

PRÁCTICA 1. VARIABLES Y BLOQUES EN PL/SQL.

1. Evalúa cada una de las declaraciones de variables siguientes y determina cuales no son válidas y por qué.

DECLARE

v_id NUMBER(4); → CORRECTO

v_x, v_y, v_z VARCHAR2(10); → Incorrecto, porque no puede declarar múltiples variables a la vez, aunque puede ponerlos en la misma línea, siempre ya que tiene tipo de datos y punto y coma después de cada uno.

v_birthdate DATE NOT NULL; → Incorrecto, porque si no va a ser nulo, entonces debe tener un valor inicial, así

v_in_stock BOOLEAN := 1; → Incorrecto, porque el tipo de datos y el valor deben coincidir. Debería ser BOOLEAN: = verdadero o NUMBER: = 1.

emp_record emp_record_type; → Correcto.

```
TYPE name_table_type IS TABLE OF VARCHAR2(20);
    INDEX BY BINARY_INTEGER;
dept_name_table name_table_type;
```

....

2. Indica el valor de las variables del siguiente bloque PL/SQL en cada momento:

DECLARE

v_customer VARCHAR2(50) := ' Womansport '; → el valor es Womansport

v_weight NUMBER(3) := 600; → el valor es 600

v_message VARCHAR2(255) := ' Producto 10012 '; → el valor es 10012

BEGIN

DECLARE

v_customer NUMBER(7) := 201; → el valor es 201

v_message VARCHAR2(255) := ' Producto 11001 '; → el valor es 11001

v_new_locn VARCHAR2(50) := ' Europa '; → el valor es Europa

3. Construye un bloque PL/SQL que pida el precio de un producto y el valor del IVA a aplicarle, e imprima el valor del total del producto.

DECLARE

p number;

iva number;

valor number;

BEGIN

p := 300;

iva := 21;

valor := p+p*iva/100

DBMS_OUTPUT.PUT_LINE('El valor es: ' || valor);

END;

4. Construye un bloque PL/SQL que seleccione el artículo de mayor PVP en la tabla ARTICULOS y almacene su valor en una variable de SQL para imprimirlo a continuación.

```
DECLARE
mayor number;
BEGIN
select max(precio_venta) into mayor from ARTICULOS;
DBMS_OUTPUT.PUT_LINE('El valor es: ' || max_pvp);
END;
```

5. Crea y almacena en un fichero un bloque PL/SQL que inserte un nuevo artículo en la tabla ARTICULOS. Los datos del código y nombre deben pedirse previamente por teclado para darlos como argumentos.

```
DECLARE OR REPLACE PROCEDURE nuevo_articulo (codigo NUMBER,
nombre
varchar2)
BEGIN
INSERT INTO ARTICULOS values (codigo,nombre);
END;
```

6. Un bloque PL/SQL debe actualizar la dirección de un cliente en la tabla CLIENTES. Para ello, el bloque pedirá el código del cliente y su nueva dirección.

```
DECLARE
codigo_c NUMBER;
direccion_c VARCHAR;
BEGIN
UPDATE CLIENTES SET direccion=direccion_c where codigo=codigo_c;
END;
```

7. Crea un bloque que borre todos los proveedores de un país que se pedirá por teclado. Debe imprimirse el número de proveedores que se han borrado.

```
DECLARE
pais%TYPE:=&n_pais;
n NUMBER;
BEGIN
select count(*) into n from fabricantes where pais=n_pais;
DELETE FROM fabricantes WHERE pais=n_pais;
DBMS_OUTPUT.PUT_LINE(n);
END;
```



PRÁCTICA 2. ESTRUCTURAS DE CONTROL EN BLOQUES PL/SQL

1. Crea una tabla MENSAJES con un solo campo VALOR, de tipo VARCHAR2(5). Crea un bloque que inserte 8 elementos en la tabla con valores del 1 al 10, excepto el 4 y 5.

```
CREATE TABLE MENSAJES(  
VALOR varchar(5),  
);  
DECLARE  
FOR numero IN 1..10 LOOP  
IF numero=4 or numero=5 THEN  
null;  
ELSE  
INSERT INTO MENSAJES VALUES (numero);  
ENDIF;  
END LOOP;  
END;
```

2. Un bloque PL/SQL debe aceptar el código de un artículo y actualizar su precio según las siguientes indicaciones. Si el artículo tiene un PVP menor a 100 ptas., su precio debe ser aumentado en 5 ptas. Si está entre 100 y 1.000 ptas. su precio subirá 15 y si excede las 1.000 ptas., el precio del producto debe subir 50 ptas. Si el PVP fuese NULL, el aumento sería 0.

```
DECLARE  
pvp%TYPE := &precio  
BEGIN  
IF precio<100 THEN  
UPDATE artículos SET pvp = pvp + 5;  
ELSEIF 100<precio>1000 THEN  
UPDATE artículos SET pvp = pvp + 15;  
ELSEIF precio>1000  
UPDATE artículos SET pvp = pvp + 50;  
ELSE  
null;  
ENDIF;  
END;
```



Pantalla de Trabajo

Archivo o URL: No se ha seleccionado ningún archivo.

Introducir Sentencias:

```
DECLARE  
pvp%TYPE := &precio  
BEGIN  
IF precio<100 THEN  
UPDATE artículos SET pvp = pvp + 5;  
ELSEIF 100<precio>1000 THEN  
UPDATE artículos SET pvp = pvp + 15;  
ELSEIF precio>1000  
UPDATE artículos SET pvp = pvp + 50;  
ELSE  
null;
```

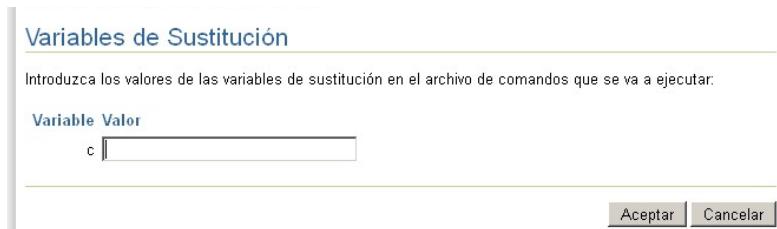
antiguo 2: pvp%TYPE := &precio
nuevo 2: pvp%TYPE := precio

3. Añade una nueva columna a la tabla ARTICULOS para almacenar cadenas de ‘*’. Crea un bloque PL/SQL que pida el código de un artículo y almacene un ‘*’ por cada 500 ptas. del precio del producto en la nueva columna de la tabla.

```

DECLARE
a_codigo%TYPE:=&c;
a_precio%TYPE:=precio;
cadena varchar2;
BEGIN
ALTER TABLE ARTICULOS ADD asteriscos varchar(50) NULL;
SELECT precio into a_precio FROM ARTICULOS where codigo=a_codigo;
WHILE a_precio=>500 LOOP
cadena := cadena+”*”;
a_precio:=a_precio-500;
END LOOP;
INSERT INTO ARTICULOS(asteriscos) values(cadena)
END;

```



PRÁCTICA 3. CURSORES EN BLOQUES PL/SQL.

1. Crea un bloque que almacene en la tabla AUX_ARTICULOS (previamente debe vaciarse) un número dado de los artículos con mayor PVP de los existentes en la tabla ARTICULOS. El número de artículos a almacenarse debe ser dado por teclado.

```

DECLARE
num := &numero;
fila := 1;
CURSOR c1 IS SELECT * FROM artículos ORDER BY DESC;

```

```

fila_articulo c1%ROWTYPE;
BEGIN
ALTER TABLE artículos ADD aux_articulos VARCHAR(20) NULL;
OPEN c1;
LOOP
FETCH INTO fila_articulos;
IF fila<num THEN
INSERT INTO aux_articulos (articulo, cod_fabricante, peso,
categoria, precio_venta, precio_costo, existencias)
values fila_articulos;
fila=fila+1;
END IF;
END LOOP;
CLOSE c1;
END;

```

Variables de Sustitución

Introduzca los valores de las variables de sustitución en el archivo de comandos que se va a ejecutar:

Variable Valor

numero



Pantalla de Trabajo

Archivo o URL: Examinar... No se ha seleccionado ningún archivo.

Introducir Sentencias:

```

DECLARE
num := &numero;
fila := 1;
CURSOR c1 IS SELECT * FROM artículos ORDER BY DESC;
fila_articulo c1%ROWTYPE;
BEGIN
ALTER TABLE artículos ADD aux_articulos VARCHAR(20) NULL;
OPEN c1;
LOOP
FETCH INTO fila_articulos;
IF fila<num THEN

```

antiguo 2: num := №
nuevo 2: num := numero0;

2. Escribe un bloque que recupere todos los proveedores por países. El resultado debe aparecer en una nueva tabla que permita almacenar datos del tipo:

'Proveedor: ONCE – País: ESPAÑA'

Utiliza un cursor para recuperar cada país de la tabla de PROVEEDORES y pasar dicho país a un cursor que obtenga el nombre de los proveedores en él. Una vez que se tiene el nombre del proveedor y su país, debe añadirse a la nueva tabla en el formato especificado.

```

DECLARE
vnumero number;
p proveedor%TYPE;
CURSOR c1 IS SELECT distinct país FROM proveedores;
a_pais c1%ROWTYPE;
BEGIN

```

```

OPEN c1;
FETCH c1 INTO a_pais;
WHILE c1%FOUND LOOP
SELECT proveedor INTO p FROM proveedores where pais=a_pais;
WHERE pais=vpais;
INSERT INTO proveedores values(a_pais,p);
END LOOP;
END c1;
END;

```



3. Modifica el ejercicio 3 de la práctica 2 mediante el uso de un cursor así como las funcionalidades FOR UPDATE y WHERE CURRENT OF para éste.

```

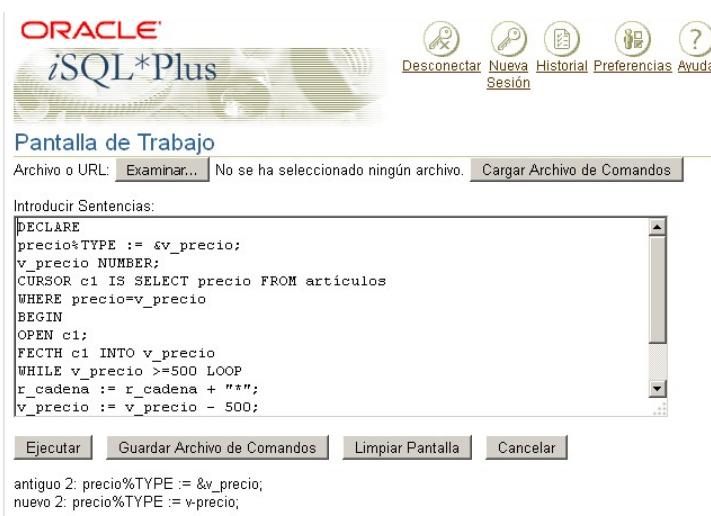
DECLARE
precio%TYPE := &v_precio;
v_precio NUMBER;
CURSOR c1 IS SELECT precio FROM artículos
WHERE precio=v_precio
BEGIN
OPEN c1;
FECTH c1 INTO v_precio
WHILE v_precio >=500 LOOP
r_cadena := r_cadena + "*";
v_precio := v_precio - 500;
INSERT INTO ARTICULOS (asteriscos) VALUES (r_cadena);
CLOSE c1;
END;

```

Variables de Sustitución

Introduzca los valores de las variables de sustitución en el archivo de

Variable	Valor
v_precio	<input type="text"/>



PRÁCTICA 4. EXCEPCIONES EN BLOQUES PL/SQL.

1. Escribe un bloque PL/SQL que seleccione el nombre de un proveedor a partir de una ciudad dada por teclado. El resultado de cada consulta debe almacenarse en una nueva tabla con una sola columna que sea una cadena de caracteres, de forma que:

- a) Si la ciudad introducida sólo proporciona una fila, la entrada a la nueva tabla será del tipo: ' Nombre_Proveedor – Ciudad '.
- b) Si la ciudad introducida no proporciona ningún proveedor, crear la excepción apropiada e insertar en la nueva tabla el mensaje: 'No existen proveedores en Ciudad'.
- c) Si la ciudad dada tiene varios proveedores, añadir la excepción adecuada e insertar en la nueva tabla el mensaje: ' Existen varios proveedores en Ciudad '.
- d) En cualquier otro caso, añadir una nueva excepción y añadir en la nueva tabla un nuevo mensaje: 'Se ha producido un error '.

DECLARE

pais := &vpais;

vproveedor VARCHAR2;

vnumpro VARCHAR2;

BEGIN

SELECT proveedor INTO vproveedor FROM artículos WHERE

pais=vpais;

SELECT count(proveedor) INTO vnumpro

FROM artículos WHERE pais=vpais;

IF vnumpro==1 THEN

INSERT INTO proveedores (nombre_proveedor, ciudad)

valu es (vproveedor, vpais);

ELSIF vnumpro==0 THEN

RAISE ningún_proveedor;

ELSE

RAISE muchos_proveedores;

END IF;

EXCEPTION

WHEN muchos_proveedores THEN

DBMS_OUTPUT.PUT_LINE('Existen varios proveedores en Ciudad');

WHEN ningún_proveedores THEN

DBMS_OUTPUT.PUT_LINE('No existen proveedores en Ciudad');

WHEN NO_DATA_FOUND THEN

DBMS_OUTPUT.PUT_LINE('Se ha producido un error');

END;

VARIABLES DE SUSTITUCIÓN

Introduzca los valores de las variables de sustitución en el archivo de comandos que se va a ejecutar:

Variable Valor

vpais

Aceptar | Cancelar

The screenshot shows the iSQL*Plus interface. At the top, there's a toolbar with icons for disconnect, new session, history, preferences, and session controls. Below the toolbar is a menu bar with 'Pantalla de Trabajo'. The main area is a large text input field labeled 'Introducir Sentencias:' containing the PL/SQL code provided in the text above. At the bottom of this field is a scroll bar. Below the input field are four buttons: 'Ejecutar', 'Guardar Archivo de Comandos', 'Limpiar Pantalla', and 'Cancelar'. Above the input field, there are two buttons: 'Examinar...' and 'Cargar Archivo de Comandos'. The status bar at the bottom displays the message 'antiguo 2: pais := &vpais;'. To the left of the main window, a smaller dialog box titled 'Variables de Sustitución' is open, prompting for the value of 'vpais'.

2. Modificar la solución al ejercicio 1.6 añadiendo un controlador de excepciones que controle cuando se introduce un código de cliente que no existe.

```
DECLARE
dir%TYPE := &direccion
emp_no%TYPE := &nombre
BEGIN
UPDATE emple SET dir = dirección WHERE emp_no = nombre;
EXCEPTION
WHEN not.data.found then
DBMS_OUTPUT_PUT_LINE ('No encontrado el cliente con código: ' ||
emp_no);
END;
```

The screenshot shows the Oracle iSQL*Plus interface. At the top, there's a 'Variables de Sustitución' dialog box for substituting variables 'direccion' and 'nombre'. Below it, the main workspace shows the PL/SQL code with these variables. The code updates a table 'emple' based on 'emp_no' and prints an error message if no row is found. Session variables like 'Sesión' and 'Ayuda' are visible at the top right. The main workspace shows the command line, output from previous commands, and execution buttons like 'Ejecutar'.

3. Escribe un bloque PL/SQL que imprima el número de artículos cuyo PVP esté entre 100 ptas. más o menos que un valor dado por teclado. En caso de que no haya artículos con precio en este rango, se debe imprimir un mensaje indicándolo. Si hay más uno o más artículos en este rango, el mensaje debe indicar cuántos artículos son los que están en este rango de valores y en caso de producirse cualquier otro error, debe crearse la excepción adecuada que indique la existencia de un error.

```
DECLARE
num number:=0;
BEGIN
SELECT count(*) INTO num
FROM articulos
WHERE pvp=100;
```

```

DBMS_OUTPUT.PUT_LINE ('Número de artículos: ' || num);
EXCEPTION
WHERE no.data.found then
DBMS_OUTPUT.PUT_LINE ('No encontrado ningún artículo: ');
END;

```

The screenshot shows the iSQL*Plus interface. The title bar says "ORACLE iSQL*Plus". The menu bar includes "Desconectar", "Nueva", "Historial", "Preferencias", and "Ayuda". Below the menu is a toolbar with icons for disconnect, new session, history, preferences, and help. The main area is titled "Pantalla de Trabajo" and has a sub-titile "Introducir Sentencias:". A code editor window contains the following PL/SQL code:

```

DECLARE
num number:=0;
BEGIN
SELECT count(*) INTO num
FROM articulos
WHERE pvp=100;
DBMS_OUTPUT.PUT_LINE ('Número de artículos: ' || num);
EXCEPTION
WHERE no.data.found then
DBMS_OUTPUT.PUT_LINE ('No encontrado ningún artículo: ');
END;

```

Below the code editor are four buttons: "Ejecutar" (Execute), "Guardar Archivo de Comandos" (Save Command File), "Limpiar Pantalla" (Clear Screen), and "Cancelar" (Cancel). At the bottom of the screen, there is a message: "WHERE no.data.found then".

PRÁCTICA 5. PROCEDIMIENTOS Y FUNCIONES EN PL/SQL

1. Crea un procedimiento que inserte un nuevo artículo en la tabla ARTICULOS.

```

CREATE OR REPLACE PROCEDURE nuevo
BEGIN
insert into articulos (articulo, cod_fabricante, peso, categoria,
precio_venta, precio_costo, existencia)
VALUES ('moto','001','350','yamaha','5000','3500','5');
END;

```

The screenshot shows the iSQL*Plus interface. The title bar says "ORACLE iSQL*Plus". The menu bar includes "Desconectar", "Nueva", "Historial", "Preferencias", and "Sesión". Below the menu is a toolbar with icons for disconnect, new session, history, preferences, and session. The main area is titled "Pantalla de Trabajo" and has a sub-titile "Introducir Sentencias:". A code editor window contains the following PL/SQL code:

```

CREATE OR REPLACE PROCEDURE nuevo
BEGIN
insert into articulos (articulo, cod_fabricante, peso, categoria,
precio_venta, precio_costo, existencia)
VALUES ('moto','001','350','yamaha','5000','3500','5');
END;

```

Below the code editor are four buttons: "Ejecutar" (Execute), "Guardar Archivo de Comandos" (Save Command File), "Limpiar Pantalla" (Clear Screen), and "Cancelar" (Cancel). At the bottom of the screen, there is a message: "Advertencia: Procedimiento creado con errores de compilación." (Warning: Procedure created with compilation errors).

2. Crea un procedimiento que permita modificar la descripción de un artículo. Incluye el control de errores con las excepciones oportunas.

```

CREATE OR REPLACE PROCEDURE modificar (n_codigo number,
v_descripcion varchar2)

```

```

BEGIN
UPDATE articulos set descripcion=v_descripcion
WHERE cod_fabricante=n_codigo;
EXCEPTION
WHEN NO_DATA_FOUND THEN
DBMS_OUTPUT.PUT_LINE ('No encontrado el articulo con codigo: ' ||
n_codigo);
END;

```



3. Escribe un procedimiento para borrar un artículo de la tabla ARTICULOS. Añade el control de errores con las excepciones adecuadas.

```

CREATE OR REPLACE PROCEDURE borrar (v_articulo varchar2(50)) is
BEGIN
DELETE from articulos
WHERE v_articulo = articulo;
WHEN NO_DATA_FOUND THEN
DBMS_OUTPUT.PUT_LINE('No encontrado el artículo para borrar ');
END;

```



4. Obtén un procedimiento para consultar la tabla de ARTICULOS y recupere el nombre y

PVP para un código de artículo dado. Proporciona el control de errores adecuado.

```
CREATE OR REPLACE PROCEDURE consulta (codigo number(3)) is
BEGIN
SELECT nombre, pvp
FROM articulo
WHERE codigo = cod_fabricante;
WHEN NO_DATA_FOUND THEN
DBMS_OUTPUT.PUT_LINE('No encontrado el artículo para consultar');
END;
```



5. Escribe una función que devuelva la descripción de un artículo cuyo código se pedirá por teclado.

```
CREATE OR REPLACE PROCEDURE (codigo) is
BEGIN
SELECT articulo, peso, categoria, precio_venta, precio_costo,
existencias
FROM articulos
WHERE codigo = cod_fabricante;
WHEN NO_DATA_FOUND THEN
DBMS_OUTPUT.PUT_LINE('No encontrado el artículo');
END;
```

6. Crea una función para obtener el precio de un artículo descontando el IVA. Para ello, la función aceptará el PVP y el IVA y devolverá el precio sin IVA.

```
CREATE OR REPLACE PROCEDURE siniva ( v_precio NUMBER, v_iva
NUMBER) IS resultado NUMBER;
BEGIN
resultado = v_iva*v_precio/100
END siniva;
```

