Funciones de usuario

Las funciones definidas por el usuario (User-Defined Function) son un bloque de código SQL que puede ser reutilizado y encapsula lógica compleja para realizar cálculos, transformar datos o devolver resultados específicos. Las funciones de usuario pueden devolver un único valor (funciones escalares) o un conjunto de resultados (funciones con tablas).
Durante la materia trabajaremos con funciones de usuario que devuelvan valores escalares.
El uso de funciones de usuario conlleva una serie de ventajas y desventajas que se detallan a continuación:

Ventajas

- Simplifica la lógica de una consulta extrayendo la complejidad de un cálculo dentro de la función de usuario.
- Da mejor legibilidad al código al extraer parte de la lógica de la consulta en una función de usuario específica o en varias funciones de usuario que resuelvan una tarea puntual.
- Al encapsular la lógica en una función de usuario es mucho más probable que podamos reutilizar dicho código en otros escenarios.
- Mejora el mantenimiento si muchas consultas hacen uso de una función de usuario para resolver una tarea puntual. Al modificar la función de usuario el cambio afecta a todas las consultas que la utilicen.

Desventajas

- El rendimiento de las consultas que llaman a funciones de usuarios puede verse afectado si manejan grandes volúmenes de datos.
- No se pueden utilizar para hacer modificaciones de datos. Es decir, no pueden ejecutar comandos de INSERT, UPDATE ni DELETE. Para ello utilizaremos otro tipo de herramientas.

Ejemplo

Veamos un ejemplo de una función de usuario que recibe dos fechas. Una de ellas representa la fecha de nacimiento de una persona y la otra la fecha de referencia para calcular la edad. La función devolverá un entero que devuelve la edad de la persona.

```
CREATE FUNCTION dbo.CalcularEdad(@fechaNacimiento DATE, @fechaActual DATE)
RETURNS INT
AS
BEGIN
	DECLARE @edad INT;

	-- Calcular la diferencia en años entre las dos fechas
	SET @edad = DATEDIFF(YEAR, @fechaNacimiento, @fechaActual);

	-- Restar un año si la persona aún no ha cumplido años en el año actual
	IF (MONTH(@fechaActual) < MONTH(@fechaNacimiento))
	OR (MONTH(@fechaActual) = MONTH(@fechaNacimiento) AND DAY(@fechaActual) <
DAY(@fechaNacimiento))
	BEGIN
	SET @edad = @edad - 1;
	END;

	RETURN @edad;
END;
```

Para poder utilizarla podemos hacer uso de una consulta de selección. Por ejemplo:

```
Set Dateformat 'YMD'
Select dbo.CalcularEdad('2000/10/10', '2024/10/15') as Edad
Select dbo.CalcularEdad('2000/10/10', '2024/10/09') as Edad
```

En la primera de las consultas, la edad que obtenemos será 24 ya que el mes y día de referencia son posteriores a la fecha de nacimiento de la persona. En cambio, en el segundo llamado esto no ocurre por lo que la edad de la persona será de 23.

También se puede hacer uso de la función obteniendo los datos de una tabla. Por ejemplo:

```
CREATE TABLE Personas(
	IDPersona int not null primary key,
	Nombre varchar(20) not null,
	Apellido varchar(20) not null,
	FechaNacimiento date not null
)
Go
INSERT INTO Personas (IDPersona, Nombre, Apellido, FechaNacimiento)
VALUES
(1, 'Juan', 'Gómez', '1990-05-12'),
(2, 'Ana', 'Martínez', '1985-08-23'),
(3, 'Luis', 'Fernández', '1992-11-30'),
```

```
(4, 'María', 'Pérez', '1988-03-15'),
(5, 'Carlos', 'López', '1995-07-09');
```

Uso de la función

```
Select Apellido, Nombre, FechaNacimiento,
       dbo.CalcularEdad(FechaNacimiento, getdate()) As Edad
From Personas
```

Aquí utiliza el valor almacenado en el campo FechaNacimiento como primer parámetro de la función y el valor devuelto por getdate como segundo parámetro. El llamado a la función se hace por cada una de las filas de la consulta por lo que si existen muchos registros en la tabla de Personas se ejecutará muchas veces el llamado a la función.