# DML

# PROGRAMACIÓN

### Crear/Ejecutar Procedimiento

**Procedimiento sin parámetros:**

```
DELIMITER $$
CREATE PROCEDURE procSinParametros()
BEGIN
  SELECT COUNT(*) FROM clientes;
END
$$
DELIMITER ;

CALL procSinParametros();
```

**Procedimiento con parámetros de entrada:**

```
DELIMITER $$
CREATE PROCEDURE procConParametro(aumento
                                  int)
BEGIN
  UPDATE empleados
  SET salario = salario + aumento;
END
$$
DELIMITER ;

CALL procConParametro(100);
```

### Borrar Procedimiento / Función

```
DROP PROCEDURE [IF EXISTS] nombre;
DROP FUNCTION [IF EXISTS] nombre;
```

### Variables

```
DELIMITER //
CREATE PROCEDURE probandoVariables()
BEGIN
  DECLARE texto1 varchar(30);
  DECLARE numero1 integer;
  DECLARE numero2 integer DEFAULT 2;

  SET texto1 = 'pepe';
  SET numero1 = 3;
  SET numero2 = numero1 * numero2;

  SELECT count(*) INTO numero2
  FROM clientes;

  SELECT texto1, numero1, numero2;
END
//
DELIMITER ;

CALL probandoVariables();
```

### Estructura condicional IF

```
DELIMITER $$
CREATE PROCEDURE esPositivo (num int)
BEGIN
DECLARE TEXTO VARCHAR(50);
IF num>0 THEN
     SET TEXTO = 'Es positivo';
ELSEIF num<0 THEN
     SET TEXTO = 'Es negativo';
ELSE
     SET TEXTO = 'Es cero';
END IF;
SELECT TEXTO AS Mensaje;
END $$
DELIMITER ;
CALL esPositivo(0);
```

### Estructura condicional CASE 1

```
DELIMITER $$
CREATE PROCEDURE esPositivo (num int)
BEGIN
DECLARE TEXTO VARCHAR(50);
CASE
 WHEN num>0 THEN
      SET TEXTO = 'Positivo';
 WHEN num<0 THEN
      SET TEXTO = 'Negativo';
 ELSE
      SET TEXTO = 'Es cero';
END CASE;
SELECT TEXTO AS Mensaje;
END
$$
DELIMITER ;
CALL esPositivo(1);
```

### Estructura condicional CASE 2

```
DELIMITER $$
CREATE PROCEDURE quePuesto (puesto
varchar(1))
BEGIN
DECLARE TEXTO VARCHAR(50);
CASE puesto
 WHEN 'P' THEN
      SET TEXTO = 'Es profe';
 WHEN 'A' THEN
      SET TEXTO = 'Es alumno';
 ELSE
      SET TEXTO = 'Error';
END CASE;
SELECT TEXTO AS Mensaje;
END $$
DELIMITER ;
CALL quePuesto('A');
```

## Bucles WHILE

```
DELIMITER $$
create procedure iterar (limite int)
BEGIN
 DECLARE contador int DEFAULT 1;
 DECLARE msg VARCHAR(200) DEFAULT '';
 WHILE contador <= limite DO
  SET msg = CONCAT (msg,'Iteración
                n° ',contador,'\n');
  SET contador=contador+1;
 END WHILE;
 SELECT msg AS Mensaje;
END $$
DELIMITER ;
```

## Crear/Ejecutar Función

```
SET GLOBAL log_bin_trust_function_creators =1;

DELIMITER $$
CREATE FUNCTION representados(codigo
                              int)
RETURNS int
BEGIN
  RETURN SELECT COUNT(*) FROM clientes
         WHERE representante = codigo;
END $$
DELIMITER ;

// Uso en un SELECT ... FROM
SELECT nombre, representados
                        (codigoEmpleado)
FROM empleados;

//Uso en un SELECT
SELECT representados (1458);
```

## Cursores

```
-- Muestra información de todas las
-- oficinas fila a fila
CREATE PROCEDURE cursor_demo ()
BEGIN
 DECLARE fin BOOLEAN DEFAULT FALSE;
 DECLARE numOfi INT;
 DECLARE ciudadOfi VARCHAR(20);
 DECLARE cOfi CURSOR FOR
      SELECT num_oficina, ciudad
      FROM oficinas;
 DECLARE CONTINUE HANDLER FOR NOT FOUND
      SET fin = TRUE;
 OPEN cOfi;
 FETCH cOfi INTO numOfi, ciudadOfi;
 WHILE (NOT fin) DO
   SELECT numOfi, ciudadOfi;
   FETCH cOfi INTO numOfi, ciudadOfi;
 END WHILE;
 CLOSE cOfi;
END
```

## Control de errores

### Declaración de un manejador

```
CREATE PROCEDURE infoError(valor int)
BEGIN
 DECLARE codError CHAR(5);
 DECLARE numError int;
 DECLARE msgError TEXT;
 DECLARE EXIT HANDLER FOR SQLEXCEPTION
 BEGIN
   GET DIAGNOSTICS CONDITION 1
    codError = RETURNED_SQLSTATE,
    numError = MYSQL_ERRNO,
    msgError = MESSAGE_TEXT;
    SELECT numError, codError, msgError
 END;
 INSERT INTO test VALUES (valor);
 SELECT 'Inserción correcta';
END
```

| Tipos de acción de un manejador | |
|---|---|
| **CONTINUE** | Continua tras error |
| **EXIT** | Se detiene tras error |

| Tipos de error | |
|---|---|
| **Cód. Error Mysql** | Código numérico de 4 cifras. Solo MySQL. *Ej.: 1022* |
| **Cód. Error SQLSTATE** | Código texto de 5 caracteres. Compatible con otros sistemas SQL. *Ej.: '45000'* |
| **SQLWARNING** | Cualquier SQLState que empiece por '01'(warning). *Por defecto no detienen la ejecución.* |
| **NOT FOUND** | Cualquier SQLState que empiece por '02'(Cursores vacíos). *Por defecto no detienen la ejecución a no ser que sean lanzados con SIGNAL.* |
| **SQLEXCEPTION** | Cualquier SQLState que no empiece por '00', '01' o '02' (warning). Es decir, **cualquier error**. *Por defecto detienen la ejecución.* |

### Lanzamiento de excepciones

```
IF EXISTS (SELECT * FROM test
           WHERE campo=param) THEN
   SIGNAL SQLSTATE '45000'
      SET MYSQL_ERRNO = 1051,
          MESSAGE_TEXT = 'PK duplic.';
END IF;
```

## Transacciones

**Comenzar transacción**

START TRANSACTION;

*Los cambios no serán permanentes hasta COMMIT*

**Confirmar la transacción**

COMMIT;

*Los cambios se hacen permanentes*

**Revertir transacción**

ROLLBACK;

*Cualquier cambio de la transacción queda sin efecto*

## Disparadores

```
create trigger ej1_trigger BEFORE
                    UPDATE on producto
                    FOR EACH ROW
Begin
  -- Modificación del valor a insertar
  SET NEW.nombre=UPPER(NEW.nombre);

  -- Inserción del valor a insertar
  -- y del valor antiguo en LOG
  INSERT INTO log(msg)
      VALUES (CONCAT('Valor anterior:
                      ',OLD.nombre,
      ' Valor nuevo: ' , NEW.nombre));
end $$
DELIMITER ;
-- Disparo del trigger con un UPDATE
UPDATE producto SET nombre='Luis
Dorado'
WHERE nombre LIKE 'Luis';
```