

# Uso de variables

Las variables se utilizan para:

- Almacenamiento de forma temporal de datos.
- Manipulación de valores almacenados.
- Reusabilidad.
- Fácil mantenimiento.

La gestión de variables en PL/SQL presentan los siguientes aspectos:

- Las variables se declaran e inicializan dentro de la sección de declaración.
- Asignar nuevos valores a las variables dentro de la sección de códigos.
- Pasar valores a los bloques PL/SQL a través de parámetros.
- Ver los resultados a través de las variables de salida.

## ❖ Variables PL/SQL:

- **Escalares:** Contienen un valor único. Los principales tipos de datos son los que corresponden a tipos de columnas en las tablas del servidor Oracle8; PL/SQL también soporta variables booleanas.
- **Compuestas:** Hacen referencia a tipos de datos compuestos, como los registros y las tablas.
- **Referenciada:** o punteros, que designan otros artículos del programa.
- **LOB:** Especifican imágenes o cualquier conjunto de caracteres o bytes que tengan una longitud excesiva.

Sintaxis:

```
Nb_variable [CONSTANT] data_type [NOT NULL]
[:= expr | DEFAULT expr ];
```

Dónde:

CONSTANT: restringe la variable para que no pueda cambiar de valor. Las constantes deben inicializarse.

DATA\_TYPE: Es un tipo de datos escalar, compuesto o LOB.

NOT NULL: restringe la variable. La obliga a contener algún valor. Las NOT NULL también se deben inicializar.

EXPR: Cualquier expresión PL/SQL, pudiendo ser un literal, otra variable, etc.

Ejemplos:

```
V_HIREDATE NUMBER(2) NOT NULL:=10;
V_HIREDATE DATE;
V_LOC VARCHAR2(30):='ATLANTA';
C_COMM CONSTANT NUMBER(4):=10;
```

### **Recomendaciones a seguir**

- 1 – Intentar poner los identificadores: V variable, C constante, G global.
- 2 – Inicializar las constantes y las variables NOT NULL.
- 3 – Inicializar los identificadores con := ó DEFAULT.
- 4 – Declarar como máximo un identificador por línea.

### **Reglas para los nombres**

- Dos variables pueden tener el mismo nombre si están en dos bloques diferentes.
- El nombre de la variable (identificador) no debe ser el mismo que el de una columna de una tabla utilizada en el bloque.
- Se puede asignar cualquier valor a una variable simplemente nb\_var:=expresion;
- Se pueden usar las funciones TO\_NUMBER, TO\_DATE y TO\_CHAR.

## Tipos de variables

---

### Variables escalares:

Tipos	Descripción
VARCHAR2(n)	Para datos tipo carácter de longitud 3270 bytes.
NUMBER[(p,s)]	Enteros y decimales.
DATE	Fechas y horas.
CHAR(n)	Cadenas de longitud fija.
LONG	Datos carácter de longitud variable.
LONG RAW	Datos binarios y cadenas.
BOOLEAN	Utilizada para cálculos lógicos, admite TRUE, FALSE y NULL. Sólo pueden estar conectadas por los operadores lógicos AND, NOT, OR y las expresiones aritméticas de carácter y fecha que pueden devolver un valor booleano.
BINARY_INTEGER	Tipo de base de enteros entre -2147483647 y 2147483647.

### Variables de datos compuestos:

Existen fundamentalmente 2 tipos de datos:

- **Registros:** trata datos relacionados pero no iguales como una unidad lógica.
- **Tablas:** hacen referencia a colecciones de datos que permiten ser manipulados.

### Variables LOB:

Almacenan gran cantidad de datos no estructurados. Las más importantes son:

- CLOB: Almacenan grandes bloques de caracteres.
- BLOB: Almacenan objetos binarios grandes en la B.D.
- BFILE: Almacenan objetos binarios grandes en archivos del sistema.
- NCLOB: Almacenan bloques grandes de datos de longitud fija.

## Registros

---

Los registros presentan las siguientes características:

- ♦ Los registros deben contener una o más componentes llamados *campos*.
- ♦ No es lo mismo que la fila de una tabla.
- ♦ Tratan una colección de campos como unidad lógica.
- ♦ Son adecuados para recuperar una fila de datos de una tabla.
- ♦ Se declaran en la zona *declare*.

Sintaxis:

```
TYPE nb_tipo_reg IS RECORD  
(field_declaration [, field_declaration] ...);  
  
nb_reg nb_tipo_reg;
```

Donde *field\_declaration* tienen la siguiente estructura:

```
Nb_campo {tip_dato_campo | variable%TYPE | tabla.column%TYPE | tabla%ROWTYPE}  
[[NOT NULL]][:= | DEFAULT] expr ]
```

Donde:

NB\_TIPO\_REG: es el nombre del tipo de registro.  
NB\_REG: es el nombre del registro.  
NB\_CAMPO: nombre del campo que está dentro del registro.  
TIPO\_DATO\_CAMPO: tipo de dato de un campo.  
EXPR: valor inicial.

Ejemplo 1:

Inicializar un registro ALUMNO con los siguientes valores nombre, número de matrícula y código en la zona declarativa.

```
TYPE REG_ALUMNO IS RECORD  
(NOM_ALUMNO VARCHAR2(25)  
NUM_MATR NUMBER(7,2),  
CODIGO NUMBER(3));  
  
ALUMNO REG_ALUMNO;
```

## Atributos %type y %rowtype

---

Hay muchos casos donde las variables PL se emplean para manipular datos almacenados en una tabla en la BD. En este caso la/s variable/s debe/n tener el mismo tipo que la/s columna/s de la tabla.

- **% type:** El atributo %TYPE define el tipo de una variable utilizando una definición previa

de otra variable o columna de la base de datos.

- **% rowtype**: Creamos una variable que es una estructura o un registro que contiene las mismas columnas que la tabla a la que estamos haciendo referencia.

#### Ejemplo:

```
DECLARE
empleado          Number(6,0);
empleado          employee.employee_id%TYPE;
empleado_todo     employee%ROWTYPE;
```

#### Ventajas del atributo %ROWTYPE

- El número y tipo de datos de las columnas de la base de datos pueden no ser conocidos.
- El número y tipos de datos de las columnas de la base de datos pueden cambiar en el momento de la ejecución.
- Es útil para recuperar una fila con la sentencia SELECT.

#### Ejemplo:

Realizar el bloque anterior utilizando todos los campos de la tabla y el atributo %ROWTYPE:

```
DECLARE
    CURSOR C1 IS
    SELECT * FROM DEPT
    ORDER BY DEPTNO;

    REG C1%ROWTYPE;

    V_NUM NUMBER:= &NÚMERO_FILAS;

BEGIN
    OPEN C1;
    FETCH C1 INTO REG;
    WHILE (C1%ROWCOUNT <= V_NUM) AND (C1%FOUND) LOOP
        DBMS_OUTPUT.PUT_LINE (REG.DEPTNO||' '||REG.DNAME||'
        '||REG.LOC);
        FETCH C1 INTO REG;
    END LOOP;
    CLOSE C1;
END;
```

## Cómo asignar valores a variables

---

Es posible asignar valores a las variables de varias formas.

La primera utiliza el operador "=". La variable se ubica al lado izquierdo y la expresión al lado derecho del símbolo. Esta asignación se puede hacer dentro del bloque declarativo o dentro del bloque de "código".

Ejemplos:

tax := price \* tax\_rate ;

V\_num number := 45;

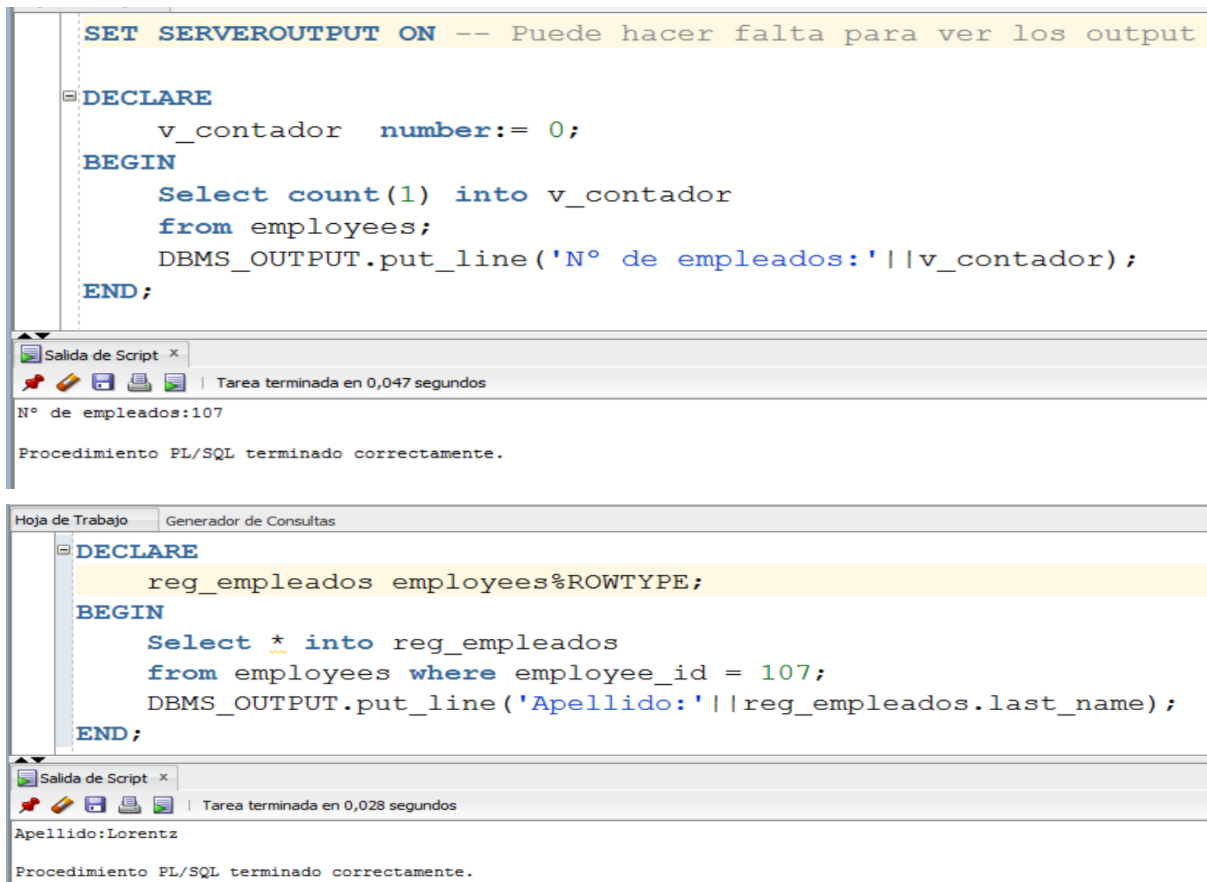
v\_var number not null := 0; (obligatorio inicializar)

### A partir de una Select

Se puede asignar valor a una variable (individual o de tipo registro) a partir de datos recuperados de una consulta:

Sintaxis:

```
SELECT lista_seleccionada
INTO {nb_variable1 [, nb_variable2 .....] | nb_registro}
FROM nb_tabla
WHERE condición;
```



The first screenshot shows a PL/SQL script in the 'Hoja de Trabajo' (Worksheet) window. The script declares a variable `v_contador` of type `number` and initializes it to 0. It then uses a `SELECT count(1) into v_contador from employees;` statement to count the number of employees. Finally, it uses `DBMS_OUTPUT.put_line('N° de empleados:' || v_contador);` to output the result. The 'Salida de Script' (Script Output) window shows the execution result: 'N° de empleados:107' and a message 'Procedimiento PL/SQL terminado correctamente.' (PL/SQL procedure terminated successfully).

The second screenshot shows a similar PL/SQL script in the 'Hoja de Trabajo' window. This script declares a record variable `reg_empleados` of type `employees%ROWTYPE`. It uses a `SELECT * into reg_empleados from employees where employee_id = 107;` statement to retrieve the details of the employee with ID 107. It then uses `DBMS_OUTPUT.put_line('Apellido:' || reg_empleados.last_name);` to output the last name. The 'Salida de Script' window shows the execution result: 'Apellido:Lorentz' and a message 'Procedimiento PL/SQL terminado correctamente.'

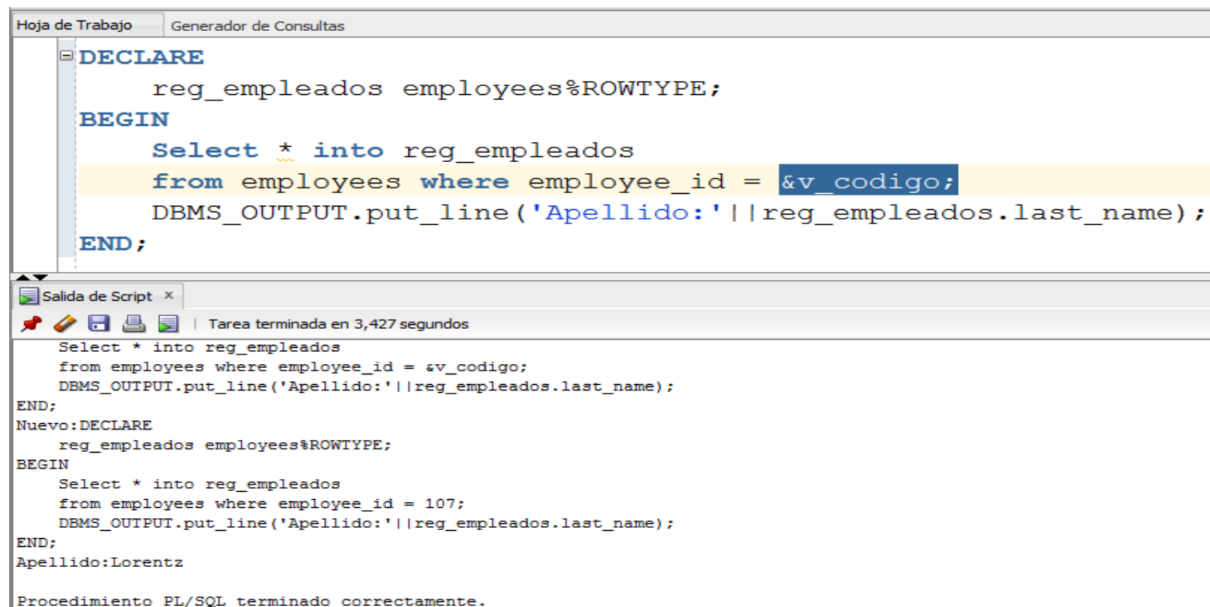
## Ejercicio

(VIVERO) Hacer un bloque anónimo de PL/SQL en el que declare una variable con un código de cliente y me saque el total de pedidos que ha hecho ese cliente.

## Variables de sustitución

Oracle dispone de un mecanismo que permite al usuario realizar operaciones de entrada. Oracle hace la sustitución textual de la variable antes de que el bloque PL o la orden SQL se envíen al servidor. Se designan mediante el carácter (&).

No hay memoria asignada para las variables de sustitución. Por eso, sólo se puede utilizar para recoger datos del teclado (luego lo guardamos en otra variable o desaparece).



The screenshot shows the Oracle SQL Developer interface. The top pane, titled 'Hoja de Trabajo' and 'Generador de Consultas', contains a PL/SQL script:

```
DECLARE
    reg_empleados employees%ROWTYPE;
BEGIN
    Select * into reg_empleados
    from employees where employee_id = &v_codigo;
    DBMS_OUTPUT.put_line('Apellido:' || reg_empleados.last_name);
END;
```

The bottom pane, titled 'Salida de Script', shows the execution results. It indicates the task was completed in 3,427 seconds. The output of the script is as follows:

```
END;
Nuevo:DECLARE
    reg_empleados employees%ROWTYPE;
BEGIN
    Select * into reg_empleados
    from employees where employee_id = 107;
    DBMS_OUTPUT.put_line('Apellido:' || reg_empleados.last_name);
END;
Apellido:Lorentz
Procedimiento PL/SQL terminado correctamente.
```

## Ejercicio

(VIVERO) Transformar el bloque anónimo anterior para que me pase el código de cliente por teclado el usuario.

## Ámbito y visibilidad de una variable PL

Es aquella parte del programa en la que se puede acceder a dicha variable. Para una variable PL el ámbito va desde la definición de la variable hasta el final del bloque

```

DECLARE
    X BINARY_INTEGER;
BEGIN
    DECLARE
        Y BINARY_INTEGER;
    BEGIN
        .....
    EXCEPTION
        .....
    END;
EXCEPTION
    .....
END;

```

Las referencias a un identificador son resueltas de acuerdo a su alcance y visibilidad dentro de un programa. El *alcance* de un identificador es aquella región de la unidad de programa (bloque, subprograma o paquete) desde la cual se puede referenciar al identificador.

Un identificador es visible sólo en las regiones en que se puede referenciar.

Los identificadores declarados en un bloque de PL/SQL se consideran locales al bloque y globales a todos sus sub-bloques o bloques anidados. De esto se desprende que un mismo identificador no se puede declarar dos veces en un mismo bloque pero sí en varios bloques diferentes, cuántas veces se desee.