Parámetros

* Cuando llamamos a un procedimiento puede ser necesario 'enviarle' información.

Por ejemplo:

* Podemos querer saber las conexiones activas de un determinado usuario.
* Podemos querer saber qué usuarios tienen permiso de selección sobre determinada base de datos.
* Podemos querer saber quien cobra más que un determinado sueldo.

....

En cada una de las consultas anteriores vamos a querer hacer uso de un dato que debe ser suministrador por el usuario en el momento de llamar al procedimiento.

* Cada uno de los datos que enviamos a un procedimiento almacenado se denomina parámetro.

Los parámetros son variables cuyo valor se utiliza dentro del procedimiento almacenado.

Como norma general, cualquier modificación que hagamos sobre el valor del parámetro dentro del procedimiento almacenado no tiene efecto fuera de él.

Este tipo de parámetros se denominan **de entrada**.

Los parámetros de salida son parámetros que son utilizados por el procedimiento almacenado para 'informar' a quien llama y asignarle un valor a dicho parámetro dentro del procedimiento. Este valor será utilizado posteriormente por quien realizó la llamada, leyendo el valor del parámetro modificado. Este tipo de parámetro se denomina **de salida**.

Los parámetros que pueden llevar información al procedimiento, que pueden ser modificados dentro del procedimiento y que pueden ser utilizados por quien realizó la llamada al procedimiento se denominan **de entrada/salida**.

Debemos de tener cuidado con el nombre de los parámetros y que estos no coincidan con el nombre de las columnas de una tabla, ya que Mysql interpretará siempre que se trata del nombre del parámetro y nunca el de la columna.

Podríamos solucionarlo referenciando el nombre de la columna de la forma: nombre\_tabla.nombre\_columna

Aunque no existe una norma que indique como llamar a los parámetros, os recomiendo seguir la siguiente:

A los parámetros, definirlos con el prefijo: **p\_**nombreParámetro tipo

**Entrada**

Es el tipo por defecto si no indicamos nada en la definición del procedimiento.

**DELIMITER** \\

**CREATE** **PROCEDURE** nombre\_proc (**IN** nombre\_parametro tipo\_dato)

**BEGIN**

**END**//

**DELIMITER** ;

Sería lo mismo que no poner IN (tipo por defecto)

**DELIMITER** \\

**CREATE** **PROCEDURE** nombre\_proc (nombre\_parametro tipo\_dato)

**BEGIN**

**END**//

**DELIMITER** ;

Una vez definido podremos disponer de él dentro del procedimiento como si tuviéramos una variable local.

La diferencia es que desde quien llama al procedimiento, va a poder enviar un valor de la forma:

**CALL** nombre\_proc(100);

En este caso el parámetro debe estar definido como de tipo numérico (por ejemplo int)

O de la forma:

**SET** @dato = 100;

**CALL** nombre\_proc(@dato);

**Nota:** Recordad que este tipo de variables son de sesión y que son globales en la conexión actual a Mysql. Por lo tanto, podríamos acceder directamente a su valor y modificarlo dentro del procedimiento almacenado sin hacer uso de parámetros, pero a nivel de programación es más correcto emplear parámetros para aquellos procedimientos que lo necesitan. Además, vamos a poder llamar a procedimientos desde otros procedimientos, donde las variables son locales.

Podemos definir tantos parámetros como queramos, siempre que el número de datos a enviar se corresponda con el número de parámetros del procedimiento.

**DELIMITER** \\

**CREATE** **PROCEDURE** nombre\_proc\_varios\_param (nombre\_parametro1 tipo\_dato,nombre\_parametro2 tipo\_dato,....)

**BEGIN**

**END**//

**DELIMITER** ;

**CALL** nombre\_proc\_varios\_param(dato1,datos2,...);

**Salida**

En este tipo de parámetros, el procedimiento almacenado, al modificar su valor, este queda reflejado en el código desde donde se realizó la llamada.

En la definición del parámetro debemos añadir la palabra **OUT**.Por ejemplo:

**DELIMITER** \\

**CREATE** **PROCEDURE** nombre\_proc (**OUT** nombre\_parametro1 tipo\_dato)

**BEGIN**

.......

**SET** nombre\_parametro1 = 1; // Cualquier instrucción que modifique el valor, como **SELECT** **INTO**, **SET**,....

**END**//

**DELIMITER** ;

Al salir del procedimiento, el valor del parámetro será de 1.

Quien lo llama:

**CALL** nombre\_proc(@dato);

**SELECT** @dato; // El valor de la **variable** @dato es uno, ya que fue modificado por el procedimiento y es de salida.

Podemos utilizar cualquier combinación de parámetros de salida y de entrada en el mismo procedimiento.

Nota: Aunque veremos algún ejercicio con parámetros de salida aplicado a una variable de sesión, recordar que la variable de sesión tiene un alcance global y por tanto podría ser modificada directamente dentro del procedimiento.

**Entrada/Salida**

Los parámetros de Entrada/Salida son el resultado de aplicar los conceptos aprendidos para los de entrada y para los de salida.

Son parámetros a los que vamos a poder dar un valor inicial, llamar al procedimiento y que pueda hacer uso del valor enviado y dentro del procedimiento va a poder cambiar su valor y este será reflejado desde donde se llamó. Por ejemplo:

**DELIMITER** \\

**CREATE** **PROCEDURE** nombre\_proc (**INOUT** nombre\_parametro1 tipo\_dato)

**BEGIN**

**SELECT**

**FROM** TABLA

**WHERE** columna = nombre\_parametro1;

.......

**SET** nombre\_parametro1 = 1; // Cualquier instrucción que modifique el valor, como **SELECT** **INTO**, **SET**,....

**END**//

**DELIMITER** ;

Al salir del procedimiento, el valor del parámetro será de 1.

Quien lo llama:

**SET** @dato = 100; // El procedimiento podrá hacer uso del valor enviado en la **variable** @dato

**CALL** nombre\_proc(@dato);

**SELECT** @dato; // El valor de la **variable** @dato es uno, ya que fue modificado por el procedimiento y es de salida.

**Ejercicios propuestos de procedimientos con parámetros**

Crea los procedimientos indicados utilizando las formas vistas (SQL, PhpMyAdmin o MysqlWorkBench) y añadiendo en todos ellos un comentario de cual es su objetivo. Si no está claro, documenta también cada uno de los parámetros.

**Ejercicio 1:** Crea un procedimiento de nombre artistas\_getAnimalesPorNif que devuelva los animales que cuida un artista. Llevará como parámetro el nif de artista.

**Ejercicio 3:** Crea un procedimiento de nombre atracciones\_getListConAntiguedad que devuelva los datos de las atracciones que hayan comenzado hace un número de años con respecto a la fecha actual. Tendrás que hacer uso de alguna de las funciones Date Time. Intenta averiguar cual.

**Ejercicio 4:** Crea un procedimiento de nombre artistas\_getListMasAnimalesCuida que devuelva los datos del/os artista/s que cuidan a más animales de los indicados (parámetro que se le envía).

Pista: Como la consulta puede devolver más de un artista no podremos hacer uso de INTO....

**Ejercicio 5:** Crea un procedimiento de nombre atracciones\_getListPorFecha que devuelva los datos de las atracciones que han comenzado a partir de la fecha indicada.

Pista: Recordar que las fechas son tratadas como cadenas...y tener en cuenta el formato.

Añade una nueva atracción con la fecha de inicio actual.

Llama al procedimiento empleando la fecha actual menos 3 días (haz uso de la función DATE\_SUB y curdate)

**Ejercicio 6**

Crea un procedimiento de nombre artistas\_getNumAnimalesCuida al que se le pase el nif de un artista y que devuelva en forma de parámetro de salida a cuantos animales cuida.

**Ejercicio 7**

Crea un procedimiento de nombre pistas\_addAforo al que se le envíe como parámetros el nombre de la pista y una cantidad que representa el incremento del aforo.

El procedimiento debe devolver en el mismo parámetro el nuevo aforo de la pista.