TATA CONSULTANCY SERVICES

Oracle PL/SQL – Expressions and Operators

Expressions

- Expressions are constructed using operands and operators.
- An operand is a variable, constant, literal, or function call that contributes a value to an expression
- For example, 3 + 2 is a simple expression where 3 and 2 are operands and '+' is a operator.
- An expression always returns one value.

Types of Operators:

PL/SQL provides the following types of operators:

- Arithmetic Operators
- Relational Operators
- Comparison Operators
- Logical Operators
- Assignment Operator (:=)
- Concatenation Operator (||)

1. Arithmetic Operators :

Below list shows the different types of arithmetic operators

Operator	Description
+	Adds two operands
-	Subtracts second operand from the first
*	Multiplies both operands
/	Divides first operand by second
**	Exponentiation Operator,one operand power of other

Example :

```
set serveroutput on;
declare
-- declare variables
num1 number(10) := 20;
num2 number(10) := 10;
begin
-- Print the expression values
dbms_output.put_line('Sum :' || (num1+num2));
dbms_output.put_line('Difference :' || (num1-num2));
dbms_output.put_line('Product :' || (num1*num2));
dbms_output.put_line('Division :' || (num1/num2));
dbms_output.put_line('Exponentiation :' || (num1/num2));
end;
/
```

Output :

```
Sum :30
Difference :10
Product :200
Division :2
Exponentiation :2

PL/SQL procedure successfully completed
```

2. Relational Operators :

Relational operators compare two expressions or values and return a Boolean result. Character values can also be checked for equality or inequality. By default, comparisons are based on the binary values of each byte in the string.

Following table shows all the relational operators supported by PL/SQL.

Operator	Description
=	Checks if the values of two operands are equal or not, if yes then condition becomes true.
!= or <>	Checks if the values of two operands are equal or not, if values are not equal then condition becomes true.
>	Checks if the values of left operand is greater than the value of right operand, if yes then condition becomes true.

<	Checks if the values of left operand is less than the value of right operand, if yes then condition becomes true.
>=	Checks if the value of left operand is greater than or equal to the value of right operand, if yes then condition becomes true.
<=	Checks if the value of left operand is less than or equal to the value of right operand, if yes then condition becomes true.

3. Comparison Operators :

Comparison operators are used for comparing one expression to another. The result is always either TRUE, FALSE OR NULL.

Operator	Description
LIKE	The LIKE operator compares a character, string, or CLOB value to a pattern and returns TRUE if the value matches the pattern and FALSE if it does not.
BETWEEN	The BETWEEN operator tests whether a value lies in a specified range. x BETWEEN a AND b means that x >= a and x <= b.
IN	The IN operator tests set membership. x IN (set) means that x is equal to any member of set.
IS NULL	The IS NULL operator returns the BOOLEAN value TRUE if its operand is NULL or FALSE if it is not NULL. Comparisons involving NULL values always yield NULL.

4. Logical Operators :

Logical operators work on Boolean operands and produces Boolean results.

Following table shows the Logical operators supported by PL/SQL.

Operator	Description
AND	Called logical AND operator. If both the operands are true then condition becomes true.
OR	Called logical OR Operator. If any of the two operands is true then condition becomes true.
NOT	Called logical NOT Operator. Used to reverse the logical state of its operand. If a condition is true then logical NOT operator will make it false and vice-versa.

5. Concatenation Operator:

'||' serve as the concatenation operator, which appends one string (CHAR, VARCHAR2, CLOB, or the equivalent Unicode-enabled type) to another.

Example :

The below example gets the employee name and employee number as input from the user.

```
SET SERVEROUTPUT ON;
DECLARE
     a VARCHAR2(10);
     b NUMBER;
BEGIN
     a := &empname;
     b := &empno;
     dbms_output.put_line('Welcome '||a||','||b);
END;
/
```

Output :

```
Welcome Mary,1112233

PL/SQL procedure successfully completed
```