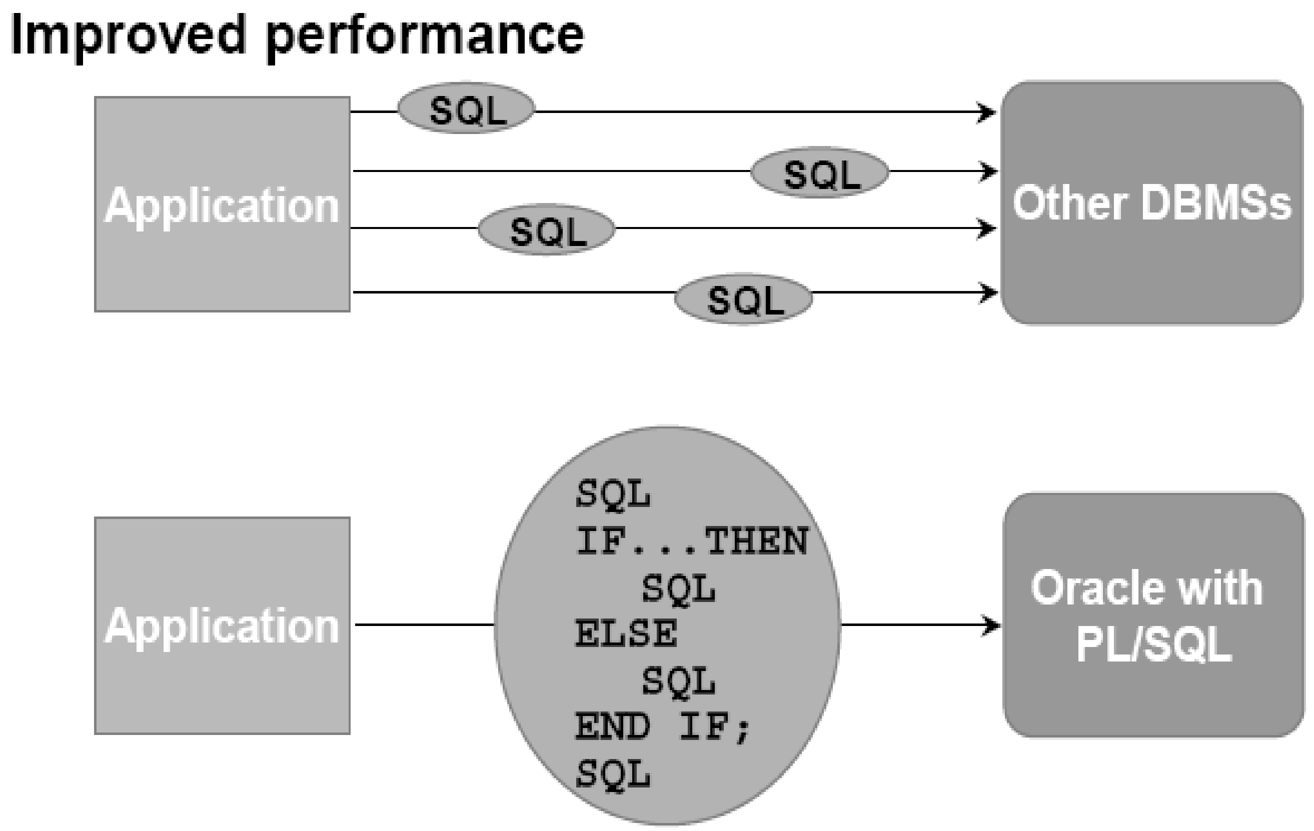
PL/SQL Introduction

By
Parteek Bhatia
Assistant Professor
Dept of Comp Sc & Engg
Thapar University
Patiala

Procedural Language/Structured Query Language

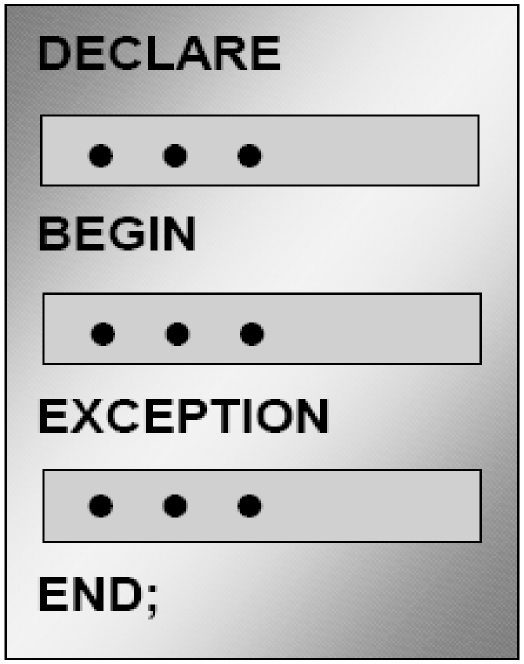
- PL/SQL is an extension of the SQL language.
- It eliminates many restrictions of the SQL Language.
- PL/SQL extends SQL by adding control structures found in the other procedural languages.
- Procedural constructs blend seamlessly with Oracle SQL, resulting in a structured, powerful language.
- PL/SQL combines the SQL's language's ease of data manipulation and the procedural language's ease of programming.

Benefits of PL/SQL



Benefits of PL/SQL

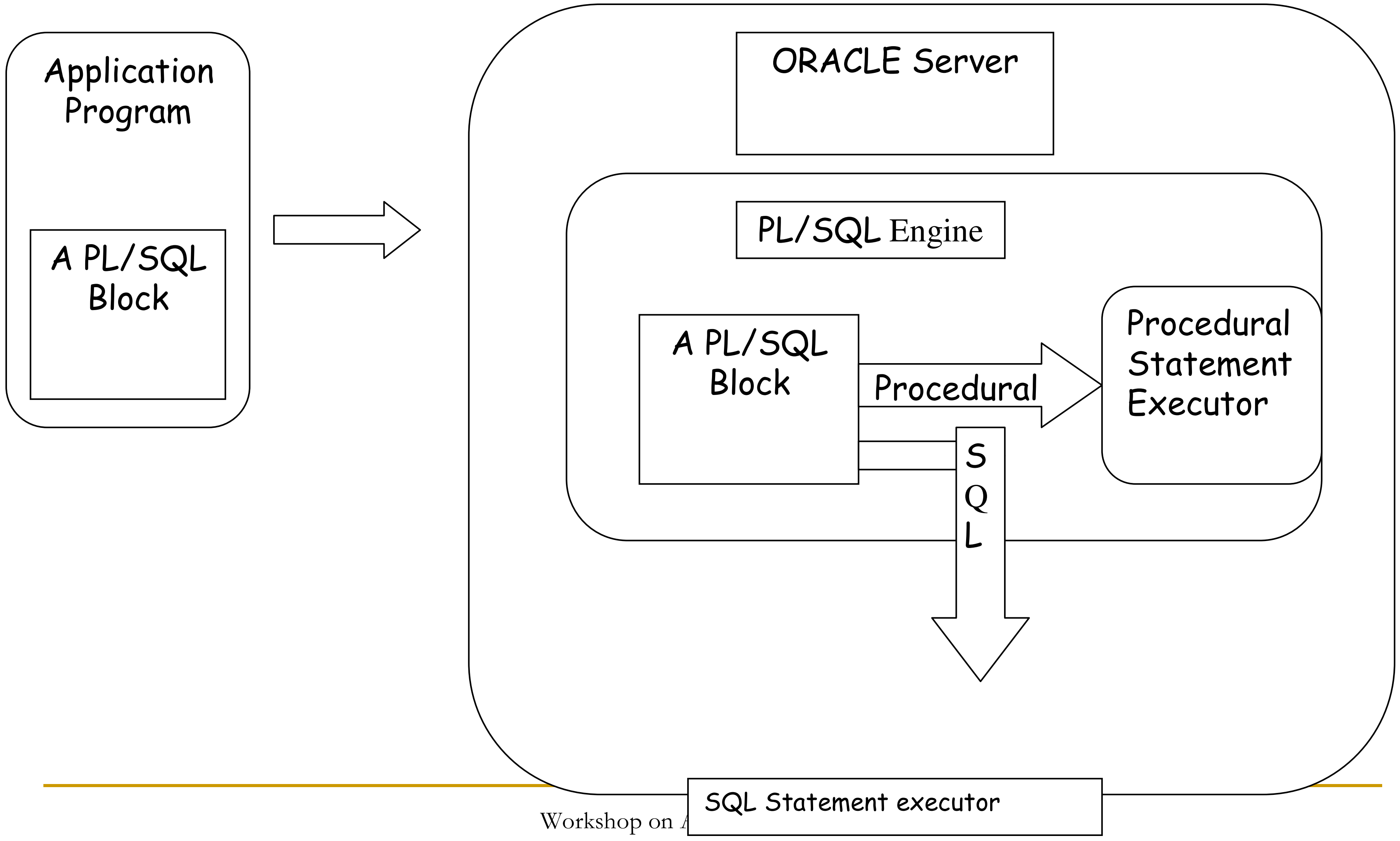
Modularize program development



Features of PL/DQL

- Supports the declaration and manipulation of object types and collections.
- Exception Handling
- Allows the calling of external functions and procedures.
- Triggers
- Cursors
- Support for SQL
- Support for Object-Oriented Programming

Architecture of PL/SQL



Arithmetic operators

Operator	Examples	Description
+	2+5; x+y;	Adds two operands
-	8-2; x-y;	Subtracts the second operand from the first
*	2*5; x*y	Multiplies two operands
/	12/2; x/y;	Divides the first operand by the second
**	3**2;x**2;	Raises the first operand to the exponent of the second

Expression operators

Operator	Examples	Description
:=	x:=y;a:=b*c;	Assigns the value of the operand or expression on the right to the operand on the left.
..	1..5; 4..d ; 4..d-1;	Defines an integer range from the first operand to the second.
||	'kit'||'kat'; x||y	Concatenates two or more strings.

Comments

- Comments can be single line or multiple lines.
- Single line comment:

  Begins with -- can appear within a statement, at end of line.

  Example: a number; --variable declaration
- Multi line comment

  Begin with /* and end with an asterisk-slash (*/).

  Example

  /* Statements to select rate and quantity into variables and calculate value */

Declaration of Variables

```
Declare
age   number (4) ;
Done Boolean ;
```

Variable Value Assignment

Three ways:

i) With Assignment operator

```
a : = b * c ;
increase : = sal * 1.5 ;
OK : = false ;
```

ii) With substitute variables

```
a:=&enter_number;
b:=&b;
```

iii) With Select into statement

```
Select col_name into var_name where cond;
```

Variable Value Assignment

- Select ename into e from emp where empno=100;
- SELECT sal * 0.15 INTO increased FROM emp where empno = emp_id ;
- In this case SELECT statement must return a single record, if it returns no record or more than one record then an error is raised.

Wap to calculate total sal of emp having empno 100. Table emp1 having empno, ename, bp, da, hra, total columns.

```
DECLARE
E NUMBER(3);
D NUMBER(3);
H NUMBER(3);
B NUMBER(3);
T NUMBER(3);
BEGIN
SELECT BP,DA, HRA INTO B, D, H FROM EMP1 WHERE
EMPNO=100;
T:=B+D+H;
UPDATE EMP1 SET TOTAL=T WHERE EMPNO=100;
END;
```

Constant Declaration

It may be useful for you to declare constants in the declaration section of the PL/SQL blocks developed as well. Constants are named elements whose values do not change.

For example, pi can be declared as a constant whose value 3.14 is never changed in the PL/SQL block. The declaration of a constant is similar to that of declaration of a variable. We can declare constants in the declaration section and use it elsewhere in the executable part. To declare the constant we must make use of the keyword constant. This keyword must precede the data type as shown below.

```
pi  constant number : = 3.14;
```

Variable Attributes

- Attributes allow us to refer to data types and objects from the database. PL/SQL variables and constants can have attributes. The following are the types of attributes, which are supported by PL/SQL.
- %TYPE
- %ROWTYPE

Variable Attributes

- %type

```
DECLARE
eno emp.emp_no%type;
```

- %rowtype

```
rec emp%rowtype;
```

BY USING % ROWTYPE

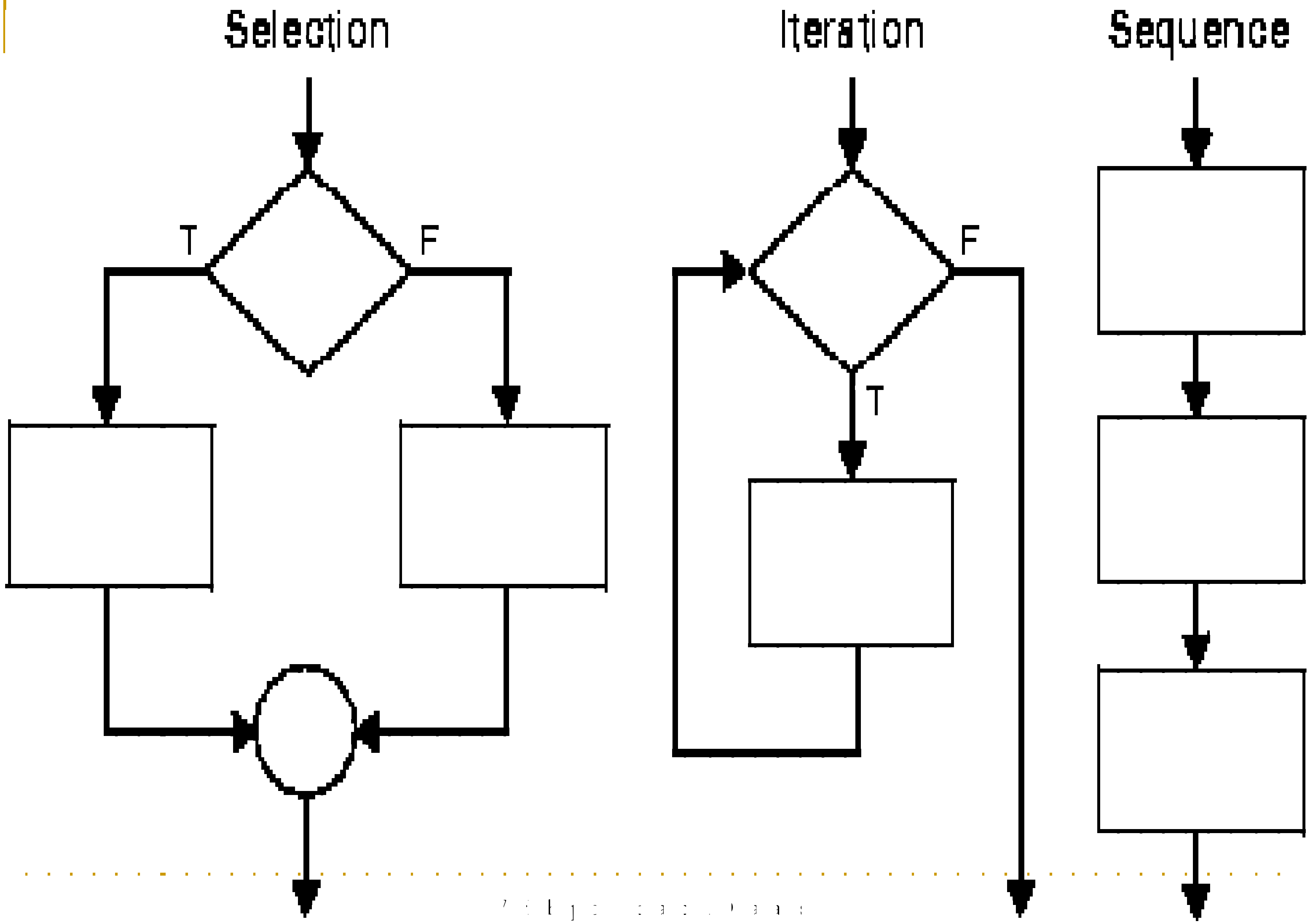
```
DECLARE

REC EMP1%ROWTYPE;

BEGIN

SELECT * INTO REC FROM EMP WHERE
EMPNO=100;

REC.TOTAL:=REC.BP+REC.HRA+REC.DA;

UPDATE EMP1 SET TOTOAL=REC.TOTAL
WHERE EMPNO=100;

END;
```

Displaying user messages

- DBMS_OUTPUT.PUT_LINE(‘ENTER THE NUMBER’);
- DBMS_OUTPUT.PUT_LINE(‘Result of Sum operation is: ’ || sum);
- To display messages to the user the SERVEROUTPUT should be set to ON. Here, DBMS_OUTPUT is a package name and PUT_LINE is a function.
- Syntax:
- SET SERVEROUTPUT [ON/OFF];

Control structures of PL/SQL

- Conditional Control
- Iterative Control
- Sequential Control


Selection
T
F
Iteration
F
T
Sequence

Conditional Control

- IF-THEN STATEMENT
- IF- THEN – ELSE STATEMENT
- IF-THEN-ELSIF STATEMENT (LADDER IF)

Conditional Control

```
IF a > b THEN
  dbms_ouput.put_line('a is greater');
END IF;
IF A > B THEN
DBMS_OUTPUT.PUT_LINE(' A IS GREATER ');
ELSE
DBMS_OUTPUT.PUT_LINE(' B IS GREATER ');
END IF;
```

Illustration of IF-THEN-ELSIF, PL/SQL block to calculate addition, subtraction, multiplication and division of two numbers according to user choice.

```
DECLARE
            A NUMBER=&A;
            B NUMBER:=&B;
            C NUMBER;
            X NUMBER;
BEGN
X:=&ENTER_CHOICE;
 IF X=1 THEN
   C:=A+B;
ELSIF X=2 THEN
C:=A-B;
ELSIF X=2 THEN
        C:=A-B;
ELSIF X=3 THEN
                C:=A*B;
```

```
ELSIF X=4 THEN
                                    C:=A/B;
ELSE
        DBMS_OUTPUT.PUT_LINE('NOT A VALID OPTION');
END IF;
            DBMS_OUTPUT.PUT_LINE('RESULT IS' || C);
END;
```

Iterative Control

- Simple Loop Statement
- While Loop Statement
- For Loop Statement

Iterative Control

- **Simple LOOP Statement**

```
LOOP
  sequence_of_statements;
END LOOP;
```

Need to use EXIT or EXIT WHEN to exit out of loop.

```
IF c > 5 THEN       |    EXIT WHEN c > 5;
  EXIT;             |
END IF;             |
```

Loop Labels

```
<<label_name>>
LOOP
      sequence_of_statements;
END LOOP;
<<my_loop>>
LOOP
...
END LOOP my_loop;

<<outer>>
LOOP
      ...
      LOOP
                ...
                EXIT outer WHEN ...  /* exit both loops, if we only use EXIT WHEN
                without label it exit only the inner loop */
      END LOOP;
...
END LOOP outer;
```

WHILE-LOOP

```
 WHILE condition LOOP
   sequence_of_statements;
END LOOP;

WHILE i <= 10 LOOP
   a:=n*i;
   i:=i+1;
END LOOP;
```

Illustration of WHILE LOOP, PL/SQL block to print multiplication table any number

•

```
 DECLARE
        TABLE_OF NUMBER :=&ENTER_TABLEOF ;
        COUNT NUMBER := 1;
        RESULT NUMBER;
BEGIN
        WHILE COUNT <= 10 LOOP
            RESULT := TABLE_OF * COUNT ;
            DBMS_OUTPUT.PUT_LINE(TABLE_OF||' * '||
                    COUNT ||'='||RESULT);
            COUNT := COUNT +1 ;
        END LOOP;
END;
```

FOR-LOOP Statement

```
FOR counter IN [REVERSE] lower_bound..higher_bound LOOP
     sequence_of_statements;
END LOOP;

FOR i IN 1..3 LOOP  -- assign the values 1,2,3 to i   sequence_of_statements;
     -- executes three times
END LOOP;

FOR i IN 3..3 LOOP  -- assign the value 3 to i   sequence_of_statements;  --
     executes one time
END LOOP;

FOR i IN REVERSE 1..3 LOOP  -- assign the values 3,2,1 to i
     sequence_of_statements;  -- executes three times
END LOOP;
```

To design a for loop that execute for multiple of n only

```
FOR j IN 5..15 LOOP
    IF MOD(j, 5) = 0 THEN       -- pass multiples of 5
          sequence_of_statements;  -- j has values 5,10,15
    END IF;
END LOOP;
```

Scope Rules

```
FOR ctr IN 1..10
      LOOP
      ..
      END LOOP;
sum := ctr - 1;  -- illegal

DECLARE
      ctr  INTEGER;
BEGIN
 ...
FOR ctr IN 1..25 LOOP
      ...
      IF ctr > 10 THEN
      ...  -- refers to loop counter
      END LOOP;
END;
```

Scope Rules

```
<<main>>
DECLARE
   ctr  INTEGER;
BEGIN
   ………..
   FOR ctr IN 1..25 LOOP
        IF main.ctr > 10 THEN
        ...  -- refers to global variable
   END LOOP;
END main;
```

Sequential Control

- GOTO statement
- NULL statement
- **GOTO Statement**

```
 BEGIN   GOTO insert_row;
………….
<<insert_row>>   INSERT INTO emp VALUES
...
END;
```

Null Statement

```
LOOP
….
IF done THEN
GOTO end_loop;
END IF;
--
<<end_loop>>  -- illegal
END LOOP;  -- not an executable statement
```

Solution is:

```
LOOP
…
IF done THEN
GOTO end_loop;
END IF;
--
<<end_loop>>  -- illegal
NULL;  -- an executable statement
END LOOP;  -- not an executable statement
END;
```

Thanks

Lets Implement it in Lab Session