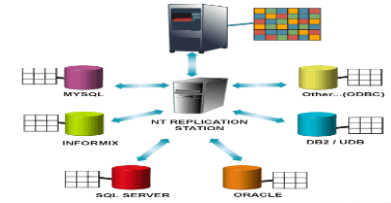


4.1. Fundamentos del Lenguaje PL-SQL -ORACLE



ESTRUCTURA DE LOS BLOQUES DE PROGRAMA

UNIDADES LEXICAS

Las unidades léxicas pueden clasificarse en *identificadores*, *delimitadores*, *literales* y *comentarios*.

LITERALES

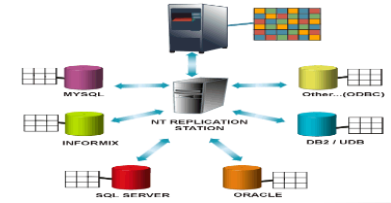
Una literal es un valor numérico, booleano o de carácter que no es un identificador.

Ejemplo de estos serian **-23.456** y **NULL**

LITERALES DE CARACTER

- Las literales de carácter, también conocidos como literales de cadena, constan de uno o mas caracteres delimitadas por comillas simple.
- Los literales de carácter pueden asignarse a variables de tipo CHAR o VARCHAR2, sin necesidad de hacer ningún tipo de conversión. Ejemplos validos:
 - **'12345'**
 - **'Four score and seven years ago...'**
 - **'100%'**
 - **''**
 - **''** que es una cadena de longitud cero el literal de la cadena de longitud cero se considera idéntico a NULL.

4.1. Fundamentos del Lenguaje PL-SQL -ORACLE



ESTRUCTURA DE LOS BLOQUES DE PROGRAMA

UNIDADES LEXICAS

Las unidades léxicas pueden clasificarse en *identificadores*, *delimitadores*, *literales* y *comentarios*.

LITERALES

Una literal es un valor numérico, booleano o de carácter que no es un identificador.

Ejemplo de estos serian **-23.456** y **NULL**

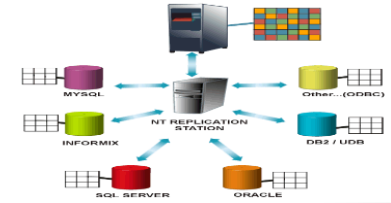
LITERALES NUMERICAS

Un literal numérica representa un valor entero o real, y puede ser asignado a una variable de tipo NUMBER sin necesidad de efectuar conversión alguna.

Algunos ejemplos validos son:

Literales Enteras	Literales reales
123	-17.1
-7	23.0
+12	3.
0	

4.1. Fundamentos del Lenguaje PL-SQL -ORACLE



ESTRUCTURA DE LOS BLOQUES DE PROGRAMA

UNIDADES LEXICAS

Las unidades léxicas pueden clasificarse en *identificadores*, *delimitadores*, *literales* y *comentarios*.

LITERALES

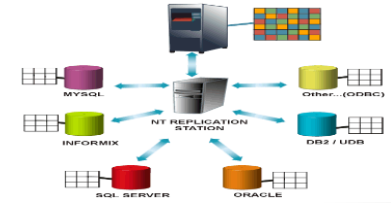
Una literal es un valor numérico, booleano o de carácter que no es un identificador.

Ejemplo de estos serian **-23.456** y **NULL**

LITERALES BOOLEANOS

- Solo hay tres posibles literales booleanos: **TRUE (verdadero)**, **FALSE (falso)** y **NULL (nulo)**.
- Estos valores solo pueden ser asignados a una variable booleana.
- Las literales representan la verdad o falsedad de una condición y se utilizan en las ordenes IF y LOOP

4.1. Fundamentos del Lenguaje PL-SQL -ORACLE



ESTRUCTURA DE LOS BLOQUES DE PROGRAMA

UNIDADES LEXICAS

Las unidades léxicas pueden clasificarse en *identificadores*, *delimitadores*, *literales* y *comentarios*.

LITERALES

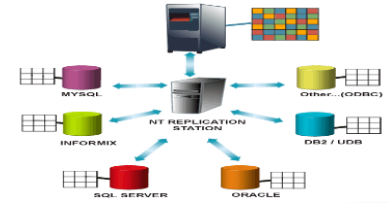
Una literal es un valor numérico, booleano o de carácter que no es un identificador.

Ejemplo de estos serian **-23.456** y **NULL**

COMENTARIOS

- Los comentarios facilitan la lectura y ayudan a comprender mejor el código fuente de los programas
- Están clasificados como **comentarios monolíneas** y **comentarios multilíneas**

4.1. Fundamentos del Lenguaje PL-SQL -ORACLE



ESTRUCTURA DE LOS BLOQUES DE PROGRAMA

UNIDADES LEXICAS

Las unidades léxicas pueden clasificarse en *identificadores*, *delimitadores*, *literales* y *comentarios*.

COMENTARIOS

DECLARE

-- Declaración de las variables que se usan en este bloque **COMENTARIO MONOLINEA**

v_Num1 NUMBER := 1;

v_Num2 NUMBER := 2;

v_String1 VARCHAR(50) := 'Hello World!';

v_String2 VARCHAR(50) := ' -This message brought to you by PL/SQL!';

V_OutputStr VARCHAR2(50);

BEGIN

-- Primero inserta dos filas en temp_table, utilizando los valores de las variable **COMENTARIO MONOLINEA**

INSERT INTO temp_table (num_col, char_col) VALUES (v_num1, v_String1);

INSERT INTO temp_table (num_col, char_col) VALUES (v_num2, v_String2);

/*Ahora consulta temp_table para las dos filas que se acaban de insertar y las presenta en pantalla utilizando el paquete

DBMS_OUTPUT */ **COMENTARIO MULTILINEA**

SELECT char_cal INTO v_OutputStar

WHERE num_col = v_num1;

DBMS_OUTPUT.PUT_LINE(v_OutputStar);

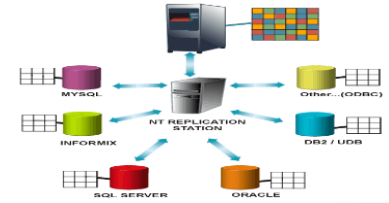
SELECT char_cal INTO v_OutputStar

WHERE num_col = v_num2;

DBMS_OUTPUT.PUT_LINE(v_OutputStar);

END;

4.1. Fundamentos del Lenguaje PL-SQL -ORACLE



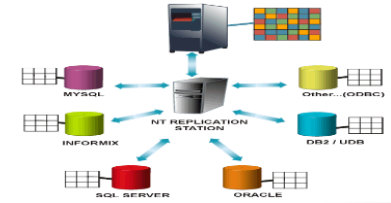
ESTRUCTURA DE LOS BLOQUES DE PROGRAMA

DECLARACIONES DE VARIABLES

La comunicación con la base de datos tiene lugar mediante el uso de variables incluidas en los bloques PL/SQL.

- Las variables son espacios de memoria que pueden contener valores de datos
- A medida que se ejecuta el programa, el contenido de la variable puede cambiar y suele hacerlo.
- Se puede asignar información de la base de datos a las variables y también puede insertar el contenido de una variable en la base de datos.
- La variables son declarada en la sección declarativa de un bloque y cada una de ella tiene un tipo específico, que describe el tipo de información que podemos almacenar en ella.

4.1. Fundamentos del Lenguaje PL-SQL -ORACLE



ESTRUCTURA DE LOS BLOQUES DE PROGRAMA

DECLARACIONES DE VARIABLES

SINTAXIS DE LAS DECLARACIONES

Las declaraciones de variables tienen lugar en la sección declarativa del bloque. Y la sintaxis general seria

nombre_variable tipo [CONSTANT][NOT NULL] [:= valor]

Algunos ejemplos de declaración Legal:

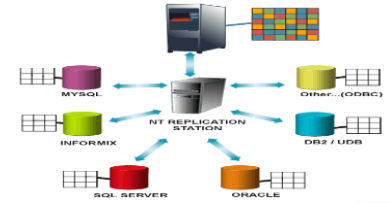
DECLARE

v_Description	VARCHAR(50);
v_NumberSeats	NUMBER := 45;
v_Counter	BINARY_INTEGER := 0;

Ejemplo de Declaración Ilegal

V_tempVar	NUMBER NOT NULL;
------------------	-------------------------

4.1. Fundamentos del Lenguaje PL-SQL -ORACLE



ESTRUCTURA DE LOS BLOQUES DE PROGRAMA

DECLARACIONES DE VARIABLES

SINTAXIS DE LAS DECLARACIONES

Ejemplo de Declaración Ilegal y corrigiendo el error:

```
DECLARE
v_tempVar      NUMBER NOT NULL := 0;
```

Si se utiliza la restricción NOT NULL, se debe asignar un valor cuando se declara la variable

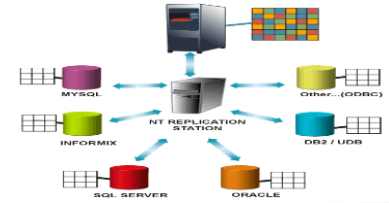
Si queremos declarar un constante:

```
DECLARE
v_MinimumStudentID CONSTANT NUMBER(5) := 10000;
```

Podemos usar la palabra clave DEFAULT:

```
v_NumberSeats      NUMBER DEFAULT 45;
v_Counter           BINARY_INTEGER DEFAULT 0;
v_FirstName         VARCHAR2 (20) DEFAULT 'Scott';
```


4.1. Fundamentos del Lenguaje PL-SQL -ORACLE



ESTRUCTURA DE LOS BLOQUES DE PROGRAMA

DECLARACIONES DE VARIABLES

SINTAXIS DE LAS DECLARACIONES

Solo puede haber una declaración de variable por línea en la sección declarativa del bloque.

Declaración Ilegal:

DECLARE

V_FirstName, v_LastName VARCHAR2 (20);

Declaración Legal:

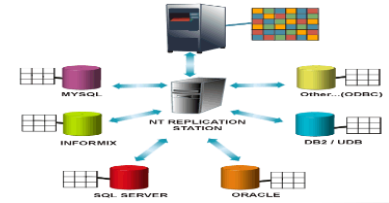
v_FirstName VARCHAR2 (20);

v_LastName VARCHAR2 (20);

INICIALIZACION DE VARIABLES:

- **EN PL/SQL se define el contenido de las variables inicializadas, al contenido de estas variables se le asigna el valor de NULL.**

4.1. Fundamentos del Lenguaje PL-SQL -ORACLE

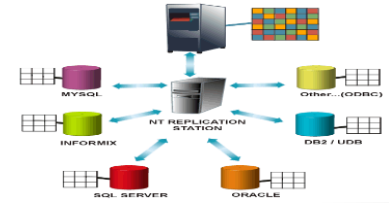


ESTRUCTURA DE LOS BLOQUES DE PROGRAMA

TIPOS DE DATOS EN PL/SQL (hacer referencia al Capitulo Anterior)

- Existen cuatro categorías de tipos de datos: Escalares, Compuestos, Referencias, y LOB (Large Objects)
- Los tipos Escalares no tienen componentes, mientras que los tipos Compuestos si, por que las referencias son punteros a otros tipos.
- Los tipos de datos PL/SQL se definen un paquete llamado STANDARD, cuyos contenidos son accesibles desde cualquier bloque PL/SQL.
- El paquete STANDARD también define las funciones predefinidas SQL y de conversión disponibles en PL/SQL.

4.1. Fundamentos del Lenguaje PL-SQL -ORACLE



ESTRUCTURA DE LOS BLOQUES DE PROGRAMA

UTILIZACION DE %TYPE

Las variables PL/SQL se emplean para manipular datos almacenados en la base de datos. La variable debe tener el mismo tipo que la columna de la tabla.

Por ejemplo, columna **first_names** de la tabla **students** que es de tipo **VARCHAR2(20)**. Tomando como referencia esta información podemos declarar:

Declare

v_FirstName VARCHAR2(20);

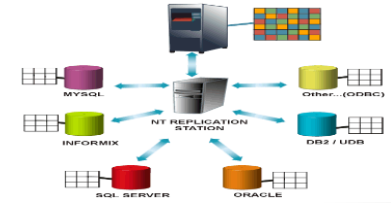
Pero que ocurre si cambia la dimensión a mayor de la columna y se esta usando en un programa, habría cambiar variable que haga referencia a la columna.

Para corregir esto usamos en %TYPE para definir la variable:

Declare

v_FirstName students.first_names%TYPE;

4.1. Fundamentos del Lenguaje PL-SQL -ORACLE



ESTRUCTURA DE LOS BLOQUES DE PROGRAMA

UTILIZACION DE %TYPE

El siguiente ejemplo ilustra varias aplicación del atributo **%TYPE**

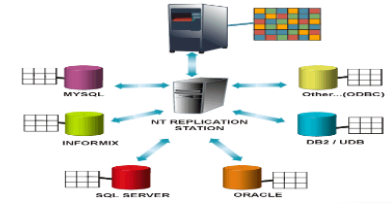
DECLARE

```
v_RoomID      classes.room_id%TYPE ;           -- devuelve NUMBER (5)
v_RoomID2      v_RoomID %TYPE ;                 -- devuelve NUMBER (5)
v_TempVar      NUMBER(7,3) NOT NULL := 12.3 ;
v_AnotherVar    v_TempVar %TYPE ;               -- devuelve NUMBER (7,3)
```

Si se aplica **%TYPE** a una variable o a una columna que haya sido definida con la restricción **NOT NULL** como **classes.room_id** y **v_TempVar**, el tipo devuelto no tiene esta restricción.

La utilización de **%TYPE** constituye una buena practica de programación, porque hace que los programas PL/SQL sean mas flexibles y capaces de adaptarse a las definiciones cambiantes de la base datos.

4.1. Fundamentos del Lenguaje PL-SQL -ORACLE



ESTRUCTURA DE LOS BLOQUES DE PROGRAMA

CONVERSIONES ENTRE TIPOS DE DATOS

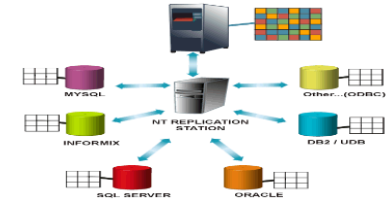
PL/SQL puede manejar conversiones entre tipos de datos escalares de las distintas familias. Para los tipos de una familia se puede realizar conversiones sin ninguna restricción, excepto las impuestas a las variables.

Por ejemplo un CHAR(10) no puede convertirse en un VARCHAR2(1), un NUMBER(3,2) no puede convertirse en un NUMBER(3) porque el cuando se producen violaciones a las restricciones el compilador PL/SQL no dará un error pero se puede producir errores en tiempo de ejecución dependiendo de los valores de las variables a convertir.

Los tipos compuesto no pueden convertirse entre si, porque son demasiados distintos. Pero se puede escribir un función que realice la conversión, basándose en tipo de dato que tenga en el programa.

Existen dos tipos de conversión: implícita y explícita

4.1. Fundamentos del Lenguaje PL-SQL -ORACLE



ESTRUCTURA DE LOS BLOQUES DE PROGRAMA

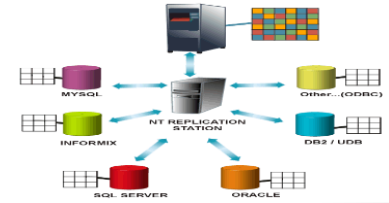
CONVERSIONES ENTRE TIPOS DE DATOS

Conversión explícita de tipos de datos

Las función de conversión de SQL también están disponibles en PL/SQL. Pueden ser empleadas cuando se requiera, para realizar conversiones explícitas entre variables de diferentes familia de tipos.

Función	Descripción	Familias que se Puede Convertir
TO_CHAR	Convierte su argumento en tipo VARCHAR2, dependiendo del especificador de formato opcional	Numéricos, Fechas
TO_DATE	Convierte su argumento en tipo DATE, dependiendo del especificador de formato opcional	Carácter
TO_NUMBER	Convierte su argumento en tipo NUMBER, dependiendo del especificador de formato opcional	Carácter
RAWTOHEX	Convierte un valor RAW en un representación hexadecimal de la cantidad en binario	Raw
HEXTORAW	Convierte una representación hexadecimal en el equivalente binario	Carácter(en representación hexadecimal)
CHARTOROWID	Convierte una representación de caracteres de un ROWID el formato interno binario	Carácter(en formato rawid d 18 caracteres)
ROWIDTOCHAR	Conviene una variable interna ROWID al formato externo de 18 caracteres	Rowid

4.1. Fundamentos del Lenguaje PL-SQL -ORACLE



ESTRUCTURA DE LOS BLOQUES DE PROGRAMA

CONVERSIONES ENTRE TIPOS DE DATOS

Conversión implícita de tipos de datos

PS/SQL realizara conversiones automáticas entre familias de tipos, siempre que se posible.

Por ejemplo, el siguiente bloque extrae el numero actual de crédito del estudiante 10002.

DECLARE

v_CurrentCredits VARCHAR2(5);

BEGIN

SELECT current_credits
INTO v_CurrentCredits
FROM students
WHERE id = 10002;

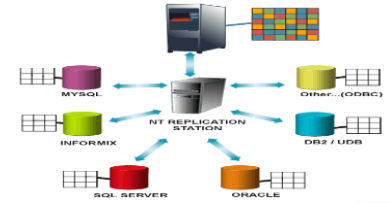
END;

En la base de datos, **current_credits** es un campo tipo **NUMBER(3)**, mientras que **v_CurrentCredits** es una variable **VARCHAR2(5)**. PLSQL convertirá automáticamente el dato numérico en una cadena de caracteres, y asignara esta a la variable de tipo carácter.

PL/SQL puede realizar conversiones entre

- **Caracteres y números**
- **Caracteres y fechas**

4.1. Fundamentos del Lenguaje PL-SQL -ORACLE



ESTRUCTURA DE LOS BLOQUES DE PROGRAMA

EXPRESIONES Y OPERADORES

Las expresiones y los operadores son el pegamento que permite unir las variables PS/SQL.

- *Los operadores definen como se asignan valores a la variables y como se manipulan dichos valores.*
- *Una expresión es una secuencia de valores y literales, separados por operadores.*
- *El valor de una expresión se determina a partir de los valores de las variables y literales que la componen y de la definición de los operadores.*

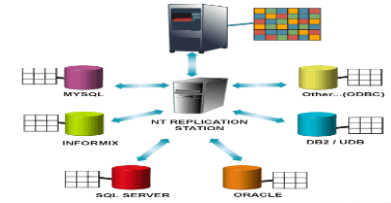
Asignación

Es el operador mas básico. Su sintaxis es

variable := expresión;

Donde variable y expresión son variables y expresión de PL/SQL

4.1. Fundamentos del Lenguaje PL-SQL -ORACLE



ESTRUCTURA DE LOS BLOQUES DE PROGRAMA

EXPRESIONES Y OPERADORES

Asignación

Ejemplo de una orden de asignación

```
DECLARE
    v_String1    VARCHAR2(10);
    v_String2    VARCHAR2(15);
    v_Numeric    NUMBER;

BEGIN
    v_String1 := 'Hello';
    v_String2 := v_String1;
    v_Numeric := -12.4;

END;
```

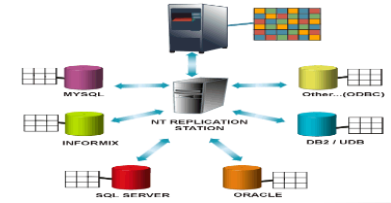
En una orden dada solo puede haber una asignación. A diferencia de otros lenguajes. Esto es una asignación ilegal

```
DECLARE
    v_Val1    NUMBER;
    v_Val2    NUMBER;
    v_Val3    NUMBER;

BEGIN
    v_Val1 := v_Val2 := v_Val3 := 0;

END;
```

4.1. Fundamentos del Lenguaje PL-SQL -ORACLE



ESTRUCTURA DE LOS BLOQUES DE PROGRAMA

EXPRESIONES Y OPERADORES

Expresiones

Las expresiones PL/SQL son valores. Por si solo una expresión no es valida como orden independiente, sino que tiene ser parte de otra orden. Los operadores que componen una expresión determinan, junto con el tipo de los operando, cual es el tipo de la expresión.

Expresiones Numérica

Estas expresión dependen de la procedencia de los operadores

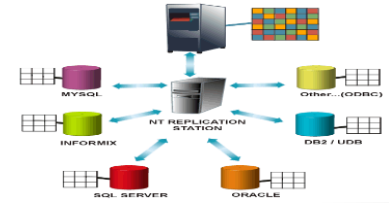
Estas dos expresiones no son equivalente $3 + 5 * 7 = (3 + 5) * 7$ ¿ Porque?

Expresiones de Caracteres

El único operador de caracteres es la concatenación (||).

Ejemplo de una expresión: 'Hello ' || 'World' || '!' el resultado 'Hello World!'

4.1. Fundamentos del Lenguaje PL-SQL -ORACLE



ESTRUCTURA DE LOS BLOQUES DE PROGRAMA

EXPRESIONES Y OPERADORES

Expresiones de Caracteres

Si todos los operando de un expresión de concatenación son tipo CHAR, entonces la expresión también lo es. Si uno de los operando es tipo VARCHAR2, entonces la expresión también lo es. Las literales de cadena se consideran de tipo CHAR de forma que la expresion resultante también lo es.

DECLARE

```
v_TempVar  VARCHAR2(10) := 'PL';
```

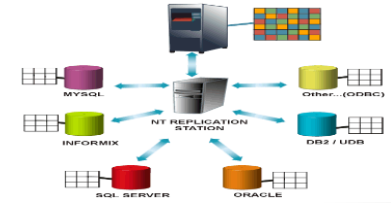
```
v_Result   VARCHAR2(10) ;
```

BEGIN

```
v_Result := v_TempVar || '/SQL';
```

END;

4.1. Fundamentos del Lenguaje PL-SQL -ORACLE



ESTRUCTURA DE LOS BLOQUES DE PROGRAMA

EXPRESIONES Y OPERADORES

Expresiones Booleanas

Todas las expresiones de control PL/SQL (excepto GOTO) incluyen expresiones booleanas, también denominadas condiciones. Una expresión booleana es una expresión que tiene como resultado un valor booleano (**TRUE**, **FALSE** o **NULL**)

Ejemplos de expresiones booleanas:

X > Y

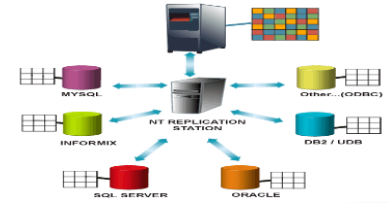
NULL

(4 > 5) OR (-1 != Z)

Hay tres operadores (**AND** , **OR** y **NOT**) que admiten argumentos booleanos y devuelven valores booleanos. Su comportamiento se describe en la tabla de la verdad.

Estos operadores implementan la lógica trivaluada estándar.

4.1. Fundamentos del Lenguaje PL-SQL -ORACLE



ESTRUCTURA DE LOS BLOQUES DE PROGRAMA

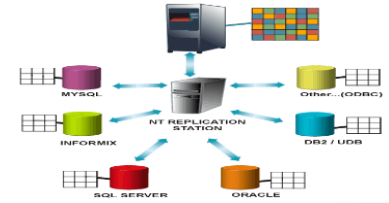
EXPRESIONES Y OPERADORES

Expresiones Booleanas

Estos operadores implementan la lógica trivaluada estándar.

- Por ejemplo **AND** devuelve **TRUE** si sus dos operando toman el valor de **TRUE**. Y **OR** devuelve **FALSE** si su dos operando toman el valor de **FALSE**
- Los valores **NULL** añaden complejidad a las expresiones booleanas '**NULL significa valor desconocido o no definido**'. La expresion
 - **TRUE AND NULL** da como resultado **NULL** porque no sabemos si el segundo operando tiene el valor de **TRUE** o no.
- El operador **IS NULL** devuelve **TRUE** solo si su operador es **NULL**.

4.1. Fundamentos del Lenguaje PL-SQL -ORACLE



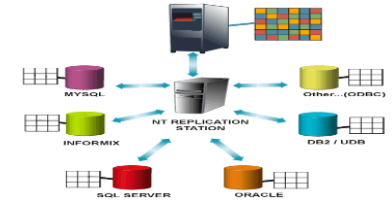
ESTRUCTURA DE LOS BLOQUES DE PROGRAMA

EXPRESIONES Y OPERADORES

Expresiones Booleanas

- El Operador **LIKE** se usa para comparaciones con patrones en cadena de caracteres. El carácter de subrayado (**_**) se corresponde con exactamente un carácter, mientras que el carácter de porcentaje(**%**) se corresponde con cero o mas caracteres.
- Ejemplo de expresiones que devuelven valor TRUE
 - 'Scott' LIKE 'Sc%t'**
 - 'Scott' LIKE 'SC_tt'**
 - 'Scott' LIKE '%'**
- El operador **BETWEEN** combina **<=** y **>=** en una única expresión. La siguiente expresion, por ejemplo :
 - 100 BETWEEN 110 AND 120** Resultado es FALSE
 - 100 BETWEEN 90 AND 110** resultado es TRUE

4.1. Fundamentos del Lenguaje PL-SQL -ORACLE



ESTRUCTURA DE LOS BLOQUES DE PROGRAMA

EXPRESIONES Y OPERADORES

Expresiones Booleanas

- El operador **IN** devuelve **TRUE** si su primer operando esta contenido en un conjunto identificado por un segundo operando. La siguiente expresion, por ejemplo :

'Scott' IN ('Mike', 'Pamela', 'Fred') Resultado es **FALSE**

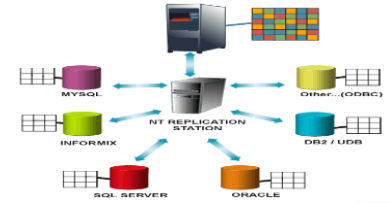
Operadores Validos

Operador	Definición
=	Igual a
!=	Distinto de
<	Menor que
>	Mayor que
<=	Menor o igual que
>=	Mayor o Igual que

TABLA DE LA VERDAD

NOT	TRUE	FALSE	NULL
	FALSE	TRUE	NULL
AND	TRUE	FALSE	NULL
TRUE	TRUE	FALSE	NULL
FALSE	FALSE	FALSE	FALSE
NULL	NULL	NULL	NULL
OR	TRUE	FALSE	NULL
TRUE	TRUE	TRUE	TRUE
FALSE	TRUE	FALSE	NULL
NULL	TRUE	NULL	NULL

4.1. Fundamentos del Lenguaje PL-SQL -ORACLE



ESTRUCTURA DE LOS BLOQUES DE PROGRAMA

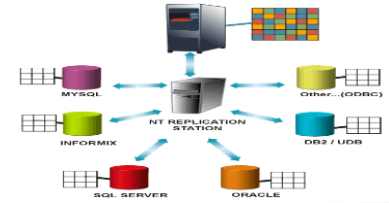
ESTRUCTURAS DE CONTROL PL/SQL

- El PL/SQL tiene diversas estructuras de control que permiten controlar el comportamiento de los bloques a medida que estos se ejecutan. Pueden contener ordenes condicionales y los bucles. Son estas estructuras que combinadas con las variables dotan a PL/SQL de su poder y flexibilidad.
- IF-THEN-ELSE

La sintaxis para esta orden sería

```
IF expresion_booleana1 THEN
    secuencia_de_ordenes1;
[ELSIF expresion_booleana2 THEN
    Secuencia_de_ordenes2;]
[ELSE
    secuencia_de_ordenes3;]
END IF;
```


4.1. Fundamentos del Lenguaje PL-SQL -ORACLE



ESTRUCTURA DE LOS BLOQUES DE PROGRAMA

ESTRUCTURAS DE CONTROL PL/SQL

- Ejemplo de la orden IF-THEN-ELSE

DECLARE

`v_NumberSeats rooms.number_seats%TYPE;`

`v_Comment VARCHAR(35);`

BEGIN

`/* Extrae el numero de asiento de la habitación, cuyo identificador es 99999 y almacena el resultado en v_NumberSeats. */`

`SELECT number_seats`

`INTO v_NumberSeats`

`FROM rooms`

`WHERE room_id = 99999;`

`IF v_NumberSeats < 50 THEN`

`v_comment := 'Fairly small';`

`ELSIF v_NumberSeats < 100 THEN`

`v_comment := 'A little bigger';`

`ELSE`

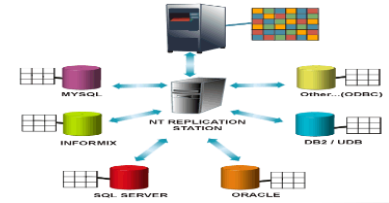
`v_comment := 'Lots of room';`

`END IF;`

END;

El comportamiento del bloque resulta bastante evidente.

4.1. Fundamentos del Lenguaje PL-SQL -ORACLE



ESTRUCTURA DE LOS BLOQUES DE PROGRAMA

ESTRUCTURAS DE CONTROL PL/SQL

- En el ejemplo anterior cada secuencia de orden consta de una sola orden procedimental, sin embargo pueden haber tantas orden (procedimientos o SQL) como sea necesario.

DECLARE

```
v_NumberSeats rooms.number_seats%TYPE;  
v_Comment VARCHAR(35);
```

BEGIN

```
/* Extrae el numero de asiento de la habitación, cuyo identificador es 99999 y almacena el resultado en  
v_NumberSeats. */
```

```
SELECT number_seats
```

```
INTO v_NumberSeats
```

```
FROM rooms
```

```
WHERE room_id = 99999;
```

```
IF v_NumberSeats < 50 THEN
```

```
    v_comment := ' Fairly small';
```

```
    INSERT INTO temp_table (char_col) VALUES ('Nice and cozy');
```

```
ELSIF v_NumberSeats < 100 THEN
```

```
    v_comment := ' A litle bigger';
```

```
    INSERT INTO temp_table (char_col) VALUES ('Some breathing room ');
```

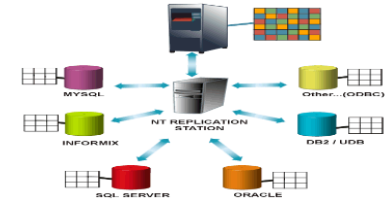
```
ELSE
```

```
    v_comment := ' Lots of room';
```

```
END IF;
```

END;

4.1. Fundamentos del Lenguaje PL-SQL -ORACLE



ESTRUCTURA DE LOS BLOQUES DE PROGRAMA

ESTRUCTURAS DE CONTROL PL/SQL

- Condiciones Nulas

Una secuencia de ordenes dentro de un orden IF-THEN-ELSE se ejecuta solo si su condición asociada es verdadera(toma el valor TRUE). Si las condición toma el valor FALSE o NULL, la secuencia de ordenes no se ejecuta. Ejemplos:

/* Bloque 1 */

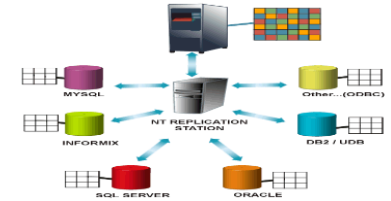
```
DECLARE
  v_Number1 NUMBER;
  v_Number2 NUMBER;
  v_Result  VARCHAR(7);
BEGIN
  ...
  IF v_Number1 < v_Number2 THEN
    v_Result := 'Yes';
  ELSE
    v_Result := 'No';
  END IF;
END;
```

/* Bloque 2 */

```
DECLARE
  v_Number1 NUMBER;
  v_Number2 NUMBER;
  v_Result  VARCHAR(7);
BEGIN
  ...
  IF v_Number1 >= v_Number2 THEN
    v_Result := 'No';
  ELSE
    v_Result := 'Yes';
  END IF;
END;
```

Si **v_Number1 = 3** y **v_Number2 = 7**. Se comportan ambos bloques de la misma forma?

4.1. Fundamentos del Lenguaje PL-SQL -ORACLE



ESTRUCTURA DE LOS BLOQUES DE PROGRAMA

ESTRUCTURAS DE CONTROL PL/SQL

- Condiciones Nulas

A los bloques mostrados se le puede añadir una comprobación adicional de nulidad:

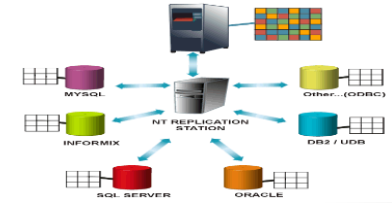
/* Bloque 1 */

```
DECLARE
  v_Number1 NUMBER;
  v_Number2 NUMBER;
  v_Result  VARCHAR(7);
BEGIN
  ...
  IF v_Number1 IS NULL OR
     v_Number2 IS NULL THEN
    v_Result := 'Unknown';
  ELSIF v_Number1 < v_Number2 THEN
    v_Result := 'Yes';
  ELSE
    v_Result := 'No';
  END IF;
END;
```

/* Bloque 2 */

```
DECLARE
  v_Number1 NUMBER;
  v_Number2 NUMBER;
  v_Result  VARCHAR(7);
BEGIN
  ...
  IF v_Number1 IS NULL OR
     v_Number2 IS NULL THEN
    v_Result := 'Unknown';
  ELSIF v_Number1 >= v_Number2 THEN
    v_Result := 'No';
  ELSE
    v_Result := 'Yes';
  END IF;
END;
```

4.1. Fundamentos del Lenguaje PL-SQL -ORACLE



ESTRUCTURA DE LOS BLOQUES DE PROGRAMA

ESTRUCTURAS DE CONTROL PL/SQL

- BUCLES

PL/SQL permite ejecutar ordenes en forma repetida, utilizando los bucles. Existen 4 tipos de bucles: *bucles simples*, *bucles WHILE*, *bucles FOR numéricos*, y *bucles FOR de cursor*.

Bucles Simples

Son el tipo de bucle más básico. Su sintaxis es:

LOOP

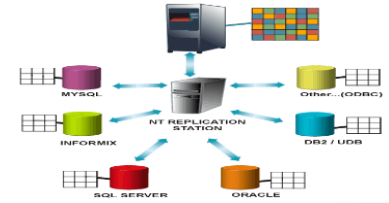
secuencia_de_órdenes;

END LOOP;

La **secuencia_de_órdenes** se ejecutara indefinidamente, puesto que el bucle no tiene ninguna condición de parada. Si embargo, podemos añadir una condición mediante la orden EXIT cuya sintaxis es:

EXIT [WHEN *condición*];

4.1. Fundamentos del Lenguaje PL-SQL -ORACLE



ESTRUCTURA DE LOS BLOQUES DE PROGRAMA

ESTRUCTURAS DE CONTROL PL/SQL

- BUCLES SIMPLE -LOOP

El siguiente bloque, del ejemplo inserta 50 filas en la tabla **temp_table**.

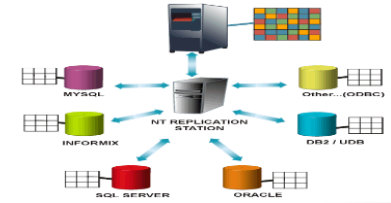
Bloque1

```
DECLARE
  v_counter BINARY_INTEGER := 1;
BEGIN
  LOOP
    -- Insertar una fila en temp__table con el valor actual del
    -- contador del bucle.
    INSERT INTO temp_table
    VALUES (v_counter, 'Loop Index');
    v_counter := v_counter + 1;
    -- Condición de salida – Cuando el contador
    -- del bucle sea > 50 se saldrá del bucle
    IF v_counter > 50 THEN
      EXIT;
    END IF;
  END LOOP;
END;
```

Bloque2-Se comporta igual al anterior

```
DECLARE
  v_counter BINARY_INTEGER := 1;
BEGIN
  LOOP
    -- Insertar una fila en temp__table con el valor actual del
    -- contador del bucle.
    INSERT INTO temp_table
    VALUES (v_counter, 'Loop Index');
    v_counter := v_counter + 1;
    -- Condición de salida – Cuando el contador
    -- del bucle sea > 50 se saldrá del bucle
    EXIT WHEN v_counter > 50;
  END LOOP;
END;
```

4.1. Fundamentos del Lenguaje PL-SQL -ORACLE



ESTRUCTURA DE LOS BLOQUES DE PROGRAMA

ESTRUCTURAS DE CONTROL PL/SQL

- BUCLES -WHILE

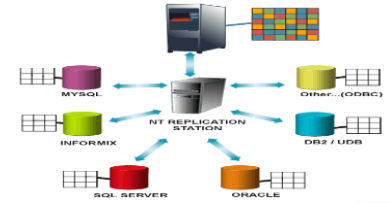
La sintaxis de un bucle WHILE es.

```
WHILE condición LOOP
    secuencia_de_órdenes;
END LOOP;
```

La condición se evalúa antes de cada iteración del bucle. Si es verdadera se ejecuta la *secuencia_de_órdenes*. Si la condición es falsa o nula, el bucle termina y el control se transfiere a lo que esta a continuación de la orden **END LOOP**. Ejemplo:

```
DECLARE
    v_counter BINARY_INTEGER := 1;
BEGIN
    -- comprueba el contador del bucle antes de cada iteración
    -- para asegurarse que todavía es menor a 50.
    WHILE v_counter <= 50 LOOP
        INSERT INTO temp_table
        VALUES (v_counter, 'Loop Index');
        v_counter := v_counter + 1;
    END LOOP;
END;
```

4.1. Fundamentos del Lenguaje PL-SQL -ORACLE



ESTRUCTURA DE LOS BLOQUES DE PROGRAMA

ESTRUCTURAS DE CONTROL PL/SQL

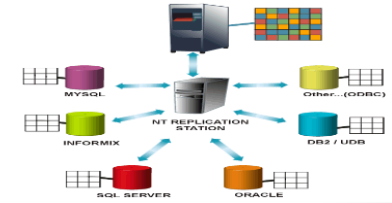
- BUCLES -WHILE

Se puede usar las ordenes **EXIT** o **EXIT WHEN** dentro un bucle **WHILE** para salir de forma prematura. Tenga presente que si la condición del bucle no toma el valor **TRUE** la primera vez que se comprueba, el bucle no llega a ejecutarse.

Ejemplo :

```
DECLARE
    v_counter BINARY_INTEGER;
BEGIN
    -- Esta condicion se evaluara como NULL, ya que
    v_Counter
    -- se inicializa con el valor predeterminado NULL.
    WHILE v_counter <= 50 LOOP
        INSERT INTO temp_table
        VALUES (v_counter, 'Loop Index');
        v_counter := v_counter + 1;
    END LOOP;
END;
```


4.1. Fundamentos del Lenguaje PL-SQL -ORACLE



ESTRUCTURA DE LOS BLOQUES DE PROGRAMA

ESTRUCTURAS DE CONTROL PL/SQL

- BUCLES -FOR numéricos

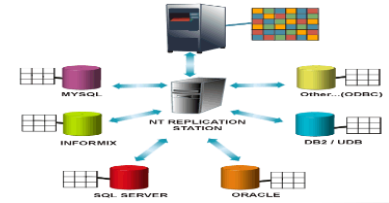
El numero de interacciones de los **bucles simples** y de los bucles **WHILE** no se conoce de antemano, sino que depende de la condición del bucle. Los bucles **FOR numéricos** por lo contrario, tienen el numero de iteración definido. Su sintaxis es:

```
FOR contador_bucle IN [REVERSE] limite_inferior... limite_superior LOOP  
    secuencia_de_órdenes;  
END LOOP;
```

Donde *contador_bucle* es la variable de indice declarada de modo explicito, *limite_inferior* y *limite_superior* especifican el numero de iteraciones y *secuencia_de_órdenes* es el contenido del bucle.

Los limites del bucle solo se evalúan una sola vez. Estos valores determinan el numero total de interacciones, en las que el *contador_bucle* varia entre los valores del *limite_inferior* y *limite_superior* incrementándose en una unidad a la vez, hasta que el bucle se completa.

4.1. Fundamentos del Lenguaje PL-SQL -ORACLE



ESTRUCTURA DE LOS BLOQUES DE PROGRAMA

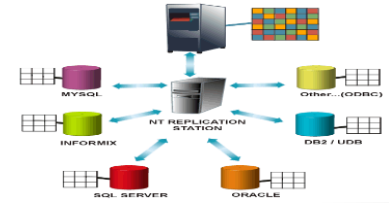
ESTRUCTURAS DE CONTROL PL/SQL

- BUCLES -FOR *numericos*

En el ejemplo del bucle utilizando el bucle FOR:

```
DECLARE
  v_counter BINARY_INTEGER;
BEGIN
  -- Esta condicion se evaluara como NULL, ya que
  v_Counter
  -- se inicializa con el valor predeterminado NULL.
  WHILE v_counter <= 50 LOOP
    INSERT INTO temp_table
    VALUES (v_counter, 'Loop Index');
    v_counter := v_counter + 1;
  END LOOP;
END;
```

4.1. Fundamentos del Lenguaje PL-SQL -ORACLE



ESTRUCTURA DE LOS BLOQUES DE PROGRAMA

ESTRUCTURAS DE CONTROL PL/SQL

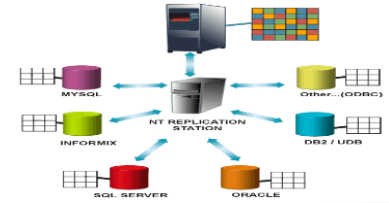
- BUCLES -FOR *numericos*

En el ejemplo del bucle utilizando el bucle FOR:

```
BEGIN
    FOR v_counter IN 1..50 LOOP
        INSERT INTO temp_table
            VALUES (v_counter, 'Loop Index');
    END LOOP;
END;
```

Reglas de ámbitos. El índice de un bucle **FOR** se declara implícitamente como un **BINARY_INTEGER**, no siendo necesario declararlo antes del bucle. Si lo declara, el índice del bucle ocultará la declaración exterior al bucle, de la misma forma que en una declaración de variable en un bloque puede ocultar una declaración en un bloque externo ese.

4.1. Fundamentos del Lenguaje PL-SQL -ORACLE



ESTRUCTURA DE LOS BLOQUES DE PROGRAMA

ESTRUCTURAS DE CONTROL PL/SQL

- BUCLES -FOR *numericos*

En el ejemplo del bucle utilizando el bucle FOR:

DECLARE

v_counter **BINARY_INTEGER** := 7;

BEGIN

-- Inserta el valor 7 en la tabla temp_table.

INSERT INTO temp_table (num_col) **VALUES** (v_counter);

-- Este bucle declara de nuevo v_counter como BINARY_INTEGER, lo

-- que anula la declaración de NUMBER de v_counter

FOR v_counter **IN** 20.. 30 **LOOP**

--Dentro del bucle, el rango de v_counter es de 20 a 30

INSERT INTO temp_table (num_col) **VALUES** (v_counter);

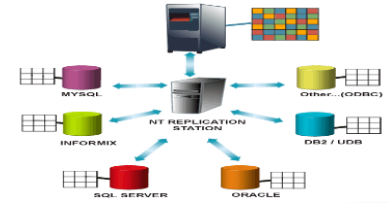
END LOOP;

--- Inserta otro 7 en la tabla temp_table

INSERT INTO temp_table (num_col) **VALUES** (v_counter);

END;

4.1. Fundamentos del Lenguaje PL-SQL -ORACLE



ESTRUCTURA DE LOS BLOQUES DE PROGRAMA

ESTRUCTURAS DE CONTROL PL/SQL

- BUCLES -FOR *numericos*

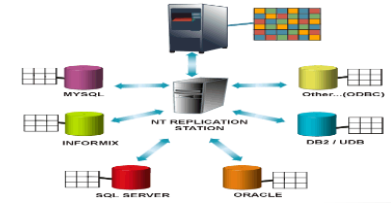
UTILIZACION de REVERSE Si se incluye la palabra clave REVERSE en el bucle FOR, el indice del bucle realizara las iteraciones desde el limite superior al limite inferior. Su sintaxis es la siguiente, con el ejemplo correspondiente

```
BEGIN
    FOR v_counter IN REVERSE 10..50 LOOP
        -- v_counter se iniciara en 50 y se decrementara en una
        -- unidad cada vez que se ejecute el bucle
        NULL;
    END LOOP;
END;
```

RANGO DE LOS BUCLES los limites inferiores o superiores no tienen porque ser literales numéricas, sino que puede ser cualquier expresión que puede ser convertida en un valor numérico. Ejemplo

```
DECLARE
    v_LowValue  NUMBER := 10;
    v_HighValue NUMBER := 40;
BEGIN
    FOR v_counter IN v_LowValue .. V_HighValue LOOP
        INSERT INTO temp_table
            VALUES (v_counter, 'Dinamically specified loop range');
    END LOOP;
END;
```

4.1. Fundamentos del Lenguaje PL-SQL -ORACLE



ESTRUCTURA DE LOS BLOQUES DE PROGRAMA

ESTRUCTURAS DE CONTROL PL/SQL

- ORDENES GOTO y etiquetas

PS/SQL también disponen de una orden de salto GOTO. La sintaxis es la siguiente

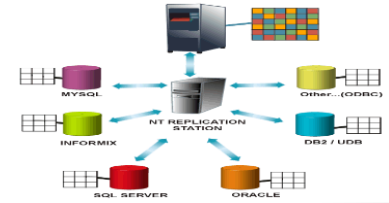
GOTO *etiqueta*

Donde *etiqueta* es una etiqueta definida en el bloque PL/SQL. Las etiquetas se encierran entre corchetes angulares dobles. Cuando se evalúa una orden **GOTO**, el control pasa inmediatamente a la identificadas por la etiqueta. Ejemplo

```
DECLARE
  v_counter BINARY_INTEGER := 1;
BEGIN
  LOOP
    INSERT INTO temp_table
    VALUES (v_counter, 'Loop Count');
    v_counter := v_counter + 1;
    IF v_counter > 50 THEN
      GOTO 1_EndOfLoop;
    END IF;
  END LOOP;

  << 1_EndOfLoop>>
  INSERT INTO temp_table (char_col) VALUES ( 'Done!');
END;
```

4.1. Fundamentos del Lenguaje PL-SQL -ORACLE



ESTRUCTURA DE LOS BLOQUES DE PROGRAMA

ESTRUCTURAS DE CONTROL PL/SQL

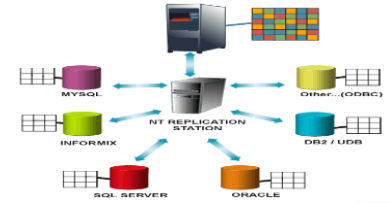
- Restriccion en el uso de la orden GOTO

PS/SQL impone una serie de restricciones para el uso de la orden GOTO. Es ilegal realizar un salto al interior de un bloque interno, de un bucle o de una orden IF, e incluso es prohibitivo hacerlo dentro del bloque de la excepciones. Ejemplo:

```
BEGIN
    GOTO 1_InnerBlock; -- Ilegal, no se puede saltar a un bloque interno
BEGIN
    .....
    << 1_InnerBlock>>
END;
    GOTO 1_Insidelf; -- Ilegal, no se puede saltar al interior de una orden IF
    IF X > 3 THEN
        << 1_Insidelf>>
        INSERT INTO.....;
    END IF;
END;
```

Si este tipo de salto fuera legal, entonces las ordenes situadas dentro de la orden IF podría ser ejecutada incluso si la condición de la orden IF no fuera cierta.

4.1. Fundamentos del Lenguaje PL-SQL -ORACLE



ESTRUCTURA DE LOS BLOQUES DE PROGRAMA

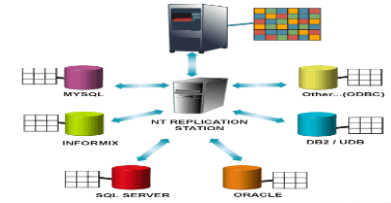
ESTRUCTURAS DE CONTROL PL/SQL

- Restriccion en el uso de la orden GOTO

Es ilegal saltar desde la rutina de manejo de excepciones de vuelta al bloque actual

```
DECLARE
  v_Room    room%ROWTYPE;
BEGIN
  -- Recupera una fila de la tabla room
  SELECT *
  INTO v_Room
  WHERE rowid = 1;
  <<1_Insert>>
  INSERT INTO temp_table (char_col)
  VALUES ('Found a row');
EXCEPTION
  .  WHERE NO_DATA_FOUND THEN
      GOTO 1_insert; -- Ilegal no se puede saltar al interior del bloque
                        -- actual
END;
```


4.1. Fundamentos del Lenguaje PL-SQL -ORACLE



ESTRUCTURA DE LOS BLOQUES DE PROGRAMA

ESTRUCTURAS DE CONTROL PL/SQL

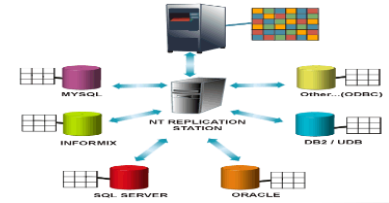
- Etiqueta de los Bucles

También los propios bucles podrían ser etiquetado. Si lo hacemos así, puede emplearse la etiqueta en la orden EXIT para indicar el bucle del que hay que salir.

```
BEGIN
  <<1_Outer>>
  FOR v_OuterIndex IN 1..50 LOOP
    ... <<1_Inner>>
    FOR v_InnerIndex IN 2.. 20 LOOP
      ...
      IF v_OuterIndex > 40 THEN
        EXIT 1_Outer;  -- Salida de ambos bucles
      END IF;
    END LOOP 1_Inner;
  END LOOP 1_Outer;
END;
```

Si se etiquetan los bucles, el nombre de las etiquetas puede ser incluido opcionalmente después de la orden **END LOOP**.

4.1. Fundamentos del Lenguaje PL-SQL -ORACLE



ESTRUCTURA DE LOS BLOQUES DE PROGRAMA

ESTRUCTURAS DE CONTROL PL/SQL

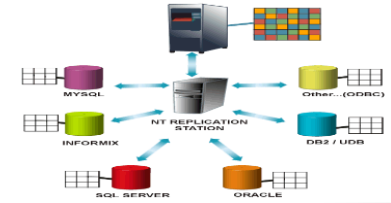
- Etiqueta de los Bucles

También los propios bucles podrían ser etiquetado. Si lo hacemos así, puede emplearse la etiqueta en la orden EXIT para indicar el bucle del que hay que salir.

```
BEGIN
  <<1_Outer>>
  FOR v_OuterIndex IN 1..50 LOOP
    ... << 1_Inner>>
    FOR v_InnerIndex IN 2.. 20 LOOP
      ...
      IF v_OuterIndex > 40 THEN
        EXIT 1_Outer; -- Salida de ambos bucles
      END IF;
    END LOOP 1_Inner;
  END LOOP 1_Outer;
END;
```

Si se etiquetan los bucles, el nombre de las etiquetas puede ser incluido opcionalmente después de la orden **END LOOP**.

4.1. Fundamentos del Lenguaje PL-SQL -ORACLE



ESTRUCTURA DE LOS BLOQUES DE PROGRAMA

ESTRUCTURAS DE CONTROL PL/SQL

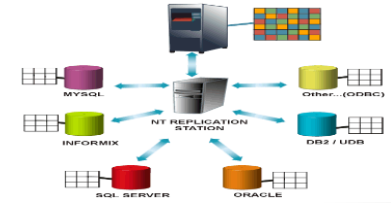
- La Orden NULL

En algunos casos explícitamente se quiere indicar que no se realice ninguna acción. Esto se puede realizar mediante la orden NULL, que es un orden que no tiene efecto alguno.

```
DECLARE
    v_TempVar NUMBER := 7;

BEGIN
    IF v_Tempvar < 5 THEN
        INSERT INTO temp_table (char_col)
        VALUES ('Too Small');
    ELSIF v_Tempvar < 10 THEN
        INSERT INTO temp_table (char_col)
        VALUES ('Just Right');
    ELSE
        NULL; -- No hace nada
    END IF;
END;
```

4.1. Fundamentos del Lenguaje PL-SQL -ORACLE

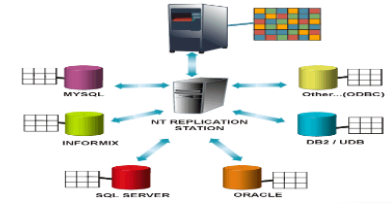


ESTRUCTURA DE LOS BLOQUES DE PROGRAMA

GUIA DE ESTILO DE PL/SQL

- No existen reglas absolutas referente al estilo de escritura de un programa. El estilo de programación incluye conceptos como los *nombres de las variables, utilización de las letras mayúsculas, espacios en blanco, y el uso de los comentarios*.
- Estos no son aspectos que afecten necesariamente a la ejecución de un programa; si reescribimos un programa con estilo distinto, el programa continuara haciendo lo mismo.
- Un programa escrito con un buen estilo será mucho mas fácil de entender y de mantener que un programa con un estilo pobre.
- Un buen estilo de programación que se tardara menos en entender lo que hace el programa, cuando se le ve por primera vez. Asimismo, ayudara al propio desarrollador a entender lo que el programa hace, tanto a medida que lo escribe como cuando lo vuelve a ver un mes mas tarde.

4.1. Fundamentos del Lenguaje PL-SQL -ORACLE



ESTRUCTURA DE LOS BLOQUES DE PROGRAMA

GUIA DE ESTILO DE PL/SQL

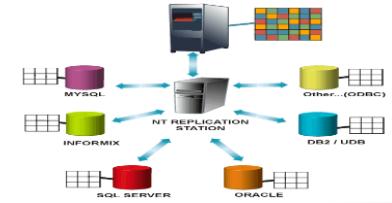
Observe los dos bloques siguientes. Cual de ellos es mas fácil de entender?

```
declare
  x number;
  y number;
begin if x < 10 then y := 7 ; else y := 13; end if; end;
```

```
DECLARE
  v_Test NUMBER;    -- Variable que se examinara
  v_Result NUMBER;  -- Variable para almacenar resultado
BEGIN
  -- Examina v_Test y asigna 7 a v_Result si v_Test < 10
  IF v_Test < 10 THEN
    v_Result := 7;
  ELSE
    v_Result := 3;
  END IF;
END;
```

Ambos programas realizan la misma función. Sin embargo, el flujo del programa resulta mas fácil de comprender en el segundo caso.

4.1. Fundamentos del Lenguaje PL-SQL -ORACLE



ESTRUCTURA DE LOS BLOQUES DE PROGRAMA

GUIA DE ESTILO DE PL/SQL

Estilo de Comentarios

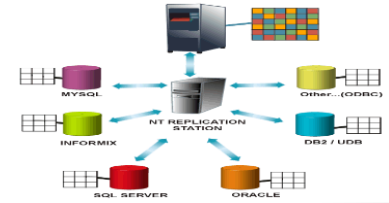
Se usan como mecanismo para informar al lector sobre cual es el propósito y como funciona el programa. Se pueden incluyen comentarios en:

- Al inicio de cada bloque y/o procedimientos. Estos comentarios deberían explicar que es o que hace el bloque o el procedimiento. De forma especial para los procedimientos, es importante enumerar las variables o parámetros que el procedimiento leerá (entrada) y escribirá (salida). También es una idea importante enumerar las tablas de la base de datos a las que accede.
- Junta a cada declaración de variable, para describir el uso que se le va a dar. A menudo es suficiente con usar comentarios de una sola línea.

`v_SSN CHAR(11); -- Numero de seguro social`

- Antes de cada una de las secciones principales del bloque. No es necesario colocar comentarios a cada orden, pero un comentario que explique el propósito del siguiente conjunto de ordenes resultara útil.

4.1. Fundamentos del Lenguaje PL-SQL -ORACLE



ESTRUCTURA DE LOS BLOQUES DE PROGRAMA

GUIA DE ESTILO DE PL/SQL

Estilo de Comentarios **continuacion...**

Se usan como mecanismo para informar al lector sobre cual es el propósito y como funciona el programa. Se pueden incluyen comentarios en:

- En los algoritmos utilizados pueden ser obvio a partir del propio código así que es mejor describir el propósito del algoritmo y para que usara los resultado, en lugar de los detalles del método.
- Los comentarios deben ser significativos y no volver a expresar lo que el propio codigo PL/SQL ya expresa. Ejemplo

DECLARE

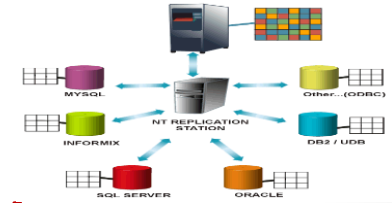
v_Temp NUMBER := 0; --Asigna 0 a v_Temp

- Este otro comentario seria mejor porque nos dice cual es el propósito de la variable **V_Temp**:

DECLARE

v_Temp NUMBER := 0; --Variable temporal usada en el bucle principal

4.1. Fundamentos del Lenguaje PL-SQL -ORACLE



ESTRUCTURA DE LOS BLOQUES DE PROGRAMA

GUIA DE ESTILO DE PL/SQL

Estilo de los nombres de variables

La clave de los nombres de variables consiste en hacerlos descriptivos. La declaracion:

```
x number;
```

No nos dice nada sobre el propósito de x. Sin embargo,

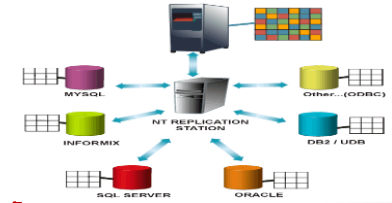
```
v_StudentID NUMBER(5);
```

Nos dice que esta variable será probablemente usada para almacenar el numero de identificación de un estudiante, incluso aunque no pongamos un comentario explicativo al lado de la declaración. Siempre debemos tomar en cuenta el tamaño del identificador PL/SQL.

El nombre de una variable también nos puede informar acerca de su uso. Se puede usar para esto un código de una letra, separado por un carácter de subrayado del resto de las variables Ejemplos:

v_NombreVariable	Variable del programa
e_NombreExcepcion	Excepción definida por el usuario
t_NombreTipo	Tipo definido por el usuario
p_NombreParametro	Parámetro de un procedimiento o función
c_ValorConstante	Variable restringida mediante la clausula CONSTANT

4.1. Fundamentos del Lenguaje PL-SQL -ORACLE



ESTRUCTURA DE LOS BLOQUES DE PROGRAMA

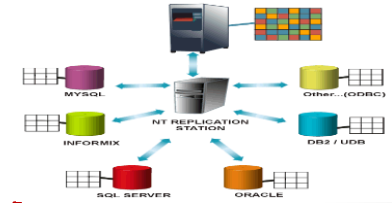
GUIA DE ESTILO DE PL/SQL

Estilo de uso de las mayúsculas

PL/SQL no diferencia entre mayúsculas y minúsculas. Si embargo, el uso apropiado de las mayúsculas y minúsculas incrementa sobremanera la legibilidad de una programa. Algunas reglas de uso:

- Las palabras reservadas se escriben en mayúsculas (por ejemplo, **BEGIN**, **DECLARE** o **ELSIF**)
- Las funciones predefinidas se escriben en mayúsculas (**SUBSTR**, **COUNT**, **TO_CHAR**)
- Los tipos predefinidos se escriben en mayúsculas (**NUMBER(7,2)**, **BOOLEAN**, **DATE**)
- Las palabras claves de SQL se escriben en mayúsculas (**SELECT**, **INTO**, **UP**, **DATE**, **WHERE**)
- Los objetos de la base de datos se escriben en minúsculas (**log_table**, **classes**, **students**)
- Los nombres de variables se escriben con una mezcla de mayúsculas y minúsculas, poniendo en mayúsculas la primera letra de cada palabra componente (**v_HireDate**, **e_TooManyStudents**, **t_StudentsRecordType**)

4.1. Fundamentos del Lenguaje PL-SQL -ORACLE



ESTRUCTURA DE LOS BLOQUES DE PROGRAMA

GUIA DE ESTILO DE PL/SQL

Estilo de Indentacion

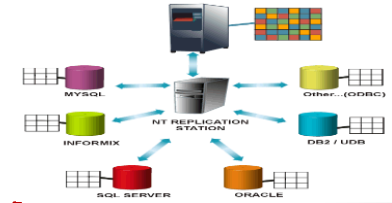
Una de las cosas que menos cuesta es usar los *espacios en blanco* (retorno de carro, caracteres de espacio y tabulaciones) que pueden tener un gran efecto sobre la legibilidad del programa.

Compare las dos estructuras IF-THEN-ELSE indentadas siguientes:

```
IF x < y THEN IF z IS NULL THEN x:= 3; ELSE x := 2 END IF;
ELSE x := 4; END IF;
```

```
IF x < y THEN
  IF z IS NULL THEN
    x := 3;
  ELSE
    x := 2;
  END IF;
ELSE
  x := 4;
END IF;
```

4.1. Fundamentos del Lenguaje PL-SQL -ORACLE



ESTRUCTURA DE LOS BLOQUES DE PROGRAMA

GUIA DE ESTILO DE PL/SQL

Estilo de Indentación

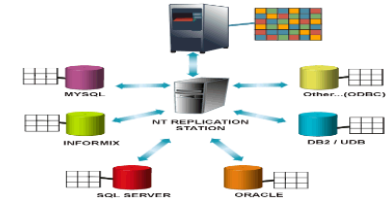
Una de las cosas que menos cuesta es usar los *espacios en blanco* (retorno de carro, caracteres de espacio y tabulaciones) que pueden tener un gran efecto sobre la legibilidad del programa.

Compare las dos estructuras IF-THEN-ELSE indentadas siguientes:

```
IF x < y THEN IF z IS NULL THEN x:= 3; ELSE x := 2 END IF;  
ELSE x := 4; END IF;
```

```
IF x < y THEN  
  IF z IS NULL THEN  
    x := 3;  
  ELSE  
    x := 2;  
  END IF;  
ELSE  
  x := 4;  
END IF;
```

4.1. Fundamentos del Lenguaje PL-SQL -ORACLE



ESTRUCTURA DE LOS BLOQUES DE PROGRAMA

GUIA DE ESTILO DE PL/SQL

Estilo de Indentación continuación

Generalmente para lograr este estilo se suele indentar cada línea dentro de un bloque con dos espacios. Se acostumbra a indentar el contenido de los bloques con respecto a la palabra clave **DECLARE.....END**, y también los bucles y las ordenes **IF-THEN-ELSE**. También dentro de las ordenes SQL que ocupan más de una línea por ejemplo:

```
SELECT id, first_name, last_name
      INTO v_StudentID, v_FirstName, v_LastName
      FROM students
      WHERE id = 10002;
```