



# Estructuras de Iteración en Bloques PL/SQL

**MDY3131**

# Experiencia de Aprendizaje y Competencia Asociada



ESCUELA DE  
INFORMÁTICA Y  
TELECOMUNICACIONES

Experiencia	Nombre	Unidad de Competencia Especialidad – Nivel de la Competencia de Empleabilidad
Nº 1	Construyendo Bloques Anónimos PL/SQL Simples	Desarrolla operaciones sobre la base de datos que permitan administrar los objetos de la misma de acuerdo a requerimientos de usuario y buenas prácticas de la industria.
		Resolución de Problemas (N1)

# Objetivos de la Clase

- Qué es una Estructura de Iteración o LOOP.
- Tipos de LOOP que se pueden usar en bloques PL/SQL.
- Cómo construir Estructuras de Iteración usando LOOP básico.
- Cómo construir Estructuras de Iteración usando WHILE LOOP
- Cómo construir Estructuras de Iteración usando FOR LOOP.
- Cómo trabajar con LOOPS Anidados.

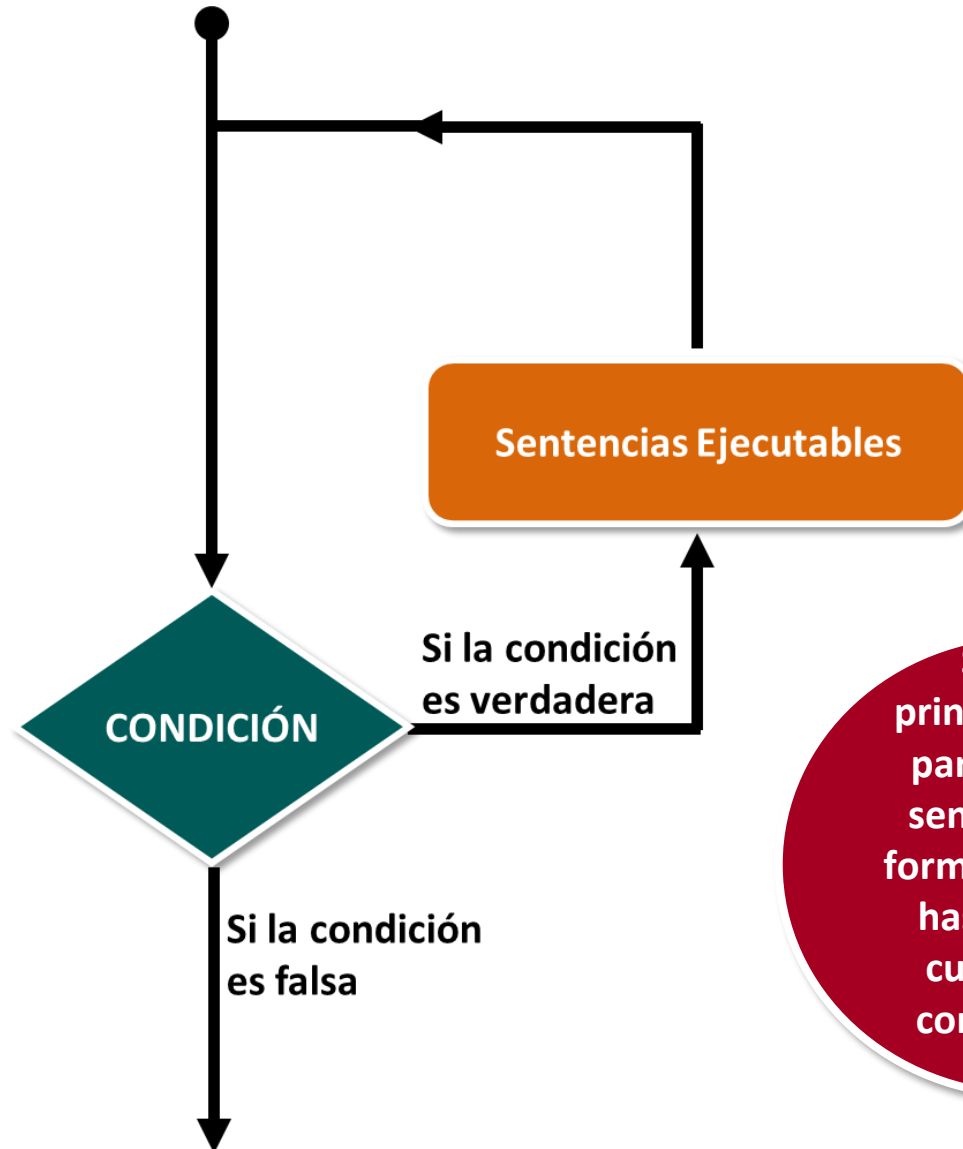


## LOOPS en bloques PL/SQL

1. Los bloques PL/SQL de los ejemplos usan tablas del esquema HR de la Base de Datos Oracle y algunas tablas nuevas. Por esta razón, en las sentencias DML (INSERT, UPDATE y DELETE) de los bloques se usan copias de esas tablas del esquema HR para no realizar los cambios de datos en las tablas originales.
2. De acuerdo con esto, antes de que Ud. ejecute los ejemplos debe crear las tablas necesarias con el archivo [script\\_crea\\_tablas\\_ejemplos.sql](#)

# LOOP o Bucle Iterativo

Repiten una  
sentencia o  
secuencia de  
sentencias un  
número específico  
de veces



Se usan  
principalmente  
para ejecutar  
sentencias en  
forma repetitiva  
hasta que se  
cumpla una  
condición de  
salida



# Loops o Bucles Iterativos

**LOOP Simple:** en cada iteración, la(s) sentencia(s) se ejecuta(n) y luego el control se reanuda en la parte superior del ciclo

**WHILE LOOP:** repite sentencia(s) mientras una condición determinada es verdadera

**FOR LOOP:** el código para ejecutar la(s) sentencia(s) es más abreviado

# LOOP Simple

Es la forma más simple de las sentencias LOOP y puede contener una serie de sentencias entre las palabras LOOP y END LOOP

Permite la ejecución de sus sentencias a lo menos una vez, incluso si la condición de salida ya se cumple al entrar en el loop

Se debe indicar en forma explícita una condición de salida del loop, de lo contrario sería infinito

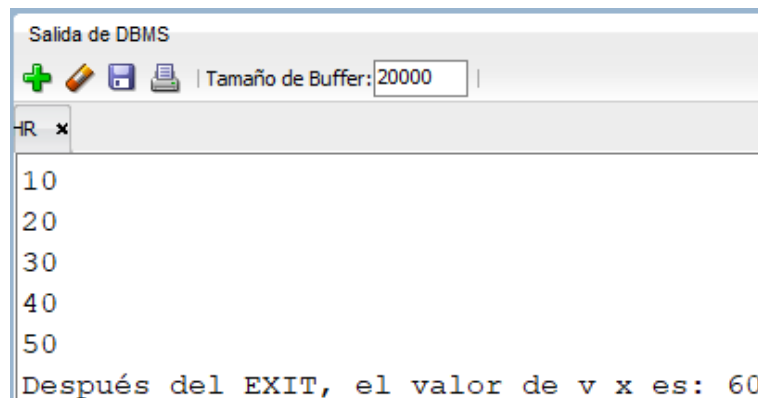
Puede contener múltiples sentencias de salida del loop, pero se recomienda tener solo un punto de salida

- Sintaxis:

```
LOOP  
  sentenciaN;  
EXIT [WHEN condición];  
END LOOP;
```

- Ejemplo:

```
DECLARE
  v_x number := 10;
BEGIN
  LOOP
    dbms_output.put_line(v_x);
    v_x := v_x + 10;
    EXIT WHEN v_x > 50;
  END LOOP;
  -- Después de la salida del LOOP, la ejecución del bloque PL/SQL continúa aquí
  DBMS_OUTPUT.PUT_LINE('Después del EXIT, el valor de v_x es: ' || v_x);
END;
```



```
Salida de DBMS
+R x
10
20
30
40
50
Después del EXIT, el valor de v_x es: 60
```



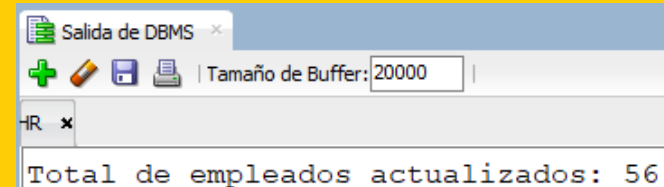
- Ejemplo:

```
DECLARE
v_countryid  locations.country_id%TYPE := 'CA';
v_loc_id     locations.location_id%TYPE;
v_contador   NUMBER(2) := 1;
v_new_city   locations.city%TYPE := 'Montreal';
BEGIN
  SELECT MAX(location_id)
    INTO v_loc_id
  FROM locations
 WHERE country_id = v_countryid;
  LOOP
    INSERT INTO ubicaciones(location_id, city, country_id)
      VALUES((v_loc_id + v_contador), v_new_city, v_countryid);
    v_contador := v_contador + 1;
    EXIT WHEN v_contador > 3;
  END LOOP;
END;
```

# LOOP Simple

- Ejemplo:

```
VAR b_porc_aumento NUMBER
EXEC :b_porc_aumento:=1.25
DECLARE
v_sal_prom NUMBER(7);
v_id_min NUMBER(3);
v_id_max NUMBER(3);
v_tot_emp_act NUMBER(3):=0;
BEGIN
  SELECT ROUND(AVG(salary)),MIN(employee_id), MAX(employee_id)
    INTO v_sal_prom,v_id_min,v_id_max
    FROM employees;
  LOOP
    UPDATE empleados
      SET salary=ROUND(salary*:b_porc_aumento)
    WHERE employee_id=v_id_min
      AND salary < v_sal_prom;
    IF SQL%ROWCOUNT > 0 THEN
      v_tot_emp_act:=v_tot_emp_act+1;
    END IF;
    v_id_min:=v_id_min+1;
    EXIT WHEN v_id_min > v_id_max;
  END LOOP;
  DBMS_OUTPUT.PUT_LINE('Total de empleados actualizados: ' || v_tot_emp_act);
ROLLBACK;
END;
```



# WHILE LOOP

Permite que las sentencias se repiten mientras la condición del loop sea verdadera (TRUE)

La condición es evaluada al comienzo de cada iteración. El loop termina cuando la condición es FALSE o NULL

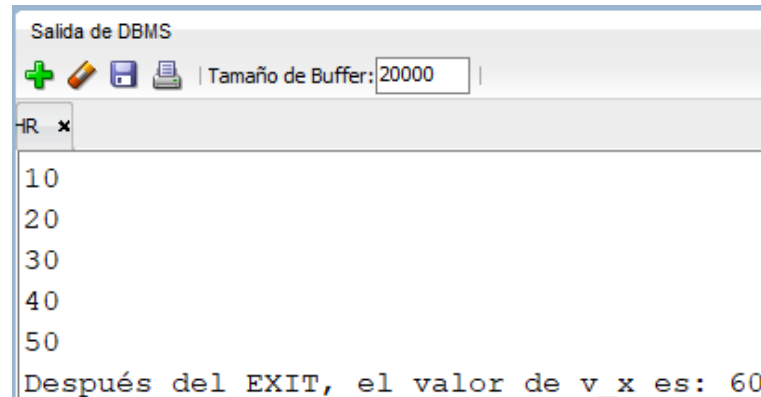
Si la variable de la condición de entrada no se inicializa, el loop no se ejecutará ni una sola vez

- Sintaxis:

```
WHILE condición LOOP  
    sentenciaN;  
END LOOP;
```

- Ejemplo:

```
DECLARE
  v_x number := 10;
BEGIN
  WHILE v_x <= 50 LOOP
    dbms_output.put_line(v_x);
    v_x := v_x + 10;
  END LOOP;
  -- Después de la salida del LOOP, la ejecución del bloque PL/SQL continúa aquí
  DBMS_OUTPUT.PUT_LINE('Después del EXIT, el valor de v_x es: ' || v_x);
END;
```



```
Salida de DBMS
+ | Tamaño de Buffer: 20000 |
-- x
10
20
30
40
50
Después del EXIT, el valor de v_x es: 60
```

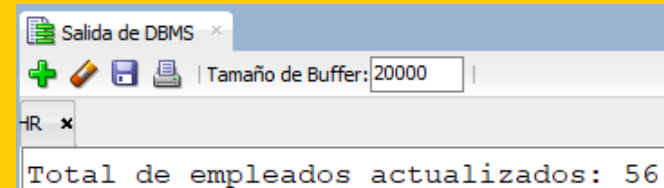
- Ejemplo:

```
DECLARE
v_countryid  locations.country_id%TYPE := 'CA';
v_loc_id     locations.location_id%TYPE;
v_new_city   locations.city%TYPE := 'Montreal';
v_contador   NUMBER := 1;
BEGIN
  SELECT MAX(location_id)
    INTO v_loc_id
  FROM locations
 WHERE country_id = v_countryid;
  WHILE v_contador <= 3 LOOP
    INSERT INTO ubicaciones(location_id, city, country_id)
      VALUES((v_loc_id + v_contador), v_new_city, v_countryid);
    v_contador:= v_contador + 1;
  END LOOP;
END;
```

# WHILE LOOP

- Ejemplo:

```
VAR b_porc_aumento NUMBER
EXEC :b_porc_aumento:=1.25
DECLARE
v_sal_prom NUMBER(7);
v_id_min NUMBER(3);
v_id_max NUMBER(3);
v_tot_emp_act NUMBER(3):=0;
BEGIN
SELECT ROUND(AVG(salary)),MIN(employee_id), MAX(employee_id)
  INTO v_sal_prom,v_id_min,v_id_max
  FROM employees;
WHILE v_id_max > v_id_min LOOP
  UPDATE empleados
    SET salary=ROUND(salary*:b_porc_aumento)
  WHERE employee_id=v_id_min
    AND salary < v_sal_prom;
  IF SQL%ROWCOUNT > 0 THEN
    v_tot_emp_act:=v_tot_emp_act+1;
  END IF;
  v_id_min:=v_id_min+1;
END LOOP;
DBMS_OUTPUT.PUT_LINE('Total de empleados actualizados: ' || v_tot_emp_act);
ROLLBACK;
END;
```





Las iteraciones se efectúan un número finito y conocido de veces

El índice es declarado implícitamente y es obligatorio indicar su límite inferior y superior

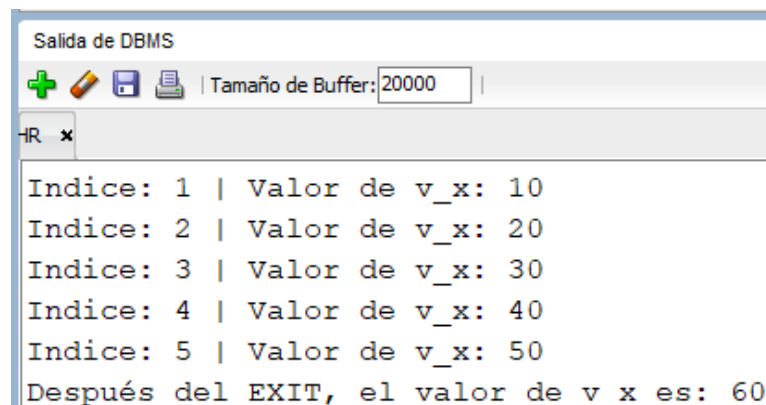
El índice declarado sólo se puede usar en las sentencias declaradas dentro del loop

- Sintaxis:

```
FOR contador IN [REVERSE]  
    límite_inferior .. límite_superior LOOP  
    sentenciaN;  
END LOOP
```

- Ejemplo:

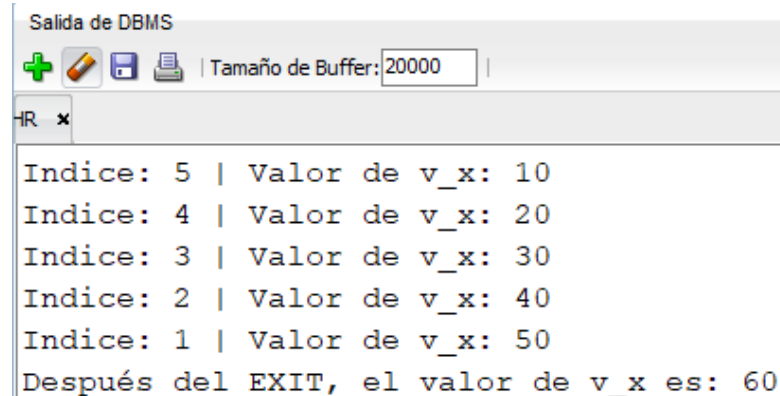
```
DECLARE
  v_x number := 10;
BEGIN
  FOR i IN 1 .. 5 LOOP
    dbms_output.put_line('Indice: ' || i || ' | Valor de v_x: ' || v_x);
    v_x := v_x + 10;
  END LOOP;
  -- Después de la salida del LOOP, la ejecución del bloque PL/SQL continúa aquí
  DBMS_OUTPUT.PUT_LINE('Después del EXIT, el valor de v_x es: ' || v_x);
END;
```



```
Salida de DBMS
+ Tamaño de Buffer: 20000
DBMS
Indice: 1 | Valor de v_x: 10
Indice: 2 | Valor de v_x: 20
Indice: 3 | Valor de v_x: 30
Indice: 4 | Valor de v_x: 40
Indice: 5 | Valor de v_x: 50
Después del EXIT, el valor de v_x es: 60
```

- Ejemplo:

```
DECLARE
  v_x number := 10;
BEGIN
  FOR i IN REVERSE 1 .. 5 LOOP
    dbms_output.put_line('Indice: ' || i || ' | Valor de v_x: ' || v_x);
    v_x := v_x + 10;
  END LOOP;
  -- Después de la salida del LOOP, la ejecución del bloque PL/SQL continúa aquí
  DBMS_OUTPUT.PUT_LINE('Después del EXIT, el valor de v_x es: ' || v_x);
END;
```



Salida de DBMS

Tamaño de Buffer: 20000

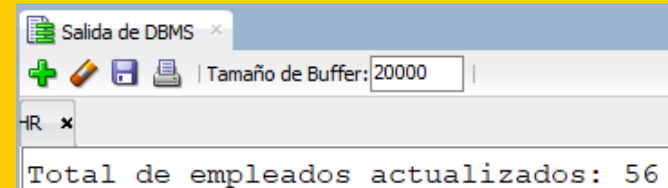
Indice: 5 | Valor de v\_x: 10  
Indice: 4 | Valor de v\_x: 20  
Indice: 3 | Valor de v\_x: 30  
Indice: 2 | Valor de v\_x: 40  
Indice: 1 | Valor de v\_x: 50  
Después del EXIT, el valor de v\_x es: 60

- Ejemplo:

```
DECLARE
  v_countryid  locations.country_id%TYPE := 'CA';
  v_loc_id     locations.location_id%TYPE;
  v_new_city   locations.city%TYPE := 'Montreal';
BEGIN
  SELECT MAX(location_id)
    INTO v_loc_id
  FROM locations
 WHERE country_id = v_countryid;
  FOR i IN 1..3 LOOP
    INSERT INTO ubicaciones(location_id, city, country_id)
      VALUES((v_loc_id + i), v_new_city, v_countryid );
  END LOOP;
END;
```

- Ejemplo:

```
VAR b_porcentaje_aumento NUMBER
EXEC :b_porcentaje_aumento:=1.25
DECLARE
v_sal_prom NUMBER(7);
v_id_min NUMBER(3);
v_id_max NUMBER(3);
v_tot_emp_act NUMBER(3):=0;
BEGIN
SELECT ROUND(AVG(salary)),MIN(employee_id), MAX(employee_id)
  INTO v_sal_prom,v_id_min,v_id_max
  FROM employees;
FOR i IN v_id_min .. v_id_max LOOP
  UPDATE empleados
    SET salary=ROUND(salary*b_porcentaje_aumento)
  WHERE employee_id=i
    AND salary < v_sal_prom;
  IF SQL%ROWCOUNT > 0 THEN
    v_tot_emp_act:=v_tot_emp_act+1;
  END IF;
END LOOP;
DBMS_OUTPUT.PUT_LINE('Total de empleados actualizados: ' || v_tot_emp_act);
COMMIT;
END;
```



Se pueden tener  
múltiples niveles  
de LOOP Simple,  
WHILE LOOP o  
FOR LOOP

El uso de  
etiquetas  
permiten  
distinguir entre  
bloques y loops

- Ejemplo:

```
DECLARE
  v_i number(3);
  v_j number(3);
BEGIN
  v_i := 2;
  LOOP << LOOP_PRINCIPAL >>
    v_j := 2;
    LOOP << LOOP_INTERNO >>
      EXIT WHEN ((MOD(v_i, v_j) = 0) OR (v_j = v_i));
      v_j := v_j + 1;
    END LOOP LOOP_INTERNO;
    IF (v_j = v_i) THEN
      DBMS_OUTPUT.PUT_LINE(v_i || ' es número primo');
    END IF;
    v_i := v_i + 1;
    EXIT WHEN v_i = 50;
  END LOOP LOOP_PRINCIPAL;
END;
```

Salida de DBMS

Tamaño de Buffer: 20000

2 es número primo  
3 es número primo  
5 es número primo  
7 es número primo  
11 es número primo  
13 es número primo  
17 es número primo  
19 es número primo  
23 es número primo  
29 es número primo  
31 es número primo  
37 es número primo  
41 es número primo  
43 es número primo  
47 es número primo



# LOOPS Anidados

- Ejemplo:

```
DECLARE
v_iddep_min NUMBER(3);
v_iddep_max NUMBER(3);
v_depto_emp NUMBER(3);
v_nom_emp VARCHAR2(30);
id_emp NUMBER(3);
BEGIN
SELECT MIN(department_id), MAX(department_id)
  INTO v_iddep_min,v_iddep_max
  FROM departments;
WHILE v_iddep_min <= v_iddep_max LOOP
  DBMS_OUTPUT.PUT_LINE('Departamento: ' || v_iddep_min);
  id_emp:=100;
  WHILE id_emp <= 206 LOOP
    SELECT first_name || ' ' || last_name, department_id
      INTO v_nom_emp, v_depto_emp
      FROM employees
      WHERE employee_id=id_emp;
    IF v_iddep_min = v_depto_emp THEN
      DBMS_OUTPUT.PUT_LINE('      ' || v_nom_emp);
    END IF;
    id_emp:=id_emp+1;
  END LOOP;
  DBMS_OUTPUT.PUT_LINE('-----');
  v_iddep_min:=v_iddep_min+10;
END LOOP;
END;
```

Salida de DBMS

Tamaño de Buffer: 20000

Departamento: 10  
Jennifer Whalen

-----

Departamento: 20  
Michael Hartstein  
Pat Fay

-----

Departamento: 30  
Den Raphaely  
Alexander Khoo  
Shelli Baida

-----

-----

-----

Departamento: 90  
Steven King  
Neena Kochhar  
Lex De Haan

-----

Departamento: 100  
Nancy Greenberg  
Daniel Faviet

-----

-----

-----

Departamento: 250

-----

Departamento: 260

-----

Departamento: 270

-----

- Se explicó qué es una Estructura de Iteración o LOOP.
- Se describieron los tipos de LOOP que se pueden usar en bloques PL/SQL.
- Se explicó cómo construir Estructuras de Iteración usando LOOP básico.
- Se explicó cómo construir Estructuras de Iteración usando WHILE LOOP
- Se explicó cómo construir Estructuras de Iteración usando FOR LOOP.
- Se explicó cómo trabajar con LOOPS Anidados.