbms_output.put_line(x);
End;
```

Using sql inside the anonymous block


```
DECLARE
   v_no number(4);
  v_name varchar2(25);
   v_salary number(4);
BEGIN
  Select empno,ename,SAL
  into v_no,v_name,v_salary
   from emp
   where empno=7369;
END;
/
```

Caution

- Use variables to store values of columns returned by the query
- Number of columns, Data Type, sequence of column must match with variables also
- Into clause is compulsory for sql inside a pl/sql block.
- Sql should return only one row. Use **where clause** in query

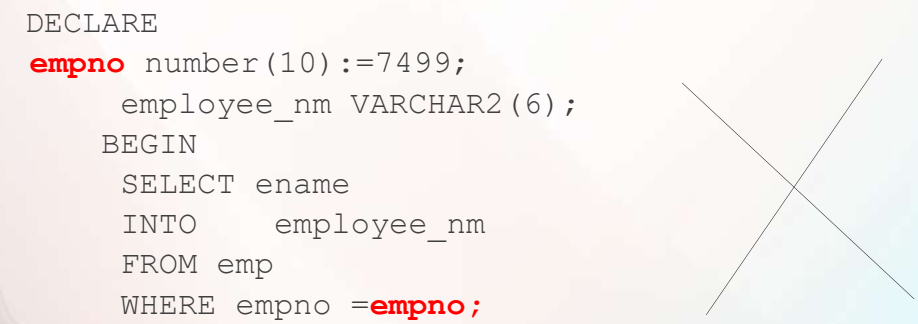
Variable Name

- Two variables can have the same name, provided they are in different blocks.
- The names of the variable must begin with a Letter, variable should not be more than 30 character long remaining characters can contain letters,special characters & number.
- The variable name (identifier) should not be the same as the name of table columns used in the block.

Variable Name


```
DECLARE
 empno NUMBER(6);
BEGIN
 SELECT empno
 INTO    empno
 FROM employees
 WHERE last_name = 'Kochhar';
END;
/
```

Use of proper Variable Names



```
DECLARE
empno number(10):=7499;
     employee_nm VARCHAR2(6);
    BEGIN
     SELECT ename
     INTO    employee_nm
     FROM emp
     WHERE empno =empno;
     DBMS_OUTPUT.PUT_LINE(EMPLOYEE_NM);
    END;
/
```

Types of variables

- PL/SQL variables:
  - **Scalar**- single value, No internal-component
  - **Composite-** can have internal components that can be manipulated indivisually
  - **LOB (large objects)**- Can store blocks of unstructured data (text , images , video clips) upto 4 GB in size.
- Non-PL/SQL variables: Bind and host variables

Scalar Datatype

- Char
- Varchar
- Long
- Longraw
- Number
- Binary_integer
- Pls_integer
- Boolean
- Date
- Timestamp

Use of Substitution Variables


```
Declare
 x varchar2(25);
 sal number(6):=&salary;
Begin
 x:='&x';
    dbms_output.put_line(x);
  dbms_output.put_line(' Your salary is '||sal);
End;
```

The %TYPE Attribute

- Declare a variable according to:
  - A database column definition
  - Another previously declared variable
- Prefix %TYPE with:
  - The database table and column
  - The previously declared variable name

The %TYPE Attribute

- SYNTAX:

identifier Table.column_name%TYPE;

- *EXAMPLES:*

```
v_name employees.last_name%TYPE;
v_balance NUMBER(7,2);
v_min_balance v_balance%TYPE := 10;
```

Example of %TYPE

```
DECLARE
  v_no emp.empno%type:=7649;
  v_nm varchar2(30);
BEGIN
  select ename
  into v_nm
  from emp
  where empno=v_no;
  DBMS_OUTPUT.PUT_LINE('Employee_id '||v_no||' is
  '||v_nm);
END;
```

Example Of Bind Variable

- To reference a bind variable in PL/SQL, you must prefix its name with a colon (:)
- Example -:

```
VARIABLE g_sal NUMBER
  BEGIN
  SELECT sal
  INTO :g_sal
  FROM emp
  WHERE empno= 7649;
END;
/
print g_sal
```

Nested blocks

- PL/SQL blocks can be nested wherever an executable statement is allowed.
- A nested block becomes a statement.
- An exception section can contain nested blocks.
- The scope of an identifier is that region of a program unit (block, subprogram, or package) from which you can reference the identifier.

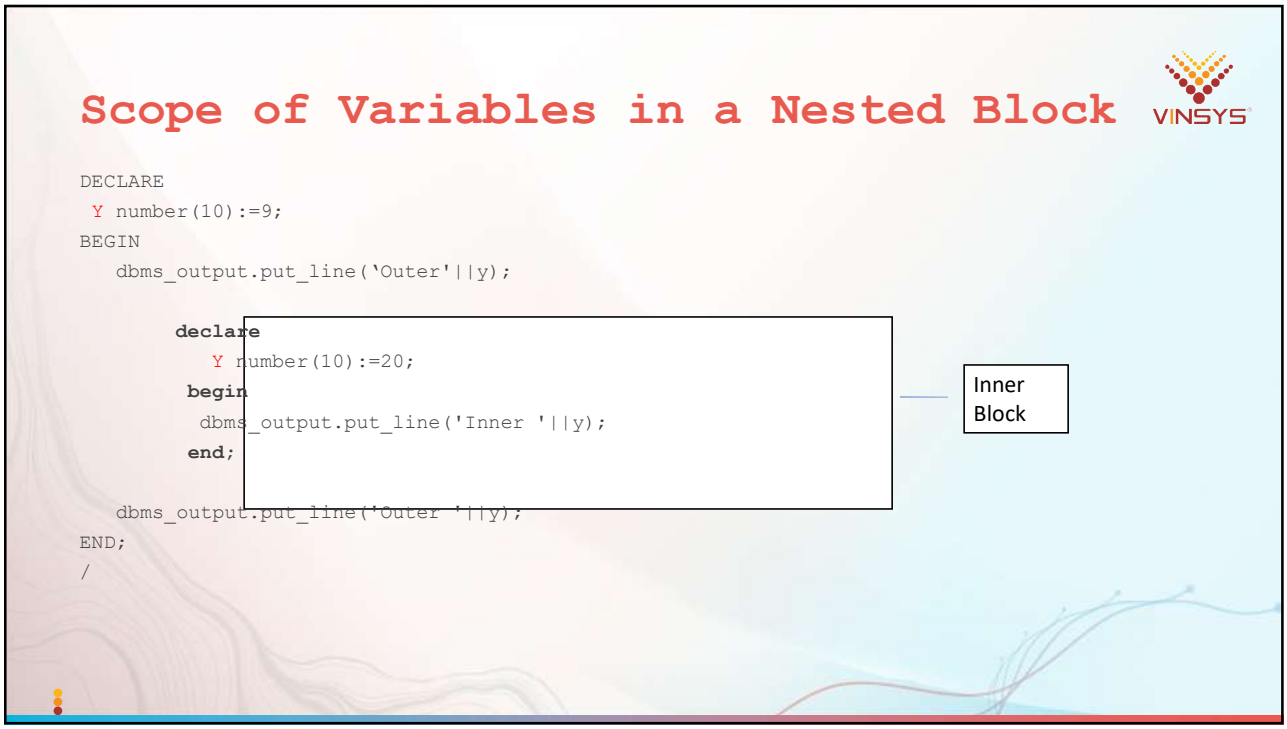
Nested blocks


```
DECLARE
  x number number(9);
BEGIN
  statment1;
  statment2;
       DECLARE
              y number(9);
              BEGIN
              y:=x;
       END;
  statment1;
  statment2;
END;
```

SCOPE OF 'x'

SCOPE OF 'y'

Scope of Variables in a Nested Block



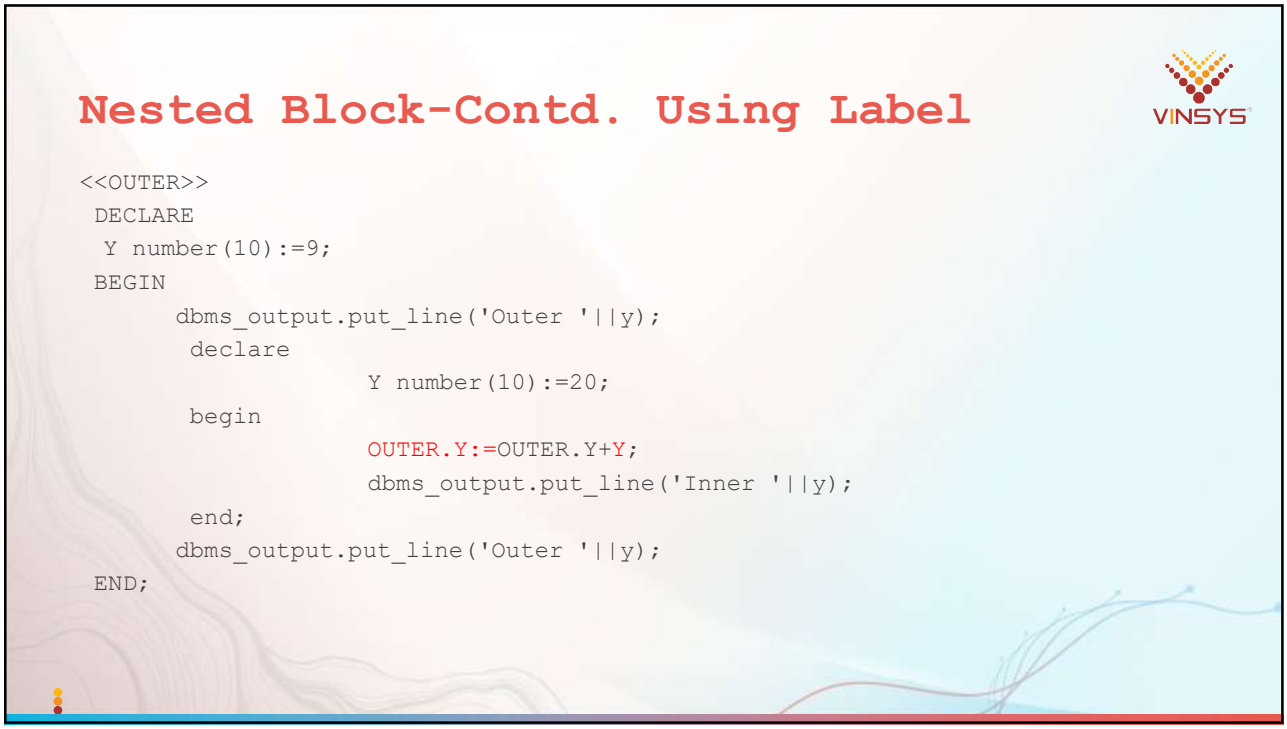
```
DECLARE
 Y number(10):=9;
BEGIN
   dbms_output.put_line('Outer'||y);

        declare
           Y number(10):=20;
         begin
          dbms_output.put_line('Inner '||y);
         end;

   dbms_output.put_line('Outer '||y);
END;
/
```

Inner Block

Nested Block-Contd. Using Label



```
<<OUTER>>
 DECLARE
  Y number(10):=9;
 BEGIN
       dbms_output.put_line('Outer '||y);
        declare
                    Y number(10):=20;
        begin
                    OUTER.Y:=OUTER.Y+Y;
                    dbms_output.put_line('Inner '||y);
        end;
       dbms_output.put_line('Outer '||y);
 END;
```

Same Variable In Nested Block


```
<<OUTER>>
DECLARE
 BALANCE NUMBER(9);
BEGIN
 SELECT SAL
 INTO BALANCE
 FROM EMP
 WHERE EMPNO=7499;
 BALANCE:=BALANCE+100;
           DECLARE
                       VCOMM NUMBER(9);
                      BALANCE NUMBER(9);
           BEGIN
                      SELECT COMM INTO VCOMM FROM EMP WHERE EMPNO=7499;
                      OUTER.BALANCE:=OUTER.BALANCE+VCOMM;
           END;
 DBMS_OUTPUT.PUT_LINE(BALANCE);
END;
/
```

SQL Statements Through PL/SQL

- Data Manipulation language –Insert,Update & Delete,Merge are possible through PL/Sql
- Transaction Control Language-Commit ,Rollback are possible through PL/Sql
- DDL & DCL are not directly possible through PL/Sql

Inserting Data

```
DECLARE
 V_no emp.empno%type:=&V_no;
BEGIN
 insert into emp(empno)
 values(V_no);
END;
/
```

Update

```
Example
DECLARE
 v_sal_increase emp.sal%TYPE := 800;
BEGIN
    UPDATE emp
    SET sal = sal + v_sal_increase
    WHERE job = 'CLERK';
END;
/
```

Delete

Delete rows that belong to department 10 from the EMPLOYEES table

Example:

```
DECLARE
 v_deptno emp.deptno%TYPE := 10;
BEGIN
 DELETE FROM emp
 *WHERE deptno = v_deptno;
END;
/
```

SQL Cursor

- A cursor is a private SQL work area.
- There are two types of cursors:
  - Implicit cursors
  - Explicit cursors
- The Oracle server uses implicit cursors to parse and execute your SQL statements.
- Explicit cursors are explicitly declared by the programmer.

SQL Cursor Attributes

Using SQL cursor attributes, you can test the outcome of your SQL statements

SQL%ROWCOUNT- -Number of rows affected by the most recent SQL statement (an integer value)

SQL%FOUND--- Boolean attribute that evaluates to TRUE if the most recent SQL statement affects one or more rows

SQL Cursor Attributes

SQL%NOTFOUND--- Boolean attribute that evaluates to TRUE if the most recent SQL statement does not affect any rows

SQL%ISOPEN--- Always evaluates to FALSE because PL/SQL closes implicit cursors immediately after they are executed

SQL Cursor Attributes


```
VARIABLE rows_deleted VARCHAR2(30)
DECLARE
 v_empno emp.empno%TYPE := 176;
BEGIN
 DELETE FROM emp
 WHERE empno = v_empno;
 :rows_deleted := (SQL%ROWCOUNT ||' row deleted.');
END;
/
PRINT rows_deleted
```

Conditional Statements

IF CONDITION

You can change the logical execution of statements using conditional IF statements and loop control structures.

- Conditional IF statements:
  - IF-THEN-END IF
  - IF-THEN-ELSE-END IF
  - IF-THEN-ELSIF-END IF

IF Statement

- It allows pl/sql statements to execute depending upon the condition value i.e TRUE/FALSE OR NULL.

```
…..
IF v_ename ='SMITH' AND salary> 1000 THEN
   V_deptno:=50;
 End If;
 ……
```

IF Statement


```
If sal>1000 then
   Bonus=500;
Else
   Bonus=300;
End if;
```

If Statement


```
If marks >=75 then
 Grade='A';
elsif  marks >=60 then
 Grade='B';
else
 Grade='C';
End If;
```

NULL Values With Conditions

When working with nulls, you can avoid some common mistakes by keeping in mind the following rules:

- Simple comparisons involving nulls always yield NULL.
- Applying the logical operator NOT to a null yields NULL.
- In conditional control statements, if the condition yields NULL, its associated sequence of statements is not executed.

Iterative Loop

- Loops repeat a statement or sequence of statements multiple times.
- There are three loop types:
  - Basic loop
  - FOR loop
  - WHILE loop

Basic loop

Syntax:

'condition' in the following example may be variable or expression(TRUE/FALSE/NULL)

```
LOOP
 statement1;
 . . .
 EXIT [WHEN condition];
END LOOP;
```

Basic Example of Basic ..Loop


```
declare
 i number(9):=1;
begin
 loop
     dbms_output.put_line(i);
     i:=i+1;
     exit when i>10;
 end loop;
end;
/
```

While loop

Syntax

```
WHILE condition LOOP
 statement1;
 statement2;
 . . .
END LOOP;
```

Example of While loop


```
declare
 z number(10):=10;
begin
 while z>1 loop
 dbms_output.put_line(z);
 z:=z-1;
 end loop;
 end;
/
```

'For' Loop

Syntax

```
FOR counter IN [REVERSE]
 lower_bound..upper_bound LOOP
 statement1;
 statement2;
 . . .
END LOOP;
```

FOR Loop

- Use a FOR loop to shortcut the test for the number of iterations.
- Do not declare the counter; it is declared implicitly.
- 'lower_bound .. upper_bound' is required syntax.

Example of 'FOR' Loop

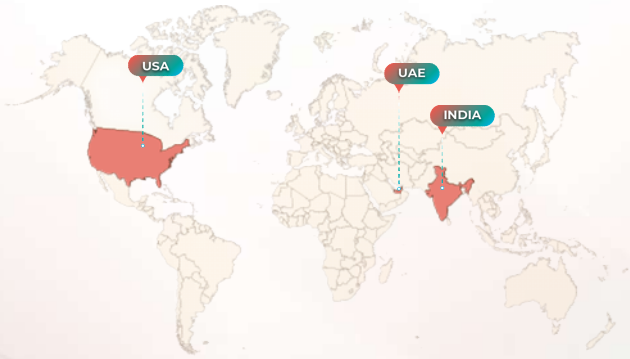

```
Begin
 for i in 1..10 loop
    dbms_output.put_line(i);
 end loop;
end;
/
```

The loop includes values of lower bound as well as values of upper bound

'FOR' loop

- Counter within this loop need not be explicitly declared.
- Counter variable in 'FOR' Loop can not change it's value explicitly & is incremented by only '1'
- Value of counter is not available outside the loop

Physical Presence



www.vinsys.com

enquiry@vinsys.com

Shivaji Niketan, Tejas Society, Behind Kothrud Bus Stand, Near Mantri Park, Kothrud, Pune - 411029.

304, City Tower 2, Near Crown Plaza, Sheikh Zayed Road. Dubai, UAE.
PO.Box - 213279

132 West 31st Street, First Floor, New York, 10001, USA