Introduction to PL/SQL

SURJEET KAUR
DEPT OF MATHEMATICS
SIES COLLEGE OF ARTS, SCIENCE AND COMMERCE

Outline

- Fundamentals of PL/SQL
- PL/SQL Data Types
- PL/SQL Control Structures
  - Conditional controls
  - Iterative controls

Fundamentals - Introduction

- Procedural Language extension of SQL
- Developed by Oracle Corporation

Fundamentals - Advantages

- Dependant execution of the sql statements
- Procedural language capability
- Changes as per the user inputs
- Better performance
- Error handling

Fundamentals – Block Structure

Declare …. (Optional)

- Variables
- Cursors
- User Defined Exceptions

Begin …. (Mandatory)

- SQL Statements
- PL/SQL Statements

Exception …. (Optional)

- Actions To Perform When Errors Occur

End …. (Mandatory)

PL/SQL Block


```
DECLARE
   a number:=3.52;
BEGIN
   a:=a+4;
   dbms_output.put_line(a);
END;
/
Set serveroutput on;
```

OUTPUT

7.52

Fundamentals - Variables

- Declaring variables –
  - Syntax –
    - *variable_name datatype [:= value];*
  - Example –
    - *salary number(6,2);*
    - *dept varchar2(10):= “HR Dept”;*

Fundamentals - Variables

- Assigning values to variables –
  - Directly assign values
  - Assign values to variables using a SELECT.. INTO statement
  - Accept from the user

Fundamentals - Variables

- Directly assign values -
  - Syntax –
    - *variable_name:= value;*
  - Example –
    - *salary := 10000;*

Fundamentals - Variables

- Assign values to variables using a SELECT.. INTO statement
  - Syntax –
    - *SELECT column_name INTO variable_name FROM table_name [WHERE condition];*
  - Example –
    - *SELECT empsalary INTO salary FROM employee WHERE emp_id = 'E0001';*

Fundamentals-variables

Var_name:=&var_name;

Fundamentals - Constants

- **Declaring constants –**
  - Syntax –
    - *constant_name CONSTANT datatype := VALUE;*
  - Example –
    - *salary_raise CONSTANT number (3) := 5000;*

Example


```
Create table salesman
(sid number,
Name varchar2(10),
sales number,
Commision number);
/
```

contd…


```
Declare
    ssid number;
    Sname varchar2(10);
    ssales number;
    Scomm number;
Begin
    Ssid:=&ssid;
    Sname:=&sname;
    Ssales:=&ssales;
    Scomm:=(ssales-100)*100;
    Insert into salesman values(ssid,sname,ssales,scomm);
End;
/
```

Data types - datetime

- **Syntax –**
  - variable_name date;

- **Example –**

```
Declare
    Age number;
    Today date;
    Birthdate date;
Begin
    Birthdate:=&birthdate;
    Select sysdate into today from dual;
    Age:=round((today-birthdate)/365);
    Dbms_output.put_line(age);
End;
/
```

Conditional Controls

If – Then – Elsif – End if

```
IF condition 1
THEN
   statement 1;
   statement 2;
ELSIF condtion2
THEN
   statement 3;
ELSE
   statement 4;
END IF
```

Nested IF's

```
IF condition 1
THEN
   IF condtion2
   THEN
      statement 1;
    ELSE
     statement 2;
    END IF;
ELSE
 statement 3;
END IF;
```

Logical Operators

Operator	Description	Example
and	Called logical AND operator.	(A and B) is false.
or	Called logical OR Operator.	(A or B) is true.
not	Called logical NOT Operator.	not (A and B) is true.

Example


```
DECLARE
   a number(3);
BEGIN
   a:=&a;
   IF ( a = 10 ) THEN
      dbms_output.put_line('Value of a is 10' );
   ELSIF ( a = 20 ) THEN
      dbms_output.put_line('Value of a is 20' );
   ELSIF ( a = 30 ) THEN
      dbms_output.put_line('Value of a is 30' );
   ELSE
      dbms_output.put_line('None of the values is matching');
   END IF;
      dbms_output.put_line('Exact value of a is: '|| a );
END;
/
```

NULL in PL/SQL

- NULL, its value is unknown -- indeterminate
- It is very different from
  - a blank or
  - a zero or
  - the Boolean value FALSE
- "Unknown" means that the variable has no value at all and so cannot be compared directly with other variables

NULL in Comparisions

- Example (Assigning NULL) –
  - my_string := NULL;
- Operators for NULL comparision
- IS NULL and IS NOT NULL operators.
- Syntax -
  - <identifier> IS NULL
  - <identifier> IS NOT NULL
  - <identifier> is the name of a variable, a constant, or a database column.

NULLs with database


```
Create table T(e number,f number);
insert into T values(8,null);
Select * from T;

Declare
    A number;
    B number;
Begin
    A:=&a;
    Select f into b from T where e=a;
    If (b is null) then
        Update T set f=0 where e=a;
    End if;
End;
/
```

Conditional Control - CASE

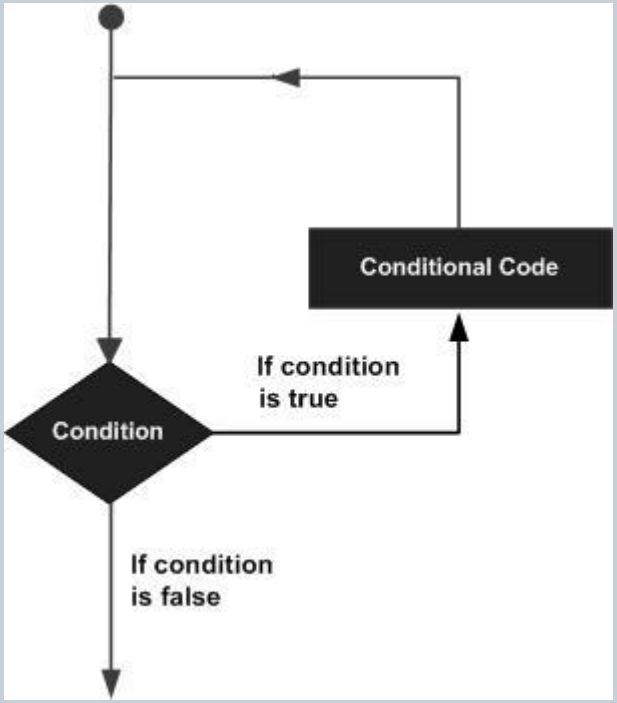
- Syntax –

```
CASE selector
   WHEN 'value1' THEN S1;
   WHEN 'value2' THEN S2;
   WHEN 'value3' THEN S3;
   ...
   ELSE Sn; -- default case
END CASE;
```

Example – CASE Statement


```
DECLARE
   grade char(1) := 'A';
BEGIN
   CASE grade
      when 'A' then dbms_output.put_line('Excellent');
      when 'B' then dbms_output.put_line('Very good');
      when 'C' then dbms_output.put_line('Well done');
      when 'D' then dbms_output.put_line('You passed');
      when 'F' then dbms_output.put_line('Better try again');
      else
      dbms_output.put_line('No such grade');
   END CASE;
END;
/
```

Iterative Control – LOOP's



Iterative Control – LOOP's

Loop Type	Description
PL/SQL Basic LOOP	Sequence of statements enclosed between the LOOP and END LOOP statements.
PL/SQL WHILE LOOP	Repeats a statement or group of statements while a given condition is true. It tests the condition before executing the loop body.
PL/SQL FOR LOOP	Execute a sequence of statements multiple times and abbreviates the code that manages the loop variable.
Nested loops in PL/SQL	You can use one or more loops inside any other basic loop, while or for loop.

Iterative Control – Basic LOOP

- Syntax –

```
LOOP
    Sequence of statements;
END LOOP;
```

Example – Loop…Exit When


```
DECLARE
   x number := 10;
BEGIN
   LOOP dbms_output.put_line(x);
      x := x + 10;
      exit WHEN x > 50;
   END LOOP; -- after exit, control resumes here
   dbms_output.put_line('After Exit x is: ' || x);
END;
/
```

contd…

Create table circle

(rad number,

Area float);

/

contd…


```
Declare
   Rad number:=1;
   Area float;
Begin
   Loop
      Area:=3.142*rad*rad;
      Insert into circle values(rad,area);
      Rad:=rad+1;
   Exit when rad>5;
   End loop;
End;
/
```

Iterative Control – While Loop

- Syntax –

```
WHILE condition LOOP
   sequence_of_statements
END LOOP;
```

Example – While Loop


```
Declare
   Rad number:=1;
   Area float;
Begin
   While(rad<6)
   loop
      Area:=3.142*rad*rad;
      Insert into circle values(rad,area);
      Rad:=rad+1;
   End loop;
End;
/
```

Iterative Control – For Loop

- Syntax –

```
FOR counter IN initial_value .. final_value LOOP
  sequence_of_statements;
END LOOP;
```

Example – For Loop


```
Declare
   Rad number;
   Area float;
Begin
   For rad in 1..5
   loop
      Area:=3.142*rad*rad;
      Insert into circle values(rad,area);
   End loop;
End;
/
```

Example – GOTO


```
DECLARE
    a number(2) := 10;
BEGIN
    <<loopstart>>
    -- while loop execution
    WHILE a < 20 LOOP
        dbms_output.put_line ('value of a: ' || a);
        a := a + 1;
        IF a = 15 THEN
            a := a + 1;
            GOTO loopstart;
         END IF;
    END LOOP;
END;
/
```

GOTO Statement

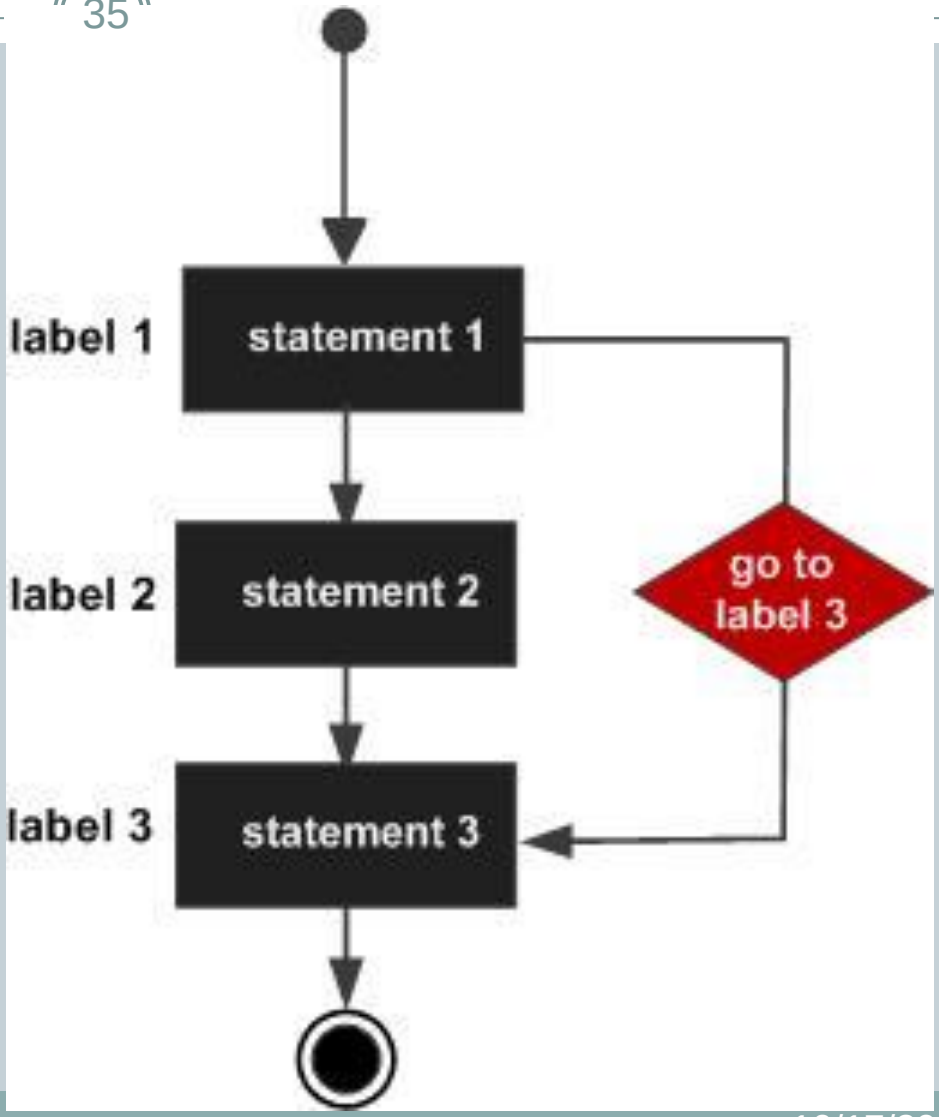
- Syntax

  GOTO label;

  .. ..

  << label >>

  statement;



Thank you