FACENA – UNNE

Licenciatura en Sistemas de Información

Bases de Datos I

Procedimientos y Funciones en SQL Server

Autor: Grupo 6

Año: 2025

1. INTRODUCCIÓN

En el desarrollo de bases de datos, los procedimientos almacenados y las funciones son herramientas esenciales para encapsular lógica, reutilizar código y garantizar la integridad de las operaciones. Su uso contribuye a mantener un sistema organizado, eficiente y fácil de mantener, evitando la repetición de instrucciones SQL y reduciendo errores.

Este informe presenta las funciones y procedimientos implementados en SQL Server con foco en ejemplos concretos y su utilidad práctica.

2. FUNCIONES DEFINIDAS POR EL USUARIO

Las funciones personalizadas permiten devolver valores calculados y encapsular operaciones repetitivas en las consultas. A continuación, se describen tres funciones clave.

2.1 fn_CalcularEdad

Calcula la edad de una persona según su fecha de nacimiento con un ajuste por mes y día actual:

```
CREATE FUNCTION fn_CalcularEdad (@fecha_nac DATE)
RETURNS INT
AS
BEGIN
    DECLARE @edad INT;
    SET @edad = DATEDIFF(YEAR, @fecha_nac, GETDATE())
              - CASE WHEN FORMAT(GETDATE(), 'MMdd') < FORMAT(@fecha_nac, 'MMdd')
THEN 1 ELSE 0 END;
    RETURN @edad;
END;
```

2.2 fn_TotalPagosSocio

Devuelve el total abonado por un socio sumando el campo monto en la tabla de pagos:

```
CREATE FUNCTION fn_TotalPagosSocio (@id_socio INT)
RETURNS DECIMAL(18,2)
AS
BEGIN
    DECLARE @total DECIMAL(18,2);
SELECT @total = ISNULL(SUM(monto),0)
    FROM pago     WHERE
id_socio = @id_socio;
    RETURN @total;
END;
```

2.3 fn_TieneSuscripcionActiva

Verifica si un socio posee una suscripción con estado ACTIVA y retorna 1 (verdadero) o 0 (falso):

```
CREATE FUNCTION fn_TieneSuscripcionActiva (@id_socio INT)
RETURNS BIT
AS
BEGIN
    DECLARE @r BIT = 0;
    IF EXISTS (
        SELECT 1
        FROM suscripcion s
        JOIN estado_suscripcion es ON es.id_estado_suscripcion =
s.id_estado
        WHERE s.id_socio = @id_socio AND es.descripcion = 'ACTIVA'
)
        SET @r = 1;
    RETURN @r;
END;
```

3. PROCEDIMIENTOS ALMACENADOS

Los procedimientos almacenados encapsulan operaciones de inserción, actualización y borrado, además de coordinar procesos complejos con control transaccional.

3.1 sp_socio_insertar

Inserta un nuevo socio y retorna su identificador generado automáticamente mediante SCOPE_IDENTITY():

```
CREATE PROCEDURE sp_socio_insertar
    @dni INT,
    @nombre_socio VARCHAR(80),
    @apellido_socio VARCHAR(80),
    @fecha_nac DATE,
    @id_contacto INT,
    @id_socio_out INT OUTPUT
AS
BEGIN
    SET NOCOUNT ON;     INSERT INTO socio (dni, nombre_socio, apellido_socio,
fecha_nacimiento, id_contacto)     VALUES (@dni, @nombre_socio, @apellido_socio,
@fecha_nac, @id_contacto);
    SET @id_socio_out = SCOPE_IDENTITY();
END;
```

3.2 sp_socio_actualizar y sp_socio_borrar

Permiten modificar o eliminar registros existentes de la tabla socio, centralizando la lógica y mejorando la seguridad.

3.3 sp_pago_insertar_tx (con transacción interna)

Ejemplo de procedimiento con manejo de transacciones anidadas o externas mediante SAVEPOINT y control de errores TRY...CATCH:

```
CREATE PROCEDURE sp_pago_insertar_tx
    @monto DECIMAL(18,2),
    @id_medio_pago INT,
    @id_suscripcion INT,
    @id_tipo_suscripcion INT,
    @id_socio INT,
    @id_estado_pago INT,
    @id_pago_out INT OUTPUT
AS
BEGIN
    SET NOCOUNT ON;
    DECLARE @interna BIT = CASE WHEN @@TRANCOUNT=0 THEN 1 ELSE 0 END;
    IF @interna=1 BEGIN TRAN ELSE SAVE TRAN SP_PAGO_CHILD;

    BEGIN TRY
        INSERT INTO pago
(monto,id_medio_pago,id_suscripcion,id_tipo_suscripcion,id_socio,id_estado_pago)
VALUES (@monto,@id_medio_pago,@id_suscripcion,@id_tipo_suscripcion,@id_so
cio,@id_estado_pago);         SET @id_pago_out=SCOPE_IDENTITY();

        IF @interna=1 COMMIT TRAN;
    END TRY
    BEGIN CATCH
```

```
        IF @interna=1 AND @@TRANCOUNT>0 ROLLBACK TRAN;
        ELSE IF @@TRANCOUNT>0 ROLLBACK TRAN SP_PAGO_CHILD;
        THROW;
    END CATCH;
END;
```

3.4 sp_registrar_socio_suscripcion_pago

Coordina la inserción de un socio, la creación de su suscripción y el registro del pago en una misma transacción explícita; asegura atomicidad y consistencia.

3.5 sp_demo_error_tx

Fuerza un error (1/0) para validar el correcto funcionamiento del bloque TRY...CATCH y el ROLLBACK.

4. CONCLUSIÓN

Como grupo comprendimos que los procedimientos y las funciones son pilares para mejorar la eficiencia, organización y confiabilidad de las bases de datos. Centralizar la lógica en el servidor SQL reduce errores, facilita el mantenimiento y garantiza operaciones coherentes y seguras.