

Gestión de bases de datos

PL/SQL

Tema 1

- Introducción.
- Estructura general bloque código anónimo.
- Estructura general funciones y procedimientos.
- Reglas escritura.
- Funciones de entrada y salida.
- Ejecución de código.

Oracle desarrolló un lenguaje procedimental para ampliar las funcionalidades de SQL, denominado **P**rocedural **L**anguage / **S**tructured **Q**uery **L**anguage, es decir, **PL / SQL**.

Un lenguaje procedimental es aquel con un número reducido de expresiones, las cuales se repiten y que suelen llamarse para su ejecución en funciones y procedimientos.

PL/SQL **soporta** el lenguaje de manipulación de datos (**dml**): select, insert, update y delete.

PL/SQL **no soporta** los lenguajes de definición de datos (ddl) ni de control de datos (dcl): create, alter, drop... grant, revoke.

¿Qué es un bloque de código anónimo?

- El bloque de código anónimo constituye la parte más elemental en PL / SQL.
- Son instrucciones que se pueden lanzar directamente desde la consola de SQL.
- Se ejecutan tras introducir el carácter /
- Este código no se guarda en la BD por lo que para reutilizarlo hay que volver a escribirlo.

Es habitual hacer llamadas a subprogramas desde un bloque de código anónimo.

El objetivo es que se ejecute código que se encuentra agrupado dentro de **funciones y procedimientos**.

Tema 1. Estructura general bloque código anónimo

La estructura de un bloque de código anónimo es la siguiente:

[DECLARE]

-- variables, cursores, excepciones, etc.

BEGIN

--sentencias SQL

--sentencias de control PL / SQL

[EXCEPTION]

--acciones a realizar cuando hay
errores

END;

/

Lo único obligatorio es BEGIN y END

[DECLARE]

En la primera parte, DECLARE, se van a declarar todas las variables, constantes, cursores, etc. que se necesitarán en el script.

BEGIN... END;

Entre BEGIN y END se escribirán:

- Secuencias: órdenes, asignaciones, llamadas a funciones, etc. **Cada secuencia termina siempre en punto y coma (;).**
- Alternativas: se evalúa una expresión y dependiendo de su valor se redirige el flujo del programa a una serie de instrucciones o hacia otras. Serán los condicionantes **IF/ELSE y CASE.**
- Bucles (iteraciones): repetición de instrucciones mientras se cumpla cierta condición o se finalice una. Serán los bucles **FOR, WHILE y LOOP.**

[EXCEPTION]

En esta zona se incluirán las excepciones.

Cuando se produce un error, este se analizará en esta parte del código PL/SQL.

Dependiendo de qué error sea, se podrá indicar en el código que haga algo concreto.

Existen por defecto numerosas **excepciones predefinidas** en el sistema, pero también se pueden crear **excepciones propias**.

[EXCEPTION]

El bloque **WHEN OTHERS** se encarga de manejar cualquier tipo de excepción que no exista predefinida ni previamente a esta de forma personalizada. Es por tanto el último caso en las excepciones.

Con este apartado de EXCEPCIONES se termina la ejecución de nuestro bloque de código anónimo. **No hay nada después.**

Como se indicaba anteriormente, cada instrucción va en una línea y **termina en punto y coma.**

Los comentarios se pueden introducir empleando dos formas:

- Para comentar una línea es habitual usar `--`
- Para comentar varias líneas, se abre el comentario con `/*` y se cierra en la última línea y tras la última palabra, con `*/`

Los bloques de código anónimo, las funciones y los procedimientos, siempre terminan con **END;**

Además, las **variables** que se usen dentro de un bloque de código anónimo, función o procedimiento, **deben declararse previamente**. Esto se hace en **DECLARE** (bloque de código anónimo) y en **IS** (procedimientos y funciones).

Tema 1. Funciones de entrada y salida

Con objeto de aprender este lenguaje, es conveniente hacer uso de la entrada y salida por consola para comprobar los resultados de nuestro código.

Para activar la salida en sqldeveloper, es necesario que la variable global SERVEROUTPUT está a ON. Esto se puede modificar en la configuración, o bien iniciar siempre nuestro código con: **SET SERVEROUTPUT ON;**

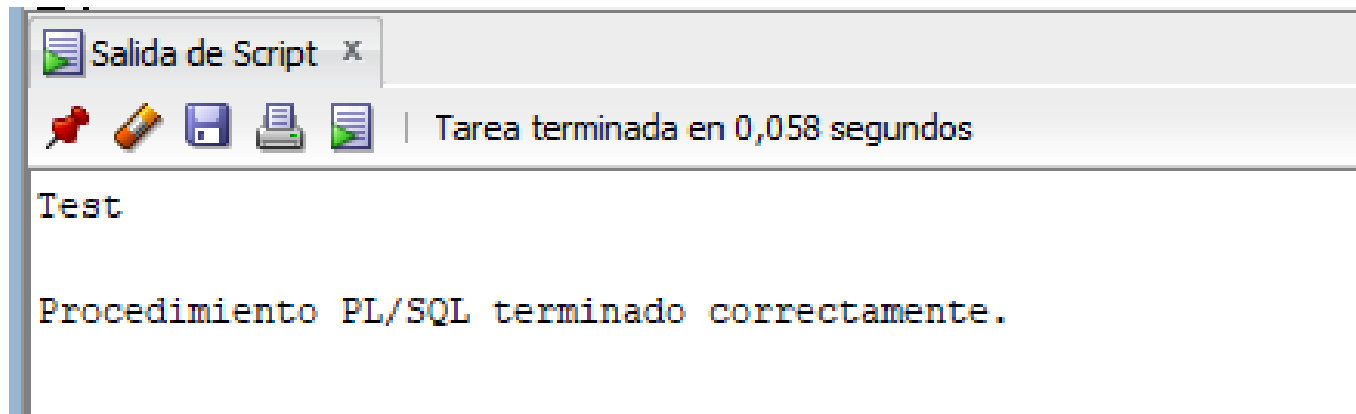
Para mostrar por pantalla una cadena de caracteres, un valor de una variable, etc. hay que usar:

dbms_output.put_line();

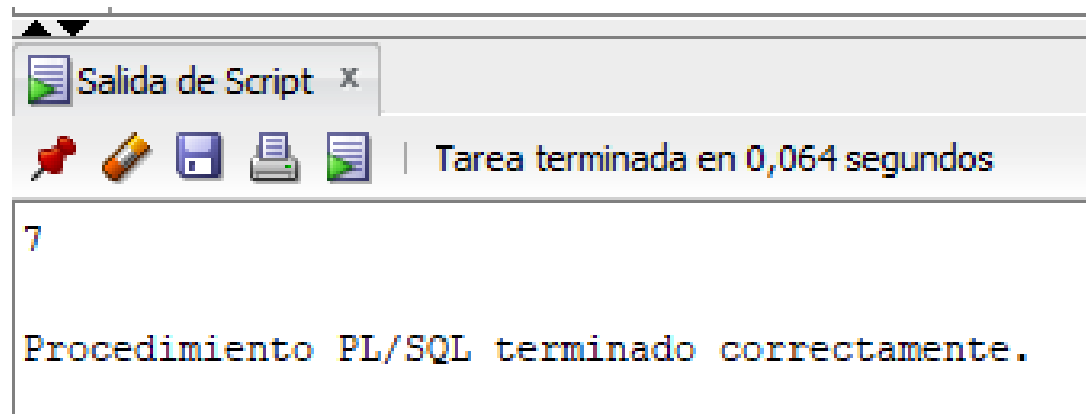
Indicaremos dentro de ese paréntesis anterior la cadena de texto, variable, etc. a mostrar.

Ejemplos:

```
SET SERVEROUTPUT ON;  
begin  
    dbms_output.put_line('Test');  
end;  
/
```

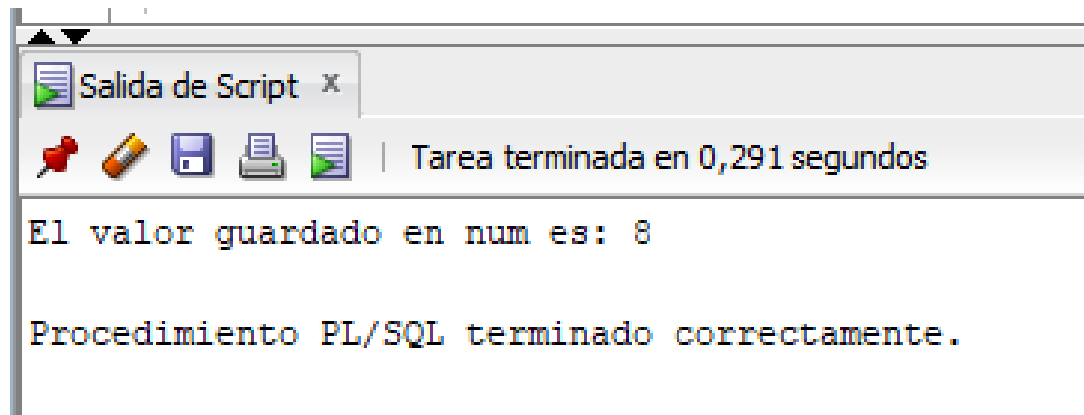



```
SET SERVEROUTPUT ON;  
declare  
    num int := 7;  
begin  
    dbms_output.put_line(num);  
end;  
/
```



Tema 1. Funciones de entrada y salida

```
SET SERVEROUTPUT ON;  
declare  
    num int;  
begin  
    num := 8;  
    dbms_output.put_line('El valor guardado en num es: ' ||  
num);  
end;  
/
```



Salida de Script x

Tarea terminada en 0,291 segundos

El valor guardado en num es: 8

Procedimiento PL/SQL terminado correctamente.

Tema 1. Funciones de entrada y salida

Para introducir un valor por el teclado, se usará el **símbolo &** en el valor de una variable para que se abra una ventana donde nos pida introducir un valor con el teclado, el cual **se guardará en esa variable**.

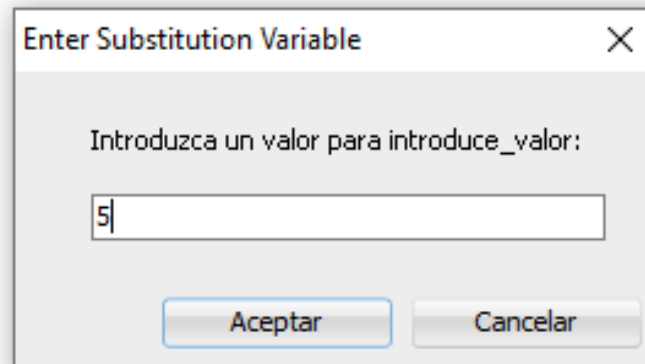
Este valor que se introduzca por el teclado puede ser textual, es decir, una cadena de texto (caracteres alfanuméricos) o bien un número (entero, flotante, etc.).

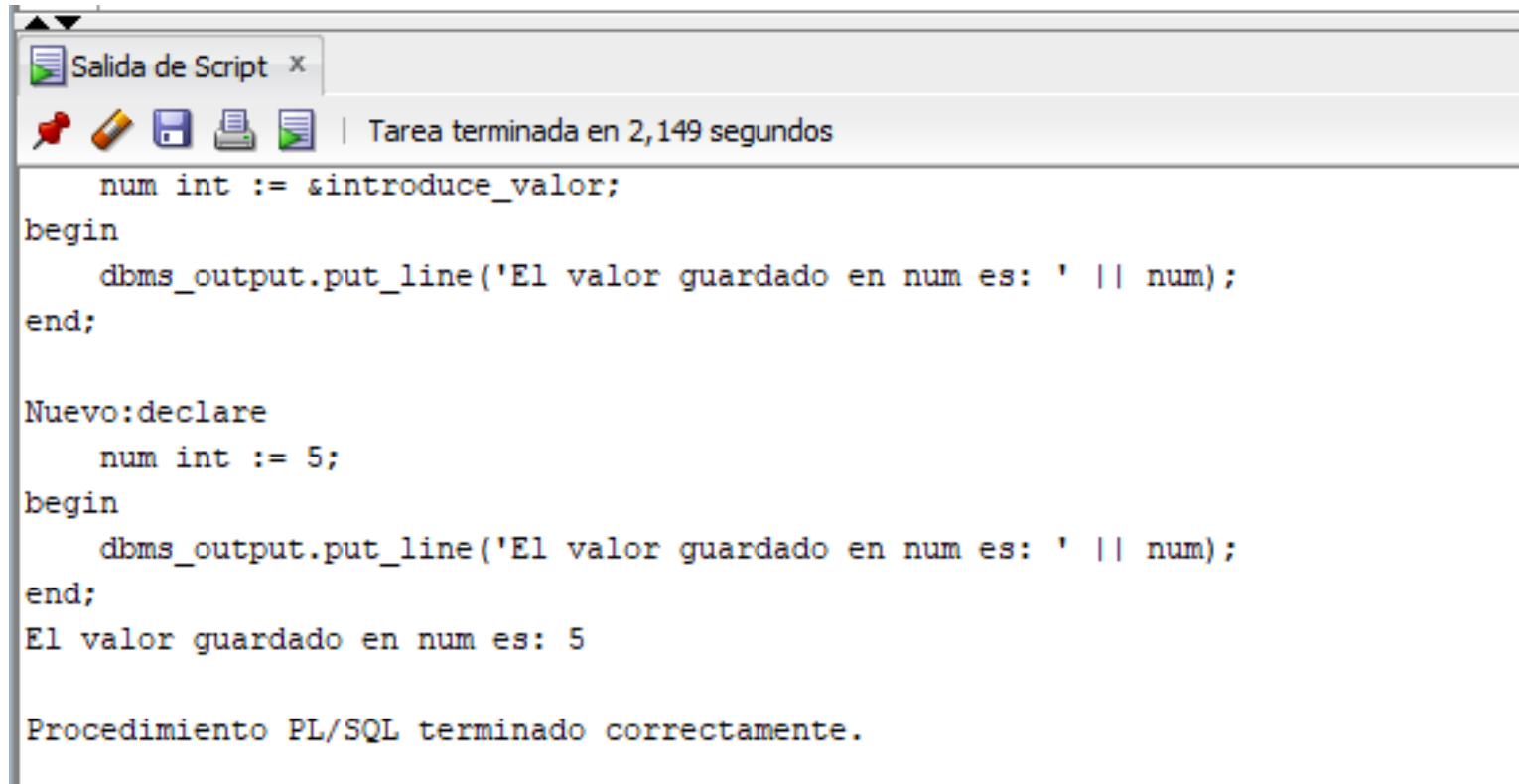
No se pueden dejar espacios ni introducir tildes, caracteres extraños, etc en la asignación de la variable, es decir, el texto que pongamos detrás del &.

Ejemplo: **num := &texto_a_mostrar;**

Ejemplos:

```
SET SERVEROUTPUT ON;  
declare  
    num int := &introduce_valor;  
begin  
    dbms_output.put_line('El valor guardado en num  
es: ' || num);  
end;  
/
```



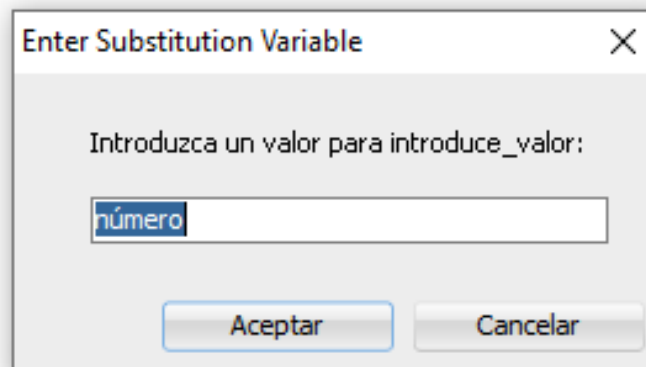


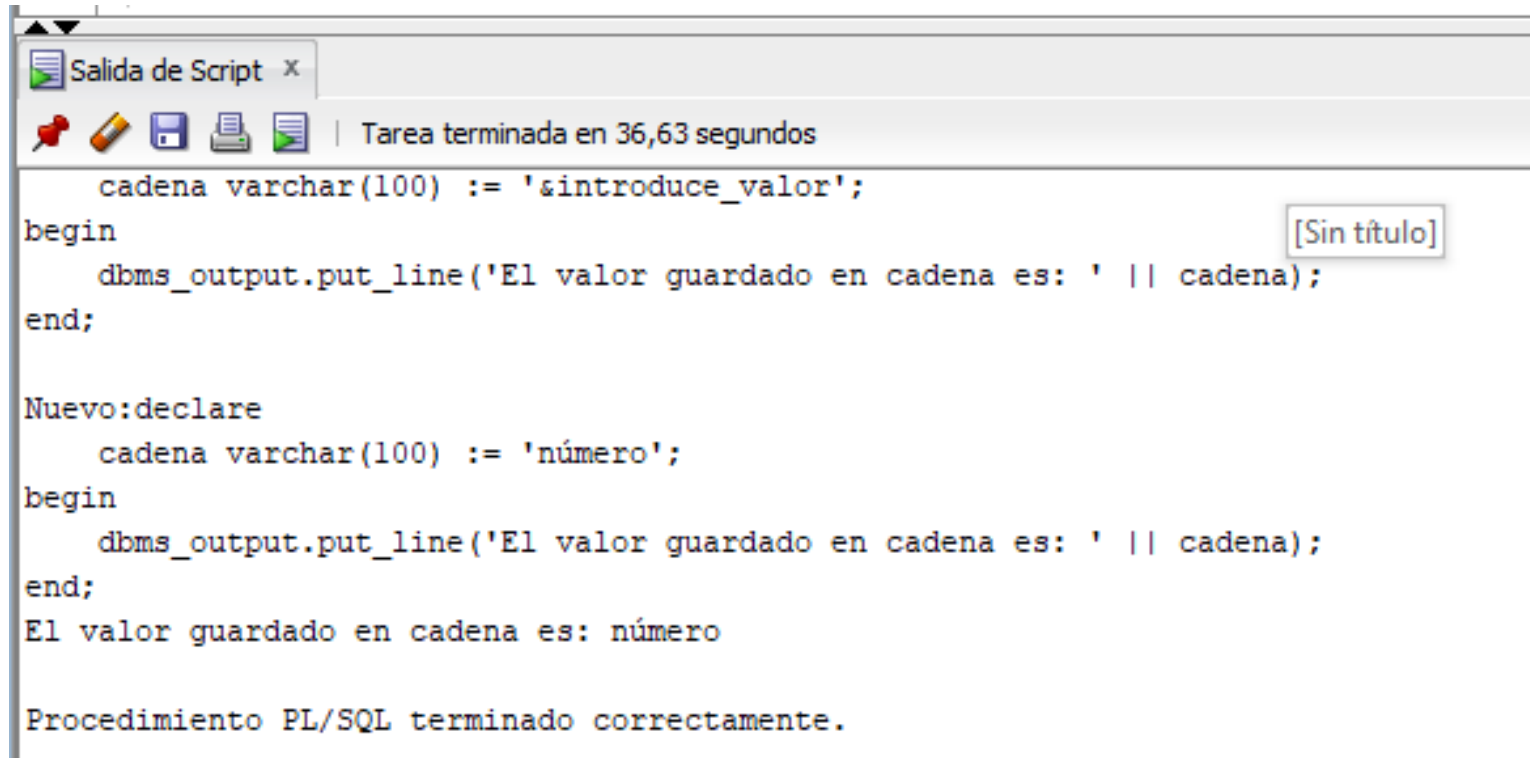
The screenshot shows a window titled "Salida de Script" with a toolbar containing icons for a pin, a pencil, a save icon, a printer, and a document. Below the toolbar, a status bar indicates "Tarea terminada en 2,149 segundos". The main area displays the following PL/SQL code and its output:

```
num int := &introduce_valor;  
begin  
    dbms_output.put_line('El valor guardado en num es: ' || num);  
end;  
  
Nuevo:declare  
    num int := 5;  
begin  
    dbms_output.put_line('El valor guardado en num es: ' || num);  
end;  
El valor guardado en num es: 5  
  
Procedimiento PL/SQL terminado correctamente.
```

Tema 1. Funciones de entrada y salida

```
SET SERVEROUTPUT ON;  
declare  
    cadena varchar(100) := '&introduce_valor';  
begin  
    dbms_output.put_line('El valor guardado en  
cadena es: ' || cadena);  
end;  
/
```





```
Salida de Script x
Tarea terminada en 36,63 segundos

  cadena varchar(100) := '&introduce_valor';
begin
  dbms_output.put_line('El valor guardado en cadena es: ' || cadena);
end;

Nuevo:declare
  cadena varchar(100) := 'número';
begin
  dbms_output.put_line('El valor guardado en cadena es: ' || cadena);
end;
El valor guardado en cadena es: número

Procedimiento PL/SQL terminado correctamente.
```


Es habitual crear **procedimientos** y **funciones**, las cuales realicen diferentes tareas, y a continuación llamarlas en un bloque de código anónimo para usarlas.

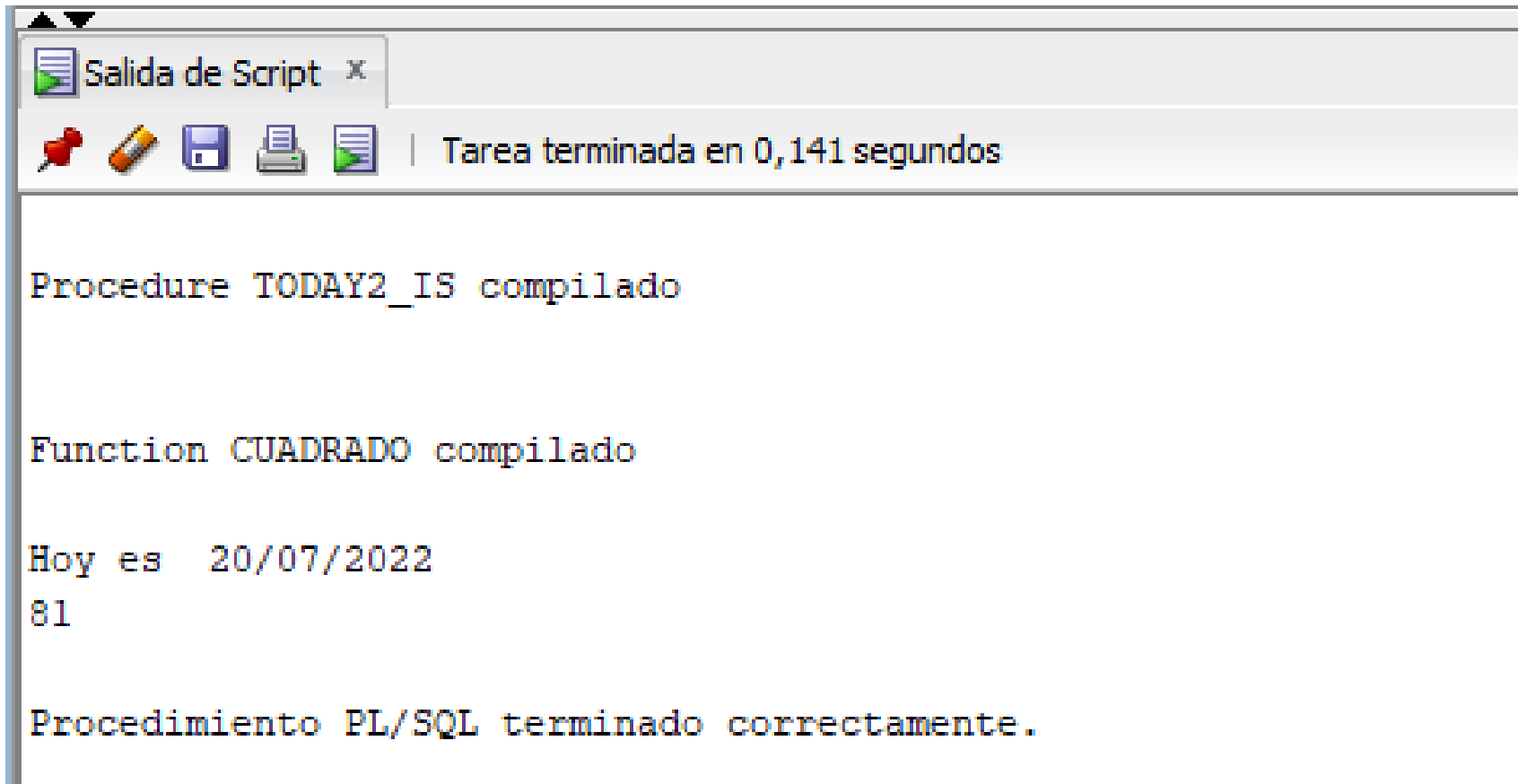
Con ello se podrán hacer cálculos, obtener datos, modificar información o borrar contenidos.

Tema 1. Ejecución de código

```
CREATE OR REPLACE PROCEDURE today2_is ( fecha DATE )  
IS  
BEGIN  
    DBMS_OUTPUT.PUT_LINE( 'Hoy es ' || TO_CHAR(fecha, ' DD/MM/YYYY') );  
END;  
/
```

```
CREATE OR REPLACE FUNCTION cuadrado (x NUMBER) RETURN NUMBER IS  
BEGIN  
    RETURN x*x;  
END;  
/
```

```
DECLARE  
BEGIN  
    today2_is(sysdate);  
    DBMS_OUTPUT.PUT_LINE(cuadrado(9));  
END;  
/
```



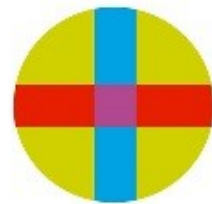
```
Salida de Script x
Tarea terminada en 0,141 segundos

Procedure TODAY2_IS compilado

Function CUADRADO compilado

Hoy es  20/07/2022
81

Procedimiento PL/SQL terminado correctamente.
```



CEU

*Centro de Estudios
Profesionales*

Fundación San Pablo Andalucía