U.T. V.- CREACIÓN DE OBJETOS EN MYSQL: PROCEDIMIENTOS.

1. Introducción.

- Un procedimiento almacenado es un conjunto de sentencias SQL que pueden almacenarse en el servidor.
  - Una vez creado, podrá ser utilizado por los usuarios autorizados dentro de cualquier script haciendo referencia a él.
- Un procedimiento almacenado puede contener cualquier número y tipo de instrucciones SQL excepto algunas de creación de determinados objetos, entre ellos las vistas.
- Por defecto, el procedimiento se asocia a la BD abierta. Si queremos asociarlo a otra BD deberemos preceder su nombre del de ésta
- Los procedimientos almacenados pueden ser particularmente útiles:
  - Cuando múltiples aplicaciones cliente se escriben en distintos lenguajes o funcionan en distintas plataformas, pero necesitan realizar la misma operación en la base de datos.
  - Cuando la seguridad es muy importante.
    - El uso de procedimientos almacenados permite crear un entorno seguro y consistente.
    - Su utilización permite impedir que los usuarios y programas de aplicación accedan a las tablas de la BD, autorizándoles únicamente a ejecutar algunos procedimientos almacenados.

- Los procedimientos almacenados pueden mejorar el rendimiento ya que disminuyen el tráfico de información entre el servidor y el cliente.
- El inconveniente es que aumenta la carga del servidor de la BD, ya que la mayoría del trabajo lo realiza él y no el cliente.
- En la actualidad MySQL permite dos tipos de procedimientos almacenados: procedimientos y funciones.

2.- Procedimientos.

Manual: *19.2. Sintaxis de procedimientos almacenados.*

2.1.- Creación:

```
CREATE PROCEDURE nombre ([lista-de-parámetros-formales])
    [BEGIN]
        sentencias;
    [END]
```

- Los parámetros deben declararse junto con su tipo y separados por comas.
  - Un procedimiento admite parámetros de entrada (**IN**), de salida (**OUT**). y de entrada/salida (**INOUT**).
    - un parámetro se considera de entrada por defecto.
  - Los parámetros de salida o de entrada/salida deben preceder a los de entrada.
- Como dijimos, dentro de un procedimiento pueden escribirse sentencias SQL de cualquier tipo excepto algunas de creación de objetos.
- Si el cuerpo del procedimiento contiene varias sentencias éstas deberán ir encerradas entre BEGIN y END.
  - Dado que MySQL considera que se ha terminado una sentencia al encontrarse con el carácter **;** deberemos

utilizar la cláusula delimiter para indicarle cuál va ser el carácter de fin de procedimiento.

2.2.- Llamada a un Procedimiento:

call nombreprocedimiento ([lista-de-parámetros-actuales]);

- Esta sentencia provoca la ejecución de las sentencias contenidas en el procedimiento nombreprocedimiento.
- La lista-de-parámetros-actuales deberá coincidir en número, orden y tipo con la lista-de-parámetros-formales.

Ejemplo 1.

```
a)   use compraventa;
     drop procedure if exists uno;
     create procedure uno (pnom varchar(15))
    select Descripción from categorias where nombrecategoría=pnom;
    call uno('Bebidas');

b) use compraventa;
    drop procedure if exists dos;
    create procedure dos (out pdescrip varchar(50),pnom varchar(15))
    set pdescrip:=(select Descripción from categorias where
    nombrecategoría=pnom);
    call dos(@descrip,'Bebidas');
    select @descrip;

c)    use compraventa;
      drop procedure if exists tres;
      create procedure tres (inout pnomdescrip varchar(50))
      set pnomdescrip:=(select Descripción from categorias where
    nombrecategoría=pnomdescrip);
```

```
set @descrip='Bebidas';
call tres(@descrip);
select @descrip;
```

d)
```
use compraventa;
drop procedure if exists cuatro;
delimiter //
create procedure cuatro (out pdescrip varchar(50),pnomprod
varchar(50))
begin
set @idcat=(select idcategoría from productos where
nombreproducto=pnomprod);
set pdescrip=(select Descripción from categorias where
idcategoría=@idcat);
end//
delimiter ;
call cuatro(@descrip,'Té Dharamsala');
select @descrip;
```

2.2.- Modificación y Borrado de Procedimientos.

- Sintaxis de modificación:

  ALTER PROCEDURE [IF EXISTS] nombreproc;

  - Esta sentencia permite modificar el procedimiento nombreproc.

- Sintaxis de borrado:

  DROP PROCEDURE [IF EXISTS] nombreproc;

  - Esta sentencia borra el procedimiento nombreproc.

- La cláusula IF EXISTS evita un error si el procedimiento no existía.

3.- Variables locales.

Manual: *19.2.9. Variables en procedimientos almacenados.*

- Hay que tener en cuenta que las variables de usuario existen mientras dure la sesión en la que se crearon.
- Si queremos utilizar variables dentro de un procedimento que sólo existan durante su ejecución, deberemos declararlas "locales al procedimiento".
- El nombre de las variables locales no debe ir precedido del carácter **@**.
- Las variables locales deben declararse al comienzo del procedimiento mediante la sentencia

  declare nombrevariablelocal tipodedatos;

- Una vez declarada, una variable local se utiliza igual que una variable de usuario.

Ejercicio.

a) Comprobar que la variable de usuario creada dentro del procedimiento "cuatro" del ejemplo 1 existe fuera del procedimiento.

b) Crear el procedimiento "cuatrobis", análogo al procedimiento "cuatro" utilizando una variable local en vez de la variable de usuario. Comprobar que la variable local no existe fuera del procedimiento.

4.- Sentencias de Control del Flujo de Ejecución.

Manual: *19.2.12. Constructores de control de flujo.*

- Dentro del cuerpo de un procedimiento almacenado pueden utilizarse diferentes sentencias de control del flujo

de ejecución, en particular la sentencia alternativa IF y la repetitiva WHILE.

4.1.- Sentencia Alternativa IF.

- Sintaxis:

```
IF expresión_lógica
        THEN
                [BEGIN]
                        sentencias;
                [END]
        [ELSE
                [BEGIN]
                        sentencias;
                [END]]
        END IF;
```

4.2.- Sentencia Repetitiva WHILE.

- Sintaxis:

```
WHILE expresión_lógica
        DO
                [BEGIN]
                        sentencias;
                [END]
        END WHILE;
```