

# Programación PL/SQL

El lenguaje de programación de bases de datos Oracle PL/SQL. Librerías y funciones estándar SQL y PLSQL.

## Anuncios en tutorial de programación PLSQL

miércoles, 29 de agosto de 2012

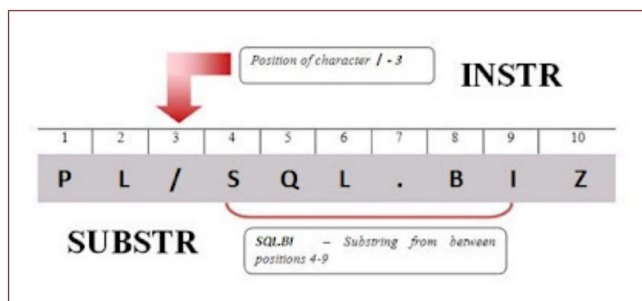
## Las funciones PL/SQL CONCAT, UPPER, LOWER, INITCAP, SUBSTR e INSTR (manejo de cadenas de caracteres I)



Twitter



Una vez que hemos visto cuales son los **tipos de datos para manejar cadenas de caracteres**, podemos pasar a estudiar las diversas funciones PL/SQL que permiten analizar el contenido de una cadena, combinarlo con otras cadenas, o simplemente cambiarlo de alguna manera controlada. Para ayudarnos con este tipo de requerimientos, la base de datos Oracle proporciona un conjunto bastante amplio de **funciones PLSQL que permiten manejar cadenas de caracteres** de una manera sencilla y cómoda.



En este capítulo hablaremos en concreto de las funciones **CONCAT**, **UPPER**, **LOWER**, **INITCAP**, **SUBSTR** e **INSTR**.

### Función CONCAT y el operador ||

Esta es una de las funciones más comúnmente utilizadas, ya que sirve para realizar una de las operaciones que con más frecuencia se tienen que llevar a cabo con las cadenas de caracteres, concatenarlas o unir las. El PL/SQL ofrece dos maneras distintas de realizar esta operación:

- La función **CONCAT**.
- El operador **||** (doble *pipe*), denominado por este motivo operador concatenación.

La función **CONCAT** admite dos cadenas de caracteres como argumentos y las devuelve unidas en una sola cadena. El operador **||** realiza la misma operación pero es mucho más sencillo de utilizar cuando necesitamos combinar más de dos cadenas de caracteres. No se me ocurre nada más sencillo que un ejemplo para demostrar lo que digo.

```
DECLARE
l_nombre VARCHAR2 (4) := 'Pepe';
```

## Sigue a programación PLSQL



## Búsquedas recomendadas en programación PL/SQL

## Recomendaciones


## Buscar en programación PLSQL

## Promociones en programación PL/SQL

Desarrollado por: ...

(<https://pbs.twimg.com/media/DP4e-LRWkAEb9Ap.jpg>)

¡Nuestro equipo ya está listo para sentir el poder del #cloud (<https://twitter.com/hashtag/cloud>)! ¡A volar a lo más alto juntos! #FlyToOracleCloud (<https://twitter.com/hashtag/FlyToOracleCloud>) <https://t.co/C4OoFf8bkn> (<https://t.co/C4OoFf8bkn>)

 (<https://blog.canalnet.es/wiki/605-15-registro-6-diciembre-doble-capitulo>) (<https://www.oracle.es>) [www.twitter.com/oracle-es](https://www.twitter.com/oracle-es)

LLEVATE ESTE WIDGET (<http://www.beon4u.com/es/mywidget>)

```
l_ape1 VARCHAR2 (3) := 'Jal';
l_ape2 VARCHAR2 (5) := 'Lopes';

BEGIN
  /* Uso de la función CONCAT */
  DBMS_OUTPUT.put_line (
    CONCAT (l_nombre,
      CONCAT (' ',
        CONCAT (l_ape1,
          CONCAT (' ', l_ape2)
        )
      )
    )
  );

  /* Uso del operador || */
  DBMS_OUTPUT.put_line (
    l_nombre
    || ' '
    || l_ape1
    || ' '
    || l_ape2
  );
END;
/
```

La salida de este script PLSQL seria:

```
Pepe Jal Lopes
Pepe Jal Lopes
```

Como podéis ver, la salida generada por las funciones DBMS\_OUTPUT es exactamente la misma, pero la forma de generarla es mucho más simple si utilizamos el operador || en lugar de la función CONCAT. De hecho, en toda mi larga experiencia programando en PL/SQL, nunca he encontrado ningún bloque de código en el que se utilice la función CONCAT.

### Funciones UPPER, LOWER e INITCAP

Estas funciones PLSQL sirven para cambiar las letras de una cadena de caracteres de minúsculas a mayúsculas o viceversa:

- **UPPER:** cambia todos los caracteres de una cadena a mayúsculas.
- **LOWER:** cambia todos los caracteres de una cadena a minúsculas.
- **INITCAP:** cambia el primer carácter de cada palabra a mayúsculas y el resto a minúsculas (las palabras dentro de una cadena de caracteres quedan delimitadas por los espacios en blanco o por cualquier carácter no alfanumérico).

A continuación os dejo un ejemplo:

```
DECLARE
  l_frase VARCHAR2 (25) := 'hoLA PeRico -un/kilo';
BEGIN
  DBMS_OUTPUT.put_line (UPPER (l_frase));
  DBMS_OUTPUT.put_line (LOWER (l_frase));
  DBMS_OUTPUT.put_line (INITCAP (l_frase));
END;
/
```

La salida de este script PLSQL seria:

```
HOLA PERICO -UN/KILO
hola perico -un/kilo
Hola Perico -Un/Kilo
```

## Favoritos de PL/SQL

[Ajedrez](#)

[Registro de dominios](#)

[Ganar dinero con tu web](#)

[Pepelu](#)

[Cursos de formación](#)

[Aplicaciones](#)

[Pregúntame](#)

[Juegos de ingenio](#)

**Ofertas de interés  
en programación  
PLSQL**

## Etiquetas en Programación PLSQL

[Bases de datos Oracle](#) [Librerías estándar](#)  
[PLSQL Optimización y tuning de bases de  
datos](#) [Tutorial PL/SQL](#) [Utilidades  
PLSQL](#)

## Función SUBSTR

Esta es una de las más utilizadas y útiles funciones PLSQL para manejar cadenas de caracteres. Sirve para extraer una subcadena de una cadena de caracteres más larga. Los parámetros que hay que pasar a la función SUBSTR son la cadena de caracteres de la que vamos a extraer la subcadena, la posición donde empieza dicha subcadena y el número de caracteres que tiene esta última.

Y como no hay nada mejor que un ejemplo, aquí os dejo uno:

```
DECLARE
  l_frase VARCHAR2 (25) := 'hoLA PeRico -un/kilo';
BEGIN
  /* Devuelve el primer carácter de la cadena */
  DBMS_OUTPUT.put_line (SUBSTR (l_frase, 1, 1));
  /* Devuelve el último carácter */
  DBMS_OUTPUT.put_line (SUBSTR (l_frase, -1, 1));
  /* Devuelve 4 caracteres desde la posición 8 */
  DBMS_OUTPUT.put_line (SUBSTR (l_frase, 8, 4));
  /* Devuelve toda la cadena desde la posición 6 */
  DBMS_OUTPUT.put_line (SUBSTR (l_frase, 6));
END;
/
```

La salida de este script PLSQL sería:

```
h
o
Rico
PeRico -un/kilo
```

Como habéis podido observar en el ejemplo, al utilizar la función SUBSTR es posible especificar un valor negativo para indicar el comienzo de la subcadena, en cuyo caso la base de dato Oracle lo que hace es contar hacia atrás desde el final de la cadena principal.

Por otro lado, si no se indica la longitud que debe tener la subcadena (es decir, el tercer parámetro de la función SUBSTR se deja en blanco), la base de datos Oracle lo que hace es devolver todos los caracteres restantes desde la posición especificada en el segundo parámetro.

## Función INSTR

La función INSTR sirve para determinar si una determinada cadena de caracteres aparece dentro de otra cadena. Esta función admite los siguientes parámetros:

- La cadena donde se busca.
- La cadena a ser buscada.
- La posición inicial de la búsqueda. Si nos se da un valor a este parámetro, la búsqueda empieza desde la primera posición. Si se asigna un valor negativo, la cuenta de posiciones empiezan desde el final de la cadena.
- El número de ocurrencia de la cadena a ser buscada. Si no se da un valor a este parámetro, la base de datos busca la primera ocurrencia de dicha cadena.

Aquí os dejo un ejemplo bastante ilustrativo:

```
DECLARE
  l_frase VARCHAR2 (25) := 'hoLA PeRico -un/kilo';
BEGIN
  /* Busca la posición de la primera "e" */
  DBMS_OUTPUT.put_line (INSTR (l_frase, 'e'));
  /* Busca la posición de la primera "o" */
```

```

    empezando desde la posición 5 */
    DBMS_OUTPUT.put_line ((l_frase, 'o', 5));
    /* Busca la posición de la primera "o"
    empezando desde la posición 19 contando
    desde el final */
    DBMS_OUTPUT.put_line ((l_frase, 'o', -19));
    /* Busca la tercera "o" empezando desde
    la posición 1 */
    DBMS_OUTPUT.put_line ((l_frase, 'o', 1, 3));
END;
/

```

La salida de este script PLSQL sería:

```

7
11
2
20

```

La función INSTR es una función a la que se le pueden dar numerosas utilidades. Por ejemplo, resulta muy sencillo crearse una **función que determine si una determinada cadena existe dentro de otra**. Sería algo tan sencillo como esto:

```

CREATE OR REPLACE FUNCTION existe_en_cadena (
    cadena IN VARCHAR2, subcadena IN VARCHAR2)
RETURN BOOLEAN
IS
BEGIN
    RETURN INSTR (cadena, subcadena) > 0;
END existe_en_cadena;
/

```

Por otro lado, utilizando las **funciones SUBSTR e INSTR**, es posible extraer de una cadena de caracteres, subcadenas que estén separadas por un determinado carácter separador. Seguro que el siguiente ejemplo os da una pista sobre lo que quiero decir.

```

DECLARE
    l_frase VARCHAR2 (25) := 'hoLA|PeRico|-un/kilo';
BEGIN
    /* Extrae la segunda palabra entre pipes "|" */
    DBMS_OUTPUT.put_line (
        SUBSTR(l_frase,
            INSTR(l_frase, '|')+1,
            INSTR(l_frase, '|', 1, 2)-INSTR(l_frase, '|')-1
        )
    );
END;
/

```

La salida de este script PLSQL sería:

```

PeRico

```

En el ejemplo podéis ver como obtenemos la segunda palabra de entre tres separadas por el carácter *pipe* "|". A vosotros os dejo pensar como se podrían sacar la primera y tercera palabras, a ver si alguien se siente tentado e incluye el código PL/SQL en algún comentario.

#### Artículos relacionados:

[Operaciones aritméticas con fechas y la función TRUNC del PL/SQL.](#)

Publicado en PLSQL por José Luis Pérez a las [7:45 p. m.](#)

Etiquetas: [Tutorial PL/SQL](#)

Compartir:



## o comentarios:

Publicar un comentario

[Entrada más reciente](#)

[Página principal](#)

[Entrada antigua](#)

### Archivo del PL/SQL

[agosto 2012 \(1\)](#)

### PL/SQL está publicado por



**José Luis Pérez**

[Seguir](#)

Si queréis conocerme un poco mejor podéis consultar [mi blog personal](#).

[Ver todo mi perfil](#)

Número de internáutas que han visitado este blog de programación PL/SQL:

Copyright © 2006-2017, Centroblogs - [Términos de uso y privacidad](#) - [Datos identificativos](#) - [Enlaces](#)