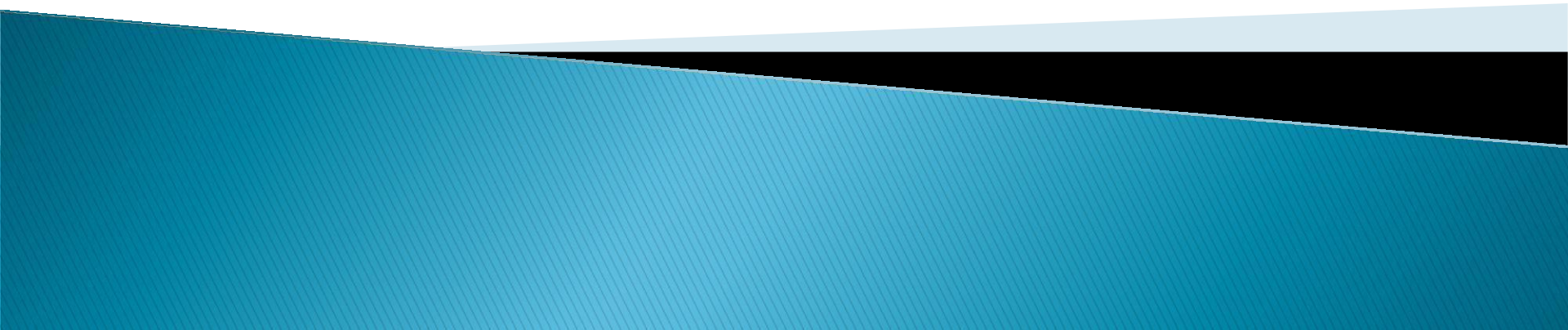


Bases de datos

Capítulo 6

Programación de Bases de Datos



Procedimientos y funciones

Creación de PROCEDIMIENTOS

CREATE PROCEDURE nombre_procedimiento ([parámetros])
instrucciones

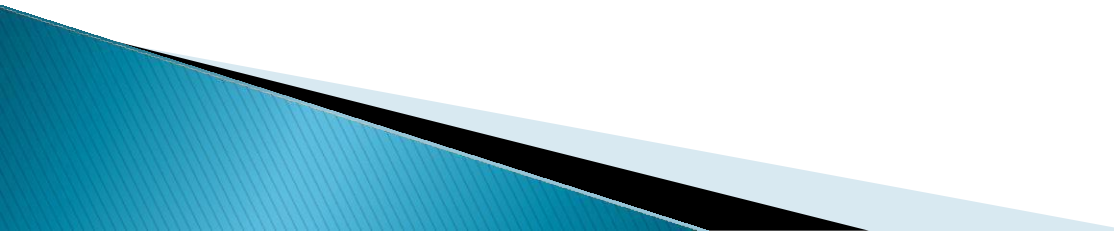
parametro: [IN | OUT | INOUT] nombre_parametro tipo

tipo: cualquier tipo de dato de MySQL

Creación de FUNCIONES

CREATE FUNCTION nombre_función ([parámetros])
RETURNS tipo
instrucciones

******La sintaxis completa se puede consultar en [CREATE PROCEDURE](#)

- La cláusula **RETURNS** puede especificarse sólo con **FUNCTION**, donde es **obligatoria**
 - Se usa para indicar el tipo de retorno de la función.
 - El cuerpo de la función debe contener un comando **RETURN valor**.
- 

Ejecución de un procedimiento (CALL)

Para llamar a un procedimiento almacenado debemos hacer uso de la orden SQL CALL

CALL nombre_procedimiento ([parámetros])

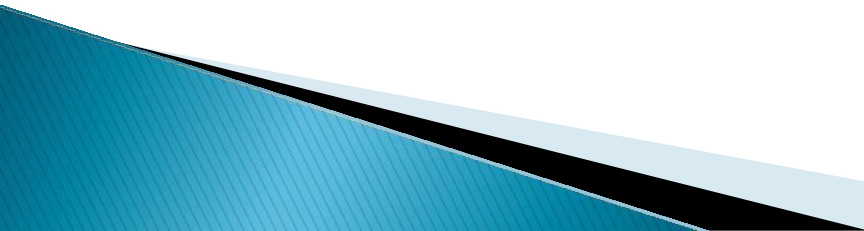
El comando CALL invoca un procedimiento definido previamente con CREATE PROCEDURE.

CALL puede pasar valores al procedimiento usando parámetros declarados como IN o INOUT

Tabla de ejemplo

```
CREATE DATABASE veterinario;  
USE veterinario;
```

```
CREATE TABLE mascotas (  
    nombre VARCHAR(10),  
    fechaNacim DATE NOT NULL,  
    propietario VARCHAR(40) NOT NULL,  
    PRIMARY KEY(nombre)  
);
```



USE veterinario;

INSERT INTO mascotas

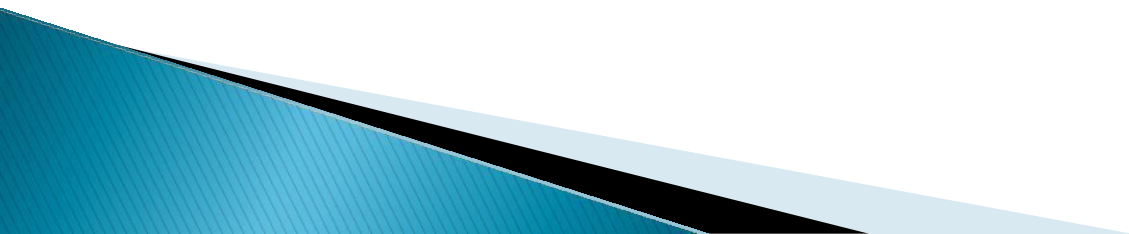
VALUES('Cuqui', '2012-05-20', 'Pedro López');

INSERT INTO mascotas

VALUES('Sultán', '2014-11-07', 'Ana Vals');

INSERT INTO mascotas

VALUES('Chati', '2021-09-13', 'Pedro López');



Ejemplos

- **Crear** un procedimiento para ver todas las mascotas:

```
USE veterinario;
```

```
CREATE PROCEDURE pa_mascotas_lista( )  
    SELECT * FROM mascotas;
```

- **Ejecutar** el procedimiento almacenado:

```
CALL pa_mascotas_lista();
```

- Crear un procedimiento para saber el número de mascotas:

```
CREATE PROCEDURE pa_mascotas_cantidad()  
    SELECT COUNT(*) FROM mascotas;
```

Eliminación de procedimientos

DROP {PROCEDURE | FUNCTION} [IF EXISTS]
nombre_procedimiento

La cláusula IF EXISTS evita que ocurra un error si la función o procedimiento no existe.

Eliminar un procedimiento almacenado y crearlo de nuevo:

```
USE veterinario;  
DROP PROCEDURE IF EXISTS pa_mascotas_lista;  
CREATE PROCEDURE pa_mascotas_lista()  
    SELECT * FROM mascotas;
```


Consultar Procedimientos

- Mostrar código:

SHOW CREATE PROCEDURE pa_mascotas_lista \G

- Detalles de un procedimiento almacenado:

SHOW PROCEDURE STATUS

LIKE 'pa_mascotas_lista' \G

- Detalles de todos los procedimientos almacenados en la base de datos:

SHOW PROCEDURE STATUS

WHERE Db=' veterinario' \G

****** Consultar sintáxis en [SHOW PROCEDURE STATUS](#)

Sentencia BEGIN ... END

BEGIN

lista de instrucciones

END

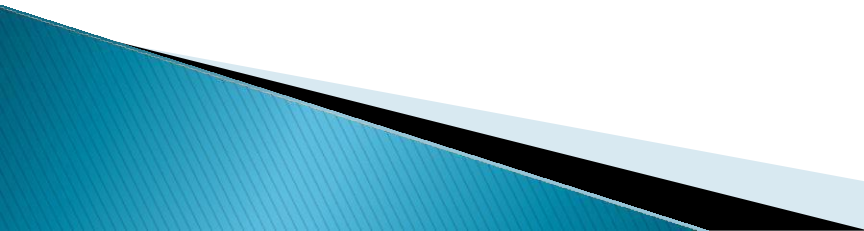
Los procedimientos almacenados pueden contener varias sentencias. En este caso es obligatorio usar el comando compuesto BEGIN ... END .

Delimitadores (DELIMITER)

- El comando DELIMITER se usa para cambiar el delimitador del comando de ; a // mientras se define el procedimiento .
- Esto permite **diferenciar entre** el ; que finaliza una **sentencia** del cuerpo del procedimiento **y** el que marca **el final del procedimiento**.

Ejemplo

```
USE veterinario;  
DROP PROCEDURE IF EXISTS pa_mascotas_lista2;  
DELIMITER //  
CREATE PROCEDURE pa_mascotas_lista2()  
BEGIN  
    SELECT *FROM mascotas;  
    SELECT count(*)    FROM mascotas;  
END//  
DELIMITER ;
```



Variables en archivos .sql

Este tipo de variables son referenciadas por Mysql como [variables de usuario \(User-Defined Variables\)](#).

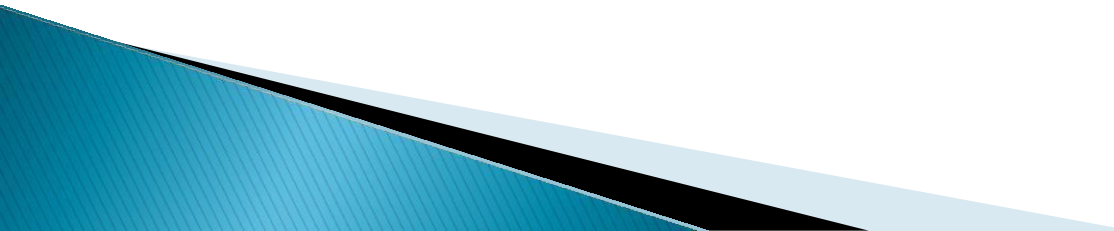
- No es necesario declararlas.
- Empiezan con el símbolo @
- No se distingue mayúsculas de minúsculas

Se les denomina **variables de sesión**:
su existencia se limita al tiempo que se permanece conectado a MySQL

Ejemplo:

```
SET @propietario =(SELECT propietario  
                     FROM mascotas  
                     WHERE nombre = "Cuqui");
```

```
SELECT nombre, fecha_Nacim  
FROM mascotas  
WHERE propietario = @propietario;
```



Declarar variables (DECLARE)

En los procedimientos las **variables locales** se definen haciendo uso de la orden DECLARE:

DECLARE nombre_variable tipo [DEFAULT valor]

DEFAULT: proporciona un valor por defecto para la variable. Puede ser una expresión o una constante.

Si la cláusula **DEFAULT** no está presente, **el valor inicial es NULL**.

```
DECLARE num_mascotas INT DEFAULT 0;
```

Declarar variables (DECLARE)

IMPORTANTE:

No declarar nombre de variables que coincidan con nombres de columnas.

Darles un nombre diferente.



Asignar valores a variables

- Directamente (**Sentencia SET**)

SET nombre_variable = expresión

Ejemplo:

```
SET num_mascotas=0;
```

- Tomando su valor de una consulta (**sentencia SELECT ... INTO**)

SELECT nomb_columna **INTO** nombre_variable

Ejemplo:

```
SELECT COUNT(*)  
INTO num_mascotas FROM mascotas;
```

Ejemplo con variables

Mostrar el número de mascotas usando una variable:

```
USE veterinario;
```

```
DROP PROCEDURE IF EXISTS pa_mascotas_cantidad2;
```

```
DELIMITER //
```

```
CREATE PROCEDURE pa_mascotas_cantidad2()
```

```
BEGIN
```

```
    DECLARE numero INT;
```

```
    SELECT COUNT(*) INTO numero FROM mascotas;
```

```
    SELECT numero;
```

```
END//
```

```
DELIMITER ;
```



Uso de Parámetros. Ejemplos.

- Procedimiento para obtener una lista de mascotas cuyo nombre comience con una determinada letra (**parámetro de entrada**)

USE veterinario;

DROP PROCEDURE IF EXISTS pa_mascotas_buscar;

CREATE PROCEDURE pa_mascotas_buscar (**letra CHAR(2)**)

 SELECT * FROM mascotas

 WHERE nombre LIKE letra;

Para buscar mascotas cuyo nombre comienza con la letra C:

CALL pa_mascotas_buscar(**'C%'**);

- Procedimiento que también calcula el número de mascotas encontradas (**parámetro de salida**)

```
USE veterinario;  
DROP PROCEDURE IF EXISTS pa_mascotas_buscar2;  
DELIMITER //  
CREATE PROCEDURE pa_mascotas_buscar2( IN letraCHAR(2),  
                                     OUT num_mascotas INT )  
BEGIN  
    SELECT * FROM mascotas WHERE nombre LIKE letra;  
    SELECT COUNT(*) INTO num_mascotas  
    FROM mascotas  
    WHERE nombre LIKE letra;  
END//  
DELIMITER ;
```

Llamada a este procedimiento:

```
CALL pa_mascotas_buscar2('C%', @cantidad);
```

Mostrar el número de mascotas:

```
SELECT @cantidad;
```

Funciones almacenadas

Función que devuelve el número de animales:

USE veterinario;

DROP FUNCTION IF EXISTS fa_mascotas_cantidad;

DELIMITER //

CREATE FUNCTION fa_mascotas_cantidad() RETURNS INT
BEGIN

 DECLARE num_animales INT;

 SELECT COUNT(*) INTO num_animales FROM mascotas;

 RETURN num_animales;

END//

DELIMITER ;

select fa_mascotas_cantidad();

set @n_mascotas = fa_mascotas_cantidad();

Estructuras de Control

Controlan el flujo del programa y la toma de decisiones.

Hay dos tipos:

- **condicionales:** se ejecuta un bloque de sentencias u otro.
- **iterativas o *bucles*:** la decisión que se toma es si repetir la ejecución de un bloque de instrucciones o no repetirlo.

Sentencia IF

```
IF condicion
  THEN  sentencia-1
    [ELSEIF condicion THEN sentencia] ...
  [ELSE  sentencia_2]
END IF
```

Si condición es cierta se ejecuta la sentencia_1.

Si no se cumple la condición se ejecuta la sentencia_2 de la cláusula ELSE.

Ejemplo:

```
IF num_cta=17
  THEN
    SET tipoUsuario='Admin';
  ELSE
    SET tipoUsuario='Invitado';
END IF;
```

****** [Documentación oficial de MySQL](#)

Sentencia CASE

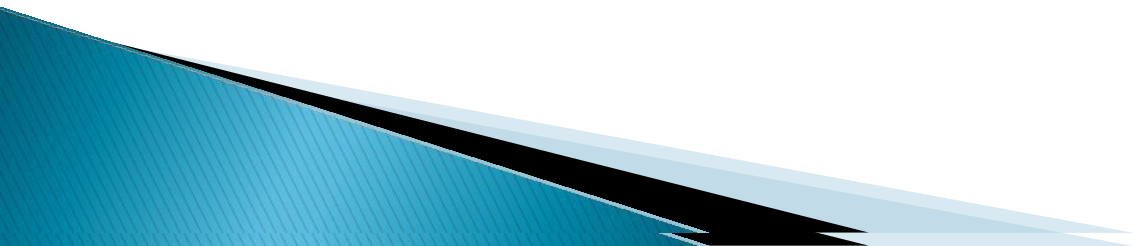
CASE

```
WHEN condicion_1 THEN sentencia_1  
[WHEN condicion_n THEN sentencia_n] ...  
[ELSE sentencia]
```

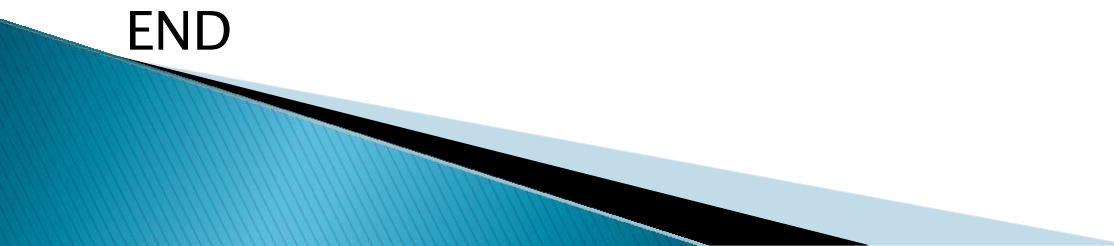
END CASE

Si una condicion se evalúa y es cierta, se ejecuta la sentencia correspondiente.

Si no coincide ninguna condición de búsqueda, se ejecuta el comando de la cláusula ELSE .




```
CREATE PROCEDURE pa_cases(IN numero INT)
BEGIN
  CASE
    WHEN numero >= 0 AND numero < 5 THEN
      SELECT "Suspenso" as nota;
    WHEN numero >= 5 AND numero < 6 THEN
      SELECT "Suficiente" as nota;
    WHEN numero >= 6 AND numero < 7 THEN
      SELECT "Bien" as nota;
    WHEN numero >= 7 AND numero < 9 THEN
      SELECT "Notable"as nota;
    WHEN numero >= 9 AND numero <= 10 THEN
      SELECT "Sobresaliente" as nota;
    ELSE SELECT "Calificacion erronea" as nota;
  END CASE;
END
```



CASE nombre_variable

WHEN valor **THEN** sentencias
[**WHEN** valor **THEN** sentencias] ...
[**ELSE** sentencias]

END CASE

```
CREATE PROCEDURE pa_cases1 (IN p1 INT)
BEGIN
    DECLARE var int ;
    SET var = p1 +2 ;
    CASE var
        WHEN 2 THEN INSERT INTO lista VALUES (66666);
        WHEN 3 THEN INSERT INTO lista VALUES (45456);
        ELSE INSERT INTO lista VALUES (77777777);
    END CASE;
END;
```

****** [Documentación oficial de MySQL.](#)



Sentencia REPEAT

REPEAT

sentencias

UNTIL *condición*

END REPEAT

Las sentencias se repiten **hasta** que la condición sea cierta.

```
DELIMITER //
```

```
CREATE PROCEDURE pa_repeat (p1 INT)
```

```
BEGIN
```

```
    DECLARE cont DEFAULT 0;
```

```
    REPEAT
```

```
        SET cont = cont + 1;
```

```
    UNTIL cont = p1
```

```
    END REPEAT;
```

```
END//
```

******[Documentación oficial de MySQL](#)

Sentencia WHILE

WHILE *condicion* **DO**
sentencias

END WHILE

Las sentencias se repiten **mientras** la condición sea cierta.

Ejemplo:

```
DELIMITER //  
CREATE PROCEDURE pa_dowhile()  
BEGIN  
    DECLARE v1 INT DEFAULT 5;  
    WHILE v1 > 0 DO  
        SET v1 = v1 - 1;  
    END WHILE;  
END//
```

****** [Documentación oficial de MySQL](#)