BD UD6 Variables



Introducción

Como en cualquier otro lenguaje de programación vamos a tener la posibilidad de emplear variables. La diferencia con respeto de los ‘lenguajes tradicionales’ es que podemos emplear dichas variables para guardar resultados de consultas SQL o emplearlos como datos a enviar a órdenes SQL como INSERT, UPDATE,....

En MYSQL tenemos que considerar dos casos cuando empleamos variables:

* Las empleamos con un conjunto de instrucciones que guardaremos en un archivo con extensión .sql
* Las empleamos dentro de un procedimiento almacenado, función o trigger.

Variables en ficheros de código SQL

* Nota: Indicar que, como comenté antes, este punto no hace referencia a los bloques de código anónimo que se pueden crear en otros gestores de bases de datos relacional.

En este punto hago referencia sobre todo, a los archivos .sql que podemos guardar en disco y que no se guardan internamente en el gestor como pasa con los procedimientos almacenados y funciones.

* Este tipo de variables son referenciadas por Mysql como [variables de usuario (User-Defined Variables)](https://dev.mysql.com/doc/refman/8.0/en/user-variables.html).

Las variables usadas:

* No son necesario declararlas y las podemos emplearlas directamente.
* Empiezan con el símbolo @
* Para asignarles un valor empleamos la orden SET o INTO
* A estas variables se les denomina **variables de sesión**. Significa que son variables que se pueden acceder desde cualquier parte (procedimiento, función,...vienen a ser como variables globales en la programación tradicional) y tendrán un valor diferente en sesiones diferentes, entendiéndose una sesión como una conexión a Mysql. Lo veremos posteriormente.

Ejemplo

Veamos un ejemplo:

[Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente](https://wiki.cifprodolfoucha.es/index.php?title=Archivo:Mod_BD_Prog_Variables_1.jpg)

Como podéis comprobar, la variable de nombre @nombre tiene de valor ‘Pepe’.

Podemos emplear variables para guardar información de una orden SQL, pero debemos asegurarnos que solamente devuelve una fila. En este caso empleamos la cláusula INTO dentro del SELECT:

[Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente](https://wiki.cifprodolfoucha.es/index.php?title=Archivo:Mod_BD_Prog_Variables_2.jpg)

En este caso estamos empleando la variable @anhos en la que guardamos los años de un animal.

Podemos emplear varias variables a la vez:

[](https://wiki.cifprodolfoucha.es/index.php?title=Archivo:Mod_BD_Prog_Variables_3.jpg)

Fijarse como el INTO lleva dos variables. Cada columna del select (y en el orden en el que se encuentran) se va a corresponder con una variable del INTO (en el mismo orden que el select). El número de columnas y variables debe de ser la misma.

Podemos hacer uso de las funciones integradas del gestor para realizar cualquier modificación sobre los datos de las variables y después emplearlas en otras instrucciones SQL:

[Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente](https://wiki.cifprodolfoucha.es/index.php?title=Archivo:Mod_BD_Prog_Variables_4.jpg)

O por ejemplo:

[Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente](https://wiki.cifprodolfoucha.es/index.php?title=Archivo:Mod_BD_Prog_Variables_5.jpg)

Podemos emplear variables en cualquier orden SQL, no tiene por qué ser solo SELECT:

[Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente](https://wiki.cifprodolfoucha.es/index.php?title=Archivo:Mod_BD_Prog_Variables_6.jpg)

Fijarse que si este bloque de código lo ejecutáis dos veces, dará un error. Esto es debido a que las variables con @ son **variables de sesión**.

Este tipo de variables, que no llevan un tipo de dato asociado (son débilmente tipadas), su valor es visible en todas las pestañas que abráis de tipo SQL dentro de la sesión en la que estáis conectados.

Es decir, si abrís otra pestaña SQL y ejecutáis la orden:

[Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente](https://wiki.cifprodolfoucha.es/index.php?title=Archivo:Mod_BD_Prog_Variables_7.jpg)

Os saldrá el nombre del animal que guardasteis en la consulta anterior. Tendríais que abrir otra sesión, es decir, otra conexión a Mysql:

[Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente](https://wiki.cifprodolfoucha.es/index.php?title=Archivo:Mod_BD_Prog_Variables_8.jpg)

para comprobar que esa variable ya no tiene el valor asignado.

La explicación de por qué os da un error al ejecutar estas órdenes dos veces:

**SELECT** nombre

**INTO** @nombreAnimalSinArtista

**FROM** ANIMALES

**WHERE** nombre **NOT** **IN** (**SELECT** nombre\_animal

**FROM** ANIMALES\_ARTISTAS)

**LIMIT** 1;

**INSERT** **INTO** ANIMALES\_ARTISTAS

**VALUES** (@nombreAnimalSinArtista,'11111111A');

La primera orden SQL (la del SELECT) no devuelve nada (ya que no encuentra otro animal que no esté cuidado por un artista). Por lo tanto nunca se ejecuta la cláusula INTO y por lo tanto la variable @nombreAnimalSinArtista sigue guardando el valor anterior y por eso da un error de clave duplicada en el INSERT.

Las variables de sesión tienen que cumplir las siguientes características:

* Comenzar su nombre con el símbolo @, por ejemplo: @nombre
* Puede incluír cualquier carácter alfanumérico.
* Puede incluír los símbolos: ".", "\_", "$"
* Puede incluír otro tipo de símbolos (incluído espacio en blanco) a condición de que se escriba el nombre de la variable entre comillas: @'my var', @'my-var', @"my-var", or @'my-var'.
* No se distingue mayúsculas de minúsculas
* Longitud máxima: 64 caracteres.

Indicar que las variables que empiezan por **@@** son variables del sistema que dependiendo del ‘scope’ que tengan, pueden ser globales o de sesión. Las variables que son globales tienen el mismo valor en todas las sesiones de todos los usuarios. Por ejemplo, en el tema de transacciones vimos que para iniciar una transacción, una opción era emplear la variable del sistema autocommit:

[Interfaz de usuario gráfica, Aplicación, Tabla

Descripción generada automáticamente con confianza media](https://wiki.cifprodolfoucha.es/index.php?title=Archivo:Mod_BD_Prog_Variables_9.jpg)

Como se puede ver (opción Scope) esta variable puede ser Global o de Sesión. Para cambiar su valor podemos poner:

* SET SESSION autocommit = 0; -- Cambia su valor en la sesión actual.
* SET GLOBAL autocommit = 0; -- Cambia su valor a nivel global.

Más información sobre las variables del sistema: [https://wiki.cifprodolfoucha.es/index.php?title=Mysql\_Instalaci%C3%B3n#Variables\_del\_sistema](https://wiki.cifprodolfoucha.es/index.php?title=Mysql_Instalaci%C3%B3n" \l "Variables_del_sistema)

Variables locales dentro de un procedimiento, función o trigger

Nota: Indicar que las variables anteriores (las de sesión) pueden emplearse en cualquier sitio, incluído dentro de procedimientos almacenados, pero no es lo habitual.

Tenemos que ver como se crea un procedimiento almacenado, función o trigger. Dentro de ellos, las variables se definen haciendo uso de la orden [DECLARE](https://dev.mysql.com/doc/refman/8.0/en/declare-local-variable.html) de la forma:

**DECLARE** nombreVar tipoDato [**default** valor\_por\_defecto]

Mirar el enlace anterior para ver la sintaxis completa.

A diferencia de las variables usada en los bloques de código SQL:

* El nombre de las variables no empieza por @
* Se debe declarar un tipo asociado a la variable pero esto no impide que puedas asignarle un dato que no se corresponde con el tipo, haciendo la conversión (siempre que pueda, como por ejemplo, asignar a una variable de cadena un valor numérico, pero no se podría a la inversa, es decir, asignar un valor de cadena a una variable definida como numérica).
* Si no se inicializan, el valor de dichas variables por defecto es null.

Por ejemplo:

**DECLARE** numDepts int **default** 0;

Estamos declarando una variable de nombre numDetps, de tipo int y con valor por defecto de 0 (recordar que si no se pone default, el valor por defecto es null).

Podemos asignar un valor a una variable con el comando SET o INTO de alguna de las siguientes formas:

* **SET** variable := valor;
* **SET** variable = valor;
* SELECT count(\*)

**INTO numDepts**

FROM DEPARTAMENTOS

A diferencia de las variables de sesión, estas variables son locales al procedimiento, función o trigger donde estén definidas.

Los tipos de datos que podemos emplear son los mismos que empleamos para crear las columnas de las tablas.

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

* A las variables locales, definirlas con el prefijo: **v\_**nombreVariable tipo

**IMPORTANTE:** Cuando declaremos variables locales, debemos de tener en cuenta lo siguiente si las vamos a emplear con sentencias SQL:

* No declarar nombre de variables que coincidan con nombres de columnas. Darles un nombre diferente.
* Si empleamos la variable para guardar el resultado de una consulta o la vamos a emplear como dato para realizar un INSERT, UPDATE o DELETE, debemos de declararla con el mismo tipo y tamaño que la columna a la que va a hacer referencia en la orden SQL. Así, si quiero guardar el nombre de un animal con SELECT INTO en una variable, dicha variable debería estar declarada como varchar(50) que es el tipo de dato y tamaño de la columna nombre de la tabla ANIMALES.