



# **BASES DE DATOS II**

## **INTRODUCCIÓN PL/SQL**

MSc. Jimena Adriana Timaná Peña

# Introducción

¿Qué es PL/SQL?

# Introducción

- Procedural Language/SQL (PL/SQL) extensión de SQL, con características de diseño de lenguajes de programación.
- Lenguaje procedimental que amplía la funcionalidad de SQL añadiendo estructuras habituales en otros lenguajes: variables y tipos, estructuras de control, procedimientos y funciones, paquetes.
- Los procedimientos, funciones, disparadores y paquetes creados con el PL/SQL se almacenan en base de datos.
- Están incluidos dentro de las políticas de seguridad de Oracle y son altamente recomendables, para el tratamiento de datos.

# Ventajas PL/SQL

- **Integración con SQL ( Consultas, DML, funciones, etc)**
  - Permiten la manipulación de datos e instrucciones de consulta de SQL para ser incluidos en un bloque estructurado y procedural de unidades de código, haciendo de PL/SQL un potente lenguaje de procesamiento de transacciones.
- **Alto rendimiento**
  - Es posible enviar un bloque de sentencias a la BD
  - Subprogramas compilados una vez y almacenados como ejecutables

# Ventajas PL/SQL

## **Portabilidad**

- Las aplicaciones PL/SQL se pueden ejecutar en cualquier plataforma donde se esté ejecutando la BD

## **Paquetes predefinidos**

- APIs que se pueden invocar para ejecutar tareas útiles como DBMS\_FILE para leer y escribir en archivos o DBMS\_OUTPUT para mostrar información en la consola

# Bloques

- La unidad básica en PL/SQL es el bloque.
- Todos los programas PL/SQL están compuestos por bloques, que pueden definirse de forma secuencial o estar anidados.
- Normalmente cada bloque realiza una unidad lógica de trabajo en el programa, separando así unas tareas de otras

# Tipos de Bloques

**A nivel general hay 2 tipos de bloques:**

- Bloques Anónimos
- Bloques con nombre

# Tipos de Bloques

## **Anónimos (Anonymous blocks):**

- Se construyen de forma dinámica y se ejecutan una sola vez.
- **No** se guardan en la BD.

## **Con nombre (Named blocks):**

- Se llaman también subprogramas.
- Se guardan en la BD.
- Pueden ser procedimientos, funciones o paquetes



# Estructura de un Bloque

Un bloque consta de 3 secciones:

- **Declarativa:** para la declaración de las variables, es opcional.
- **Ejecutable:** sentencias SQL y PL/SQL, es obligatorio.
- **Manejo de excepciones:** se especifican las acciones a realizar cuando ocurren errores, es opcional.



Las únicas palabras requeridas son **BEGIN** y **END**

Las palabras **DECLARE**, **BEGIN** y **EXCEPTION** no llevan al final punto y coma (;) solo las palabras **END**, sentencias SQL y PL/SQL lo requieren

# Identificadores

- Se emplean para dar nombre a los objetos PL/SQL, tales como variables, cursores, tipos y subprogramas.
- Los identificadores inician con una letra, seguida por una secuencia opcional de caracteres, que pueden incluir letras, números, signos de dólar (\$), caracteres de subrayado y símbolos de almohadilla (#). Los demás caracteres no pueden emplearse.
- La longitud máxima es de 30 caracteres
- No diferencia entre mayúsculas y minúsculas
- No usar palabras reservadas

# Salida de Datos

## **DBMS\_OUTPUT.PUT\_LINE(*mensaje a mostrar*)**

- Es un paquete de Oracle que se utiliza para desplegar datos en un bloque PL/SQL.
- Para ser utilizado debe colocarse previamente: *SET SERVEROUTPUT ON*

```
SET SERVEROUTPUT ON  
DECLARE  
    v_num NUMBER := 4;  
BEGIN  
    DBMS_OUTPUT.PUT_LINE('Imprimiendo...');  
    DBMS_OUTPUT.PUT_LINE(v_num);  
END;
```

# Variables

Con PL/SQL se puede declarar variables y usarlas para almacenamiento temporal, manipulación de valores almacenados, entre otros.

- Se declaran y se inicializan en la sección de **DECLARE**.
- Permite asignar nuevos valores a variables en la sección de ejecución

## REGLAS DE NOMBRADO

- Trate de no usar el mismo nombre de la variable con el nombre de alguna de las columnas de una respectiva tabla.
- Por convención a los nombres de las variables, declárelas precedidas de **v\_** para indicar que es una variable, **g\_** para indicar que es una variable global y **c\_** para indicar que es una constante.

# Tipos de Datos de Variables

VARCHAR2 (maximum_length)	Tipo carácter de longitud variable hasta 32.767 bytes. No hay un tamaño definido por defecto.
• NUMBER [(precision, scale)]	Tipo numérico fijo y de punto flotante.
DATE	Fecha y hora
CHAR [(maximum_length)]	Tipo carácter de longitud fija hasta 32.767 bytes. Si no se especifica la longitud máxima por defecto se fija en 1.
LONG	Tipo carácter de longitud variable hasta 32.760 bytes.
LONG RAW	Tipos binarios.
BOOLEAN	Almacena uno de los tres posibles valores usados para cálculos lógicos: TRUE, FALSE, NULL.
BINARY_INTEGER	Enteros entre -2,147,483,647 y 2,147,483,647
PLS_INTEGER	Enteros con signo entre -2,147, 483,647 y 2, 147,483,647. Requieren menos almacenamiento y son más rápidos que valores de tipo NUMBER y BINARY_INTEGER.
LOB	(large object). CLOB (character large object): Book, BLOB(binary large object): Photo, BFILE(binary file): Movie

# Uso de Variables

## Declaración:

- Para declarar variables se debe seguir la siguiente sintaxis:

```
identifier [CONSTANT] datatype [NOT NULL]  
[:= | DEFAULT expr];
```

## Asignación de valores a variables

```
v_deptno          NUMBER(2) NOT NULL := 10;
```

```
c_comm           CONSTANT NUMBER := 1400;
```

```
v_valid          BOOLEAN NOT NULL := TRUE;
```

Para inicializar un identificador se usa el operador de inicialización (:=) o la palabra reservada **DEFAULT**.

# EL ATRIBUTO %TYPE

Es usado cuando se declara una variable de acuerdo a otra variable declarada previamente.

```
v_ename          emp.ename%TYPE;  
v_balance        NUMBER(7,2);  
v_min_balance    v_balance%TYPE := 10;
```

# SENTENCIA SELECT DE PL/SQL

**INTO** : Extraer una fila de datos de utilizando el comando SELECT y almacenar los datos en una o más variables. Sólo puede ser devuelta una fila

```
SELECT    columnas
INTO      {variable1[, variable2]...
            | nombre_registro}
FROM      tabla
WHERE      condicion;
```



# VARIABLES BIND

Una variable bind es una variable que se declara en un entorno host y luego se usa para pasar valores en tiempo de ejecución, ya sea número o carácter, dentro y fuera de uno o más programas PL / SQL, la cual puede usarla como usaría cualquier otra variable.

Para declarar una variable bind, debe usar el comando **VARIABLE**.

```
VARIABLE return_code NUMBER  
VARIABLE return_msg VARCHAR2(30)
```

- Para desplegar variables bind se usa el comando PRINT. Sin embargo, dicho comando no puede ser usado dentro de un bloque PL/SQL.

```
SQL> VARIABLE g_n NUMBER  
...  
SQL> PRINT g_n
```

# VARIABLES BIND

- La referencia a la variable host o bind se hace anteponiendo a la variable : (dos puntos) para poderlas diferenciar de las variables declaradas en el bloque PL/SQL.

## Ejemplo:

```
VARIABLE resultado NUMBER;  
BEGIN  
  SELECT SAL*2 INTO :resultado  
  FROM EMP WHERE EMPNO = 7934;  
END;  
PRINT resultado;
```

# VARIABLES DE SUSTITUCIÓN

- Igualmente usadas para almacenar valores.
- pueden aparecer directamente en la sentencia SELECT sin necesidad de definirla, anteponiendo el símbolo **&** y SQL nos preguntará el valor que queremos asignarle.
- Variable de sustitución: **&**
- Para reutilizar el valor de una variable digitada: **&&**

## Notas importantes:

- *Cuando trabajamos pidiendo datos al usuario es habitual especificar la opción SET VERIFY OFF para evitar que el sistema nos muestre el valor que tenía la variable antes y que nos confirme el nuevo valor que toma.*
- *La opción SET ECHO ON/OFF despliega o no en pantalla los comandos*

# Ejemplos con Variables de Sustitución

- Seleccione el nombre, el cargo y el salario de la tabla EMP, donde la identificación del empleado sea igual a 7782. El usuario será quien digite dicho valor. Utilice una variable de sustitución para pedir dicho ID.

# Ejemplos con Variables de Sustitución

- Seleccione el nombre, el cargo y el salario de la tabla EMP, donde la identificación del empleado sea igual a 7782. El usuario será quien digite dicho valor. Utilice una variable de sustitución para pedir dicho ID.

**SET VERIFY OFF**

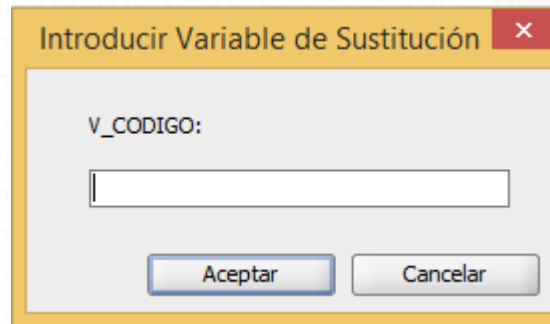
**SET ECHO OFF**

**SELECT** ename,job,sal

**FROM** emp

**WHERE** empno=&v\_Codigo;

Cuando ejecute la consulta se le pedirá que ingrese el valor que tomará la variable de sustitución:



Introducir Variable de Sustitución

V\_CODIGO:

Aceptar Cancelar

Una vez ingrese el valor se mostrará en pantalla el resultado de la consulta:

	ENAME	JOB	SAL
1	CLARK	MANAGER	2450

# Ejemplos Bloque Anónimo y Variables de Sustitución

- Cree un bloque anónimo que declare variables para nombre, número de departamento, salario y fecha de ingreso, todas las anteriores con un valor inicial y luego muéstrelas utilizando el paquete DBMS\_OUTPUT. Para mostrar la fecha utilice la función TO\_CHAR.

# Ejemplos Bloque Anónimo y Variables de Sustitución

- Cree un bloque anónimo que declare variables para nombre, número de departamento, salario y fecha de ingreso, todas las anteriores con un valor inicial y luego muéstrelas utilizando el paquete DBMS\_OUTPUT. Para mostrar la fecha utilice la función TO\_CHAR.

```
SET SERVEROUTPUT ON
```

```
DECLARE
```

```
v_nombre VARCHAR(50) := 'Alan Brito';  
v_numdep NUMBER := 30;  
v_salario NUMBER(11,2) := 1000000;  
v_fechai DATE := '13/06/12';
```

```
BEGIN
```

```
DBMS_OUTPUT.PUT_LINE('El nombre del empleado es: ' || v_nombre);  
DBMS_OUTPUT.PUT_LINE('El número del departamento es: ' || v_numdep);  
DBMS_OUTPUT.PUT_LINE('El salario del empleado es: ' ||  
TO_CHAR(v_salario,'$999,999,999.00'));  
DBMS_OUTPUT.PUT_LINE('La fecha de contrato del empleado es: ' || TO_CHAR(v_fechai,'DAY  
- MON - YEAR'));  
END;
```

```
El nombre del empleado es: Alan Brito
```

```
El número del departamento es: 30
```

```
El salario del empleado es:      $1,000,000.00
```

```
La fecha de contrato del empleado es: MIÉRCOLES - JUN - TWENTY TWELVE
```

# Ejemplos Bloque Anónimo y Variables de Sustitución

- A través de un bloque anónimo seleccione el nombre, salario, bono (que corresponde al valor del 10% del salario) y el salario incrementado con el bono de la tabla empleados (emp) del empleado con identificación 7839. El valor de la identificación será ingresado por el usuario.

EL mensaje resultante tendrá el siguiente formato:

- El empleado KING que gana un salario de \$5,000.00 recibirá un bono del 10% de su sueldo equivalente a \$500.00 así su nuevo salario será de \$5,500.00



# Ejemplos Bloque Anónimo y Variables de Sustitución

```
SET SERVEROUTPUT ON
SET VERIFY OFF
SET ECHO OFF
```

```
DECLARE
```

```
  v_id emp.empno%TYPE;
  v_nombre emp.ename%TYPE;
  v_salario emp.sal%TYPE;
  v_bono emp.sal%TYPE;
  v_nuevo emp.sal%TYPE;
```

```
BEGIN
```

```
  SELECT ENAME, SAL, (SAL*0.1) AS BONO, (SAL+(SAL*0.1)) AS NUEVO
  INTO v_nombre, v_salario, v_bono, v_nuevo
  FROM EMP
  WHERE empno = &v_id;
```

```
  DBMS_OUTPUT.PUT_LINE('El empleado ' || v_nombre || ' que gana un salario de ' ||
  TO_CHAR(v_salario, '$999,999,999.00') || ' recibirá un bono del 10% de su sueldo equivalente a ' ||
  TO_CHAR(v_bono, '$999,999,999.00') || ' así su nuevo salario será de ' || TO_CHAR(v_nuevo,
  '$999,999,999.00') );
```

```
END;
```