

UNIDAD IV - UDF y CTE

¿Qué es una Función en MySQL?

- Bloque de código SQL que devuelve un valor.
- Recibe parámetros y encapsula lógica.
- Se puede reutilizar en consultas.

Tipos de Funciones

Escalares: Operan sobre una sola fila.

Funciones Matemáticas:

- o ABS(x), ROUND(x, d), SQRT(x)

Funciones de Cadena:

- o UPPER(str), LOWER(str), CONCAT(str1, str2, ...)

Funciones de Fecha:


- o NOW(), DATE_ADD(date, INTERVAL expr type),
YEAR(date)

Agregadas: Operan sobre un conjunto de filas.


- o SUM(), AVG(), COUNT()

Determinísticas / No Determinísticas

Funciones Determinísticas

- Devuelven el mismo resultado para los mismos parámetros.
-  Ej: `ABS()`, `LENGTH()`

No Determinísticas:

- Resultado puede variar (depende de variables externas).
-  Ej: `NOW()`

¿Por qué indicar DETERMINISTIC?

- Mejora el rendimiento (caché y optimización).
- Ayuda en replicación y mantenimiento del código.
- ¡Buena práctica declararlo siempre!

Sintaxis de Función

```
CREATE FUNCTION nombre_funcion(parametro1 tipo_dato, parametro2 tipo_dato, ...)  
RETURNS tipo_dato_retorno  
BEGIN  
    /* aquí escribimos el código de la función */  
    RETURN valor_de_retorno;  
END;
```

Ejemplo: Función para Calcular el Descuento

```
CREATE FUNCTION calcular_descuento(precio DECIMAL(7,2))  
DETERMINISTIC  
RETURNS DECIMAL(7,2)  
BEGIN  
    DECLARE descuento DECIMAL(7,2);  
    SET descuento = precio * 0.1;  
    RETURN precio - descuento;  
END;
```

Indica que la función devuelve el mismo resultado para los mismos parametros de entrada.

Cuidado!!!
(no es lo mismo)

¿Procedimientos o Funciones?

Característica	Función	Procedimiento
Retorno	Un único valor	No retorna valor directo
Uso	Dentro de consultas SQL	Mediante CALL
Operaciones	Cálculos simples	Operaciones complejas
Control de flujo	No	Sí (IF, WHILE...)

Ejemplo de Funciones y su invocación

```
CREATE FUNCTION calcular_precio_final(  
venta_id INT,  
descuento DECIMAL(7,2),  
impuesto DECIMAL(7,2))  
RETURNS DECIMAL(10,2)  
BEGIN  
    DECLARE subtotal DECIMAL(10,2);  
    DECLARE total DECIMAL(10,2);  
    SELECT precio * cantidad INTO subtotal  
    FROM ventas  
    WHERE venta_id = venta_id;  
    SET total = subtotal * (1 - descuento) * (1 +  
        impuesto);  
RETURN total;  
END;
```

```
CREATE PROCEDURE actualizar_precio_final(  
venta_id INT,  
descuento DECIMAL(7,2),  
impuesto DECIMAL(7,2))  
BEGIN  
    UPDATE ventas  
    SET precio = calcular_precio_final(venta_id,  
        descuento, impuesto)  
    WHERE venta_id = venta_id;  
END;
```

Introducción a CTE (Common Table Expressions)

 Definen tablas temporales para usar en una consulta.

Ventajas:

- ✓ Legibilidad
- ✓ Reutilización
- ✓ Recursividad

Tipos de CTE

1. Simple

```
WITH mi_CTE AS (...) SELECT * FROM mi_CTE;
```

2. Recursiva

```
WITH RECURSIVE nombre AS (...) SELECT * FROM nombre;
```

CTE vs. Subconsultas

Característica	Subconsulta	CTE
Legibilidad	Baja	Alta
Reutilización	No	Sí
Recursiva	No	Sí
Compatibilidad	General	MySQL 8+

Ejemplo de CTE

```
WITH ventas_totales AS (  
  SELECT cliente_id, SUM(total) AS total_ventas  
  FROM ventas  
  GROUP BY cliente_id  
)  
SELECT * FROM ventas_totales;
```