# TEMA 6.

## PROGRAMACIÓN EN BASES DE DATOS.

SESIÓN 2

# 3.- Desarrollo de procedimientos almacenados

- Un **procedimiento o PROCEDURE** es una rutina formada por un conjunto de instrucciones SQL y:
  - Tiene un determinado nombre formado por una combinación de caracteres alfanuméricos y cualquiera de estos (. $ _).
  - Puede recibir valores al ser llamado a ejecución a través de parámetros encerrados entre paréntesis.
  - Puede devolver valores a través de parámetros encerrados entre paréntesis.
  - Un procedimiento se crea con la instrucción **CREATE PROCEDURE**.
  - Un procedimiento se ejecuta con la instrucción **CALL nombreProc (params)**

# 3.- Desarrollo de procedimientos almacenados

- Todo procedimiento queda asociado a la base de datos abierta cuando se creó el procedimiento.
- De esta forma, podemos ejecutar un procedimiento asociado a una base de datos distinta a la que tenemos abierta especificando, en la llamada al procedimiento, un **cualificador** de la base de datos, de la forma:

  ***CALL nomBASEDATOS.nomProc (parámetros)***.

- Si ejecutamos

  **CALL nomProc(parámetros),**

- será necesario que el procedimiento se encuentre en la base de datos que tengamos abierta.

# 3.- Desarrollo de procedimientos almacenados

- Cuando creas un procedimiento:
  - Si está bien, lo almacena pero no lo ejecuta.
  - Si está mal, te dará mensaje de error diciendo donde está el error.
- Los nombres del procedimiento no se pueden repetir.
- **Sintaxis para crear un procedimiento:**

**CREATE PROCEDURE *NomProc* (*parametros*) [*caracteristicas* ...]**

**BEGIN**

***Cuerpo_procedimiento***

**END**

# 3.- Desarrollo de procedimientos almacenados

***Ejemplo 1 de creación y ejecución de un procedimiento***

```
Delimiter//
CREATE PROCEDURE listados()
BEGIN
SELECT * FROM clientes;
SELECT * FROM automoviles;
END//

CALL LISTADOS()//
```

- O bien:

```
DELIMITER;
CALL LISTADOS();
```

# 3.- Desarrollo de procedimientos almacenados

- El delimitador de final de instrucciones de SQL es el punto y coma (;).
- Las instrucciones del cuerpo de un procedimiento deben terminar con punto y coma.
- Si mantenemos el delimitador punto y coma, al escribir las instrucciones dentro del procedimiento, intentaría ejecutarlo, y no lo hemos finalizado.
- Para poder crear procedimientos, tendremos que cambiar temporalmente, antes de empezar a crearlos, el carácter delimitador o finalizador de instrucciones SQL en MySQL.
- Para cambiar el carácter delimitador se usa la instrucción **DELIMITER**. Por ejemplo, para hacer que el delimitador de instrucciones sea '//', habrá que ejecutar:
  - ***DELIMITER //***

# 3.- Desarrollo de procedimientos almacenados

- ***Ejemplo 2:*** Crear y ejecutar un procedimiento **numcontratos** que recibe en un parámetro de entrada la matrícula de un coche y, a continuación, muestra las características del coche y devuelve en un parámetro de salida el número de contratos realizados sobre ese coche.

```
CREATE  PROCEDURE numcontratos(IN m CHAR(7), OUT c INT)
BEGIN
    SELECT * FROM automoviles WHERE matricula=m;
    SELECT count(*) INTO c FROM contratos WHERE matricula=m;
END//
```

- *Llamada al procedimiento creado:*

```
SET @Num=0//
CALL numcontratos('3273BGH', @Num)//
SELECT @Num//
```

# 3.- Desarrollo de procedimientos almacenados

- *Elementos de la sintaxis de la instrucción CREATE PROCEDURE*
- ***Parámetro*** tiene la sintaxis:

  **[ IN | OUT | INOUT ]** ***NomParam tipo***

- ***tipo*****:**

  Cualquier tipo de dato MySQL

- ***característica*****:**

  LANGUAGE SQL | [NOT] DETERMINISTIC | SQL SECURITY {DEFINER | INVOKER} | COMMENT '*string*'

- ***cuerpo_procedimiento*****:**

  Instrucciones SQL para realizar la tarea.

# 3.- Desarrollo de procedimientos almacenados

- Los parámetros declarados en un procedimiento se pueden usar dentro del  procedimiento.
- Si un parámetro ha sido declarado IN, no se le puede asignar un valor dentro del procedimiento, aunque si que se podría consultar su valor. Si tenemos:

  **CREATE PROCEDURE ejemplo (IN num INT)**

- No podríamos usar esta instrucción dentro del procedimiento:

  **SELECT count(*) INTO num FROM contratos;**

# 3.- Desarrollo de procedimientos almacenados

- Si un parámetro es declarado OUT, no se puede usar ese parámetro para consultar su valor, si para modificarlo. Si tenemos:

  **CREATE PROCEDURE ejemplo (OUT num INT)**

- No podríamos usar esta instrucción dentro del procedimiento:

  **SELECT count(*) FROM contratos WHERE numcontrato=num;**

- En cambio, un parámetro INOUT podríamos usarlo tanto para lectura como para escritura.

# 3.- Desarrollo de procedimientos almacenados

- ***Variables locales:***
- Además de los parámetros, en un procedimiento podemos declarar y usar **variables locales.**
- Estas variables locales sólo tienen existencia mientras se ejecuta el procedimiento, después quedan destruidas.
- La declaración de variables debe estar dentro del bloque BEGIN .... END.
- Para definir o declarar cualquier variable se usa la instrucción:

  ***DECLARE nombre tipo[DEFAULT valor];***
- Donde tipo es cualquiera de los tipos admitidos por MySQL.
- Para modificar el valor de una variable o de un parámetro con el operador de asignación =, debe usarse la instrucción:

  ***SET variable=expresión;***

# 3.- Desarrollo de procedimientos almacenados

- ***Ejemplo 3 de uso de variables locales:*** Crear un procedimiento **usovariable** que lista los vehículos con menos de 2500 kilómetros y, después, los vehículos con menos kilómetros que los anteriores más 5000 kilómetros.

```
CREATE  PROCEDURE usovariable()
 BEGIN
 DECLARE a INT;
 SET a=2500;
SELECT * FROM automoviles WHERE kilometros<a;
 SET a=a+5000;
SELECT * FROM automoviles WHERE kilometros<a;
 END//
```

# 3.- Desarrollo de procedimientos almacenados

- ***Ejemplo 4:***
- Realiza un procedimiento que recibe la matrícula de un automóvil y escribe u obtiene:
  - La marca y modelo del automóvil.
  - El número de contratos de alquiler realizados para el automóvil.
  - El número de clientes que han alquilado ese automóvil.
  - Los nombres de los usuarios que han alquilado ese automóvil.

Se puede usar la hoja 2 para explicar la parte de procedimientos almacenados, en lugar de ver la teoría.
Hoja 3 y hoja 4.

```
create procedure ejemplo4(in mat char(7))
begin
  SELECT marca,modelo FROM automoviles WHERE matricula=mat;
  SELECT count(*) FROM contratos WHERE matricula=mat;
  SELECT count(DISTINCT dnicliente) FROM contratos WHERE matricula=mat;
  SELECT DISTINCT nombre,apellidos FROM clientes INNER JOIN contratos ON dnicliente = dni
WHERE matricula=mat;
END
```