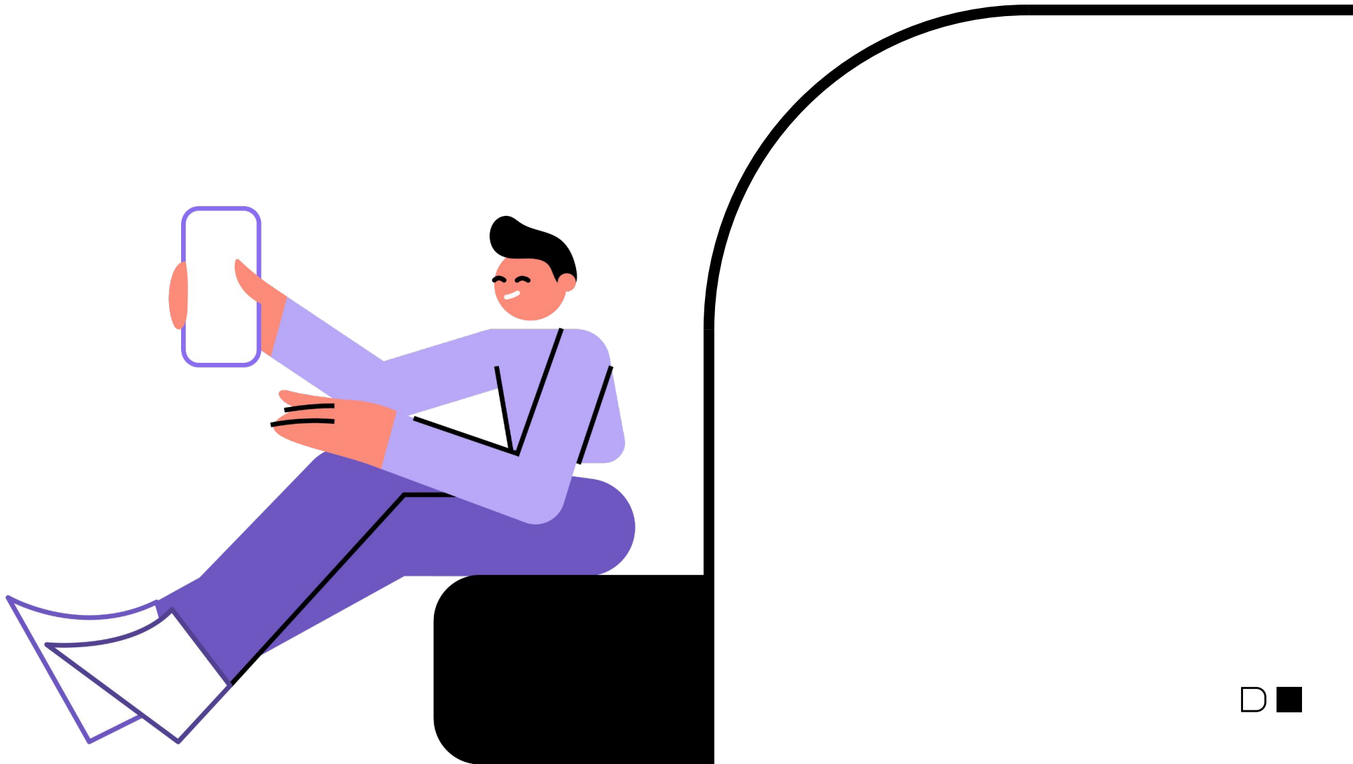
DIGITALHOUSE

Funciones almacenadas

Certified Tech Developer
The Ultimate Degree

Índice



01

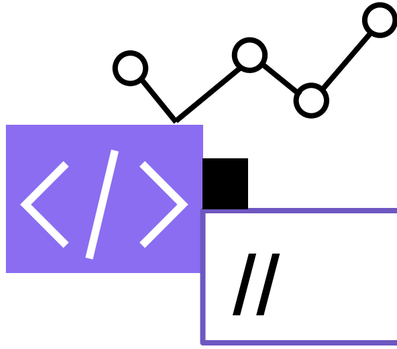
Concepto, estructura y definición

¿Qué es una función almacenada?

Una función almacenada en MySQL es una **rutina creada para tomar uno o más parámetros, realizarles algún procedimiento y retornar los resultados en un salida**.

- Pueden incluir parámetros solamente de entrada.
- Deben retornar un valor con algún tipo de dato definido.
- Solo retornan un valor individual, no un conjunto de registros. A esto se le llaman resultados escalares.

Por lo general, se los utiliza para realizar cálculos sobre los datos, obteniendo así lo que llamamos **datos derivados**. De esta forma, se reduce la necesidad de codificar dicha lógica en programas clientes.



Estructura de una función

- **CREATE FUNCTION:** se escribe este comando seguido del nombre que identificará a la función.
- **RETURNS:** se utiliza para indicar qué tipo de dato se retornará. La característica es para definir el tipo de función.
- **BEGIN:** esta cláusula se utiliza para indicar el inicio del código SQL.
- **RETURN:** utilizamos este comando para retornar el Bloque de instrucciones SQL.
- **END:** se utiliza para indicar el final del código SQL.

```SQL
CREATE FUNCTION nombre_function()
RETURNS [TIPO DE DATO] [CARACTERISTICA]
BEGIN
     RETURN -- Bloque de instrucciones SQL;
END
```

Estructura de una función - Características

Después de la definición del tipo de dato, tenemos que indicar las características de la función. Las opciones disponibles son las siguientes:

1. **DETERMINISTIC**: indica que la función siempre devuelve el mismo resultado cuando se utilizan los mismos parámetros de entrada.

2. **NOT DETERMINISTIC:** indica que la función no siempre devuelve el mismo resultado, aunque se utilicen los mismos parámetros de entrada.

3. **CONTAINS SQL:** indica que la función contiene sentencias SQL, pero no contiene sentencias de manipulación de datos.

4. **NO SQL:** indica que la función no contiene sentencias SQL.

5. **READS SQL DATA:** indica que contiene sentencias de lectura de datos, como la sentencia SELECT.

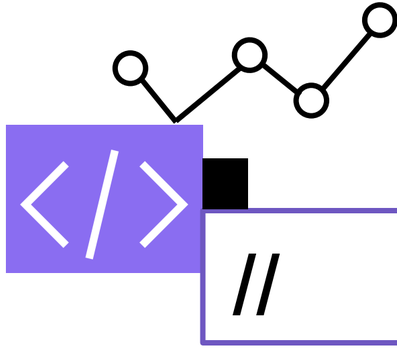
6. **MODIFIES SQL DATA:** indica que la función modifica los datos de la base y que contiene sentencias como INSERT, UPDATE o DELETE.

Estructura de una función - Aclaraciones

Si no queremos especificar una característica a la función, debemos ejecutar el siguiente comando para que no nos muestre mensaje de error a la hora de crearla:
SET GLOBAL log_bin_trust_function_creators = 1;

¡**Ojo!** A no confundir **RETURNS** con **RETURN**. La primera es para indicar el tipo de dato de retorno de la función y la segunda es para retornar el valor en el cuerpo de la función.

Si estamos trabajando sobre una query de trabajo, no olvidarse de usar el **DELIMITER** como se utilizan en los stored procedures.



Definición de una función

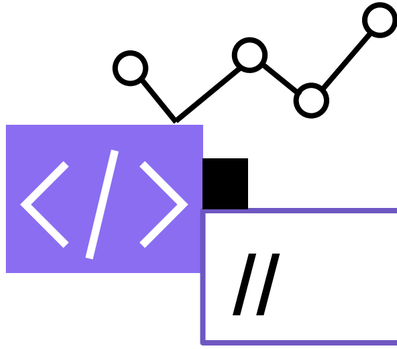
- **CREATE FUNCTION:** crea una función.

```SQL
CREATE FUNCTION nombre_function()
```

- **DROP FUNCTION:** elimina una función. Se requiere del privilegio de ALTER ROUTINE.

```SQL
DROP FUNCTION [IF EXISTS] nombre_function();
```

02 Variables



Declaración de variables

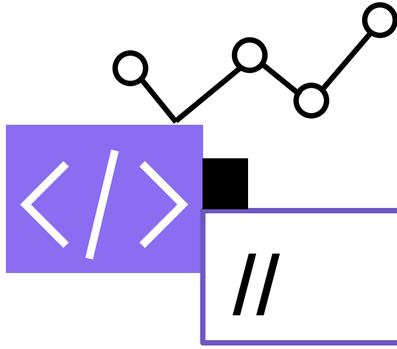
- Dentro de una **función** se permite declarar variables, es decir, elementos que almacenan datos que pueden ir cambiando a lo largo de la ejecución.
- La declaración de variables se coloca después de la cláusula BEGIN y antes del bloque de instrucciones SQL.
- Opcionalmente, se puede definir un valor inicial mediante la cláusula DEFAULT.

Sintaxis:

```sql
DECLARE nombre_variable TIPO_DE_DATO [DEFAULT valor];
```

Ejemplo:

```sql
DECLARE salario FLOAT DEFAULT 1000.00;
```



Asignación de valores a variables

- Para asignar un valor a una variable se utiliza la cláusula SET.
- Las variables solo pueden contener valores escalares. Es decir, un solo valor.

Sintaxis:

```sql
SET nombre_variable = expresión;
```

Ejemplo:

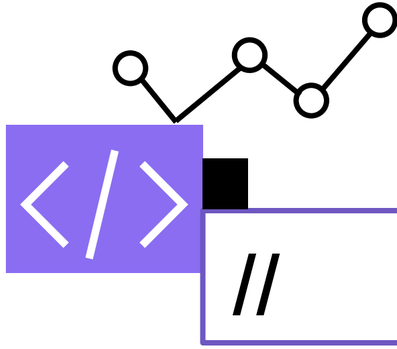
```sql
CREATE FUNCTION agregar_IVA(precio_sin_impuestos DOUBLE(10,12))
RETURNS DOUBLE(10,12) DETERMINISTIC
BEGIN
     DECLARE IVA INT DEFAULT 21;
     RETURN ((precio_sin_impuestos * IVA) / 100) + precio_sin_impuestos;
END
```

03 Parámetros

¿Qué es un parámetro?

- Los parámetros son variables por donde se envían y reciben datos de programas clientes.
- Se definen dentro de la cláusula CREATE.
- Las funciones pueden tener uno, varios o ningún parámetro de entrada.
- No pueden ingresarse parámetros del tipo OUT o INOUT.

Parámetro	Tipo	Función
IN	Entrada	Recibe datos



Declaración del parámetro IN

Es un parámetro de entrada de datos y se utiliza para recibir valores. Este parámetro viene definido por defecto cuando no se especifica su tipo.

Sintaxis:

```sql
CREATE FUNCTION nombre_function(IN param1 TIPO_DE_DATO, IN param2 TIPO_DE_DATO);
```

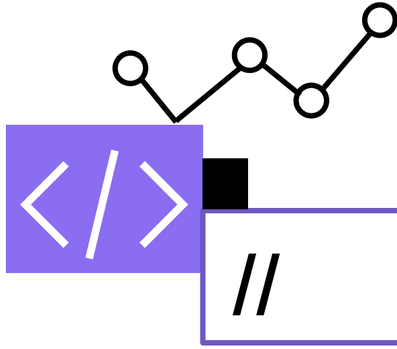
Ejemplo:

```sql
CREATE FUNCTION nombre_function(IN id_usuario INT)
BEGIN
    -- Bloque de instrucciones SQL;
END
```

Ejecución:

```sql
SELECT *, nombre_function(idUsuario) FROM usuarios;
```

04 Ejecución y ejemplos



Ejecución y ejemplos de funciones

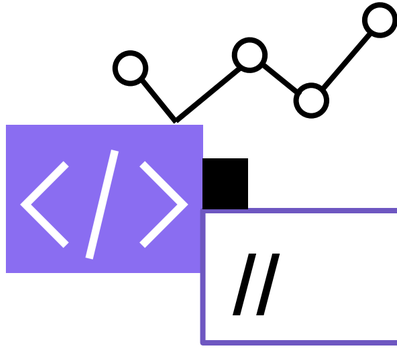
SQL

```sql
CREATE FUNCTION nombre_completo(nombre VARCHAR(45), apellido VARCHAR(45))
RETURNS VARCHAR(90) DETERMINISTIC
BEGIN
     RETURN CONCAT(nombre, ‘ ’, apellido);
END
```

SQL

```sql
SELECT idUsuario, nombre_completo(nombre, apellido) FROM usuarios;

SELECT legajo, nombre_completo(nombre, apellido) FROM empleados;
```



Ejecución y ejemplos de funciones

SQL
```
CREATE FUNCTION categoria_sueldo(sueldo DOUBLE)
RETURNS VARCHAR(15) DETERMINISTIC
BEGIN
   DECLARE categoria VARCHAR(15);
   CASE WHEN sueldo < 200 THEN SET categoria = ‘Bajo’;
   WHEN sueldo < 1000 THEN SET categoria = ‘Promedio’;
   ELSE SET categoria = ‘Alto’; END CASE;
        RETURN categoria;
END
```

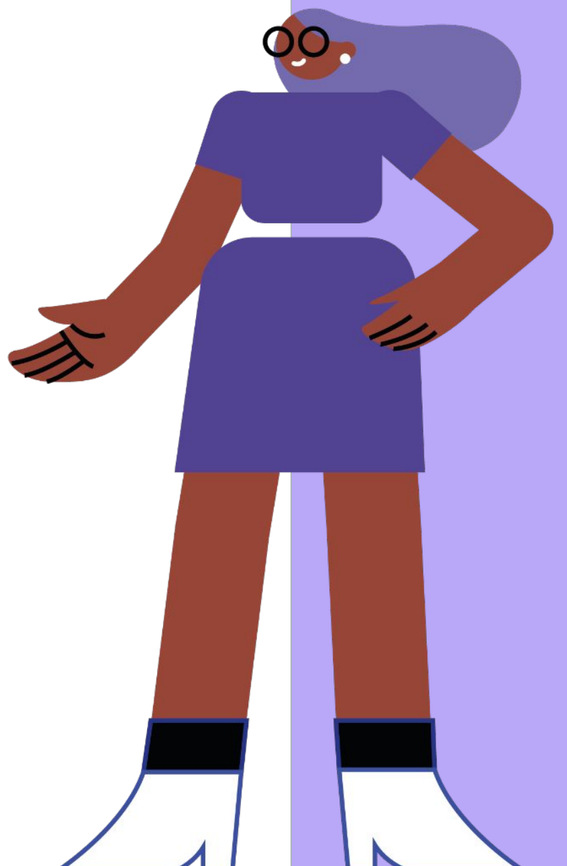
SQL
```
SELECT legajo, sueldo, categoria_sueldo(sueldo) FROM empleados;
```

05 Resumen

Resumen

Podemos decir que las funciones son nuestras **funciones de alteración personalizadas**. Mayormente son utilizadas para cuando se necesitan atributos derivados de los datos de nuestras tablas y que se utilicen en varias consultas de nuestro negocio.

Otra **ventaja** es que el uso de funciones puede ayudar a mejorar en gran medida el rendimiento y la performance general del sistema.



DIGITALHOUSE

¡Muchas gracias!