PROCESO ALMACENADOS (STORY PROCED (SP))

* Es un bloque de código almacenado como un objeto
* Un proceso almacenado es más rapite que una clase
* Se puede ocultar la tabla, esto en aspecto de seguridad
* Ayuda en evitar un SQL inyection

Ejemplo de SQL Injection :

Slect \* from usuario

where usuario =’$t.var’ and pasword ‘$t.var2’

CREATE TABLE Cliente (

ClienteID int IDENTITY(1,1) PRIMARY KEY,

Nombre varchar(100) not NULL,

Telefono varchar(15) NULL,

Correo varchar(100) NULL

)

CREATE TABLE Producto (

ProductoID INt IDENTITY(1,1) PRIMARY KEY,

Nombre varchar (100) not null,

Precio decimal (10,2) not NULL,

Stock int not null

)

create table Orden (

OrdenID Int IDENTITY(1,1) PRIMARY key,

ClienteID Int not null,

Fecha DateTime Default GETDATE(),

Total Decimal (10,2) not null DEFAULT 0,

Foreign key (ClienteID) REFERENCES Cliente(ClienteID)

)

Create table DetalleOrden(

DetalleID int IDENTITY(1,1) PRIMARY KEY,

OrdenID int not null,

ProductoID int not null,

Cantidad int not null,

PrecioUnitario Decimal(10,2) not null,

SubTotal AS (Cantidad \* PrecioUnitario) PERSISTED,

FOREIGN key (OrdenID) REFERENCES Orden(OrdenID),

FOREIGN KEY (ProductoID) REFERENCES Producto(ProductoID)

)

CREATE PROCEDURE InsertarCliente

@Nombre VARCHAR(100),

@Telefono VARCHAR(15) = NULL,

@Correo VARCHAR(100) = NULL

AS

BEGIN

INSERT INTO Cliente (Nombre, Telefono, Correo)

VALUES (@Nombre, @Telefono, @Correo)

END

EXEC InsertarCliente @Nombre='Kevin', @Telefono='99887766', @Correo='ejemplo@noreply.com';

SELECT \* FROM Cliente;

# TAREA 1

CREATE PROCEDURE ObtenerCliente

@ClienteID INT = NULL

AS

BEGIN

IF @ClienteID IS NULL

SELECT \* FROM Cliente

ELSE

SELECT \* FROM Cliente WHERE ClienteID = @ClienteID

END

CREATE PROCEDURE BuscarCliente

@Nombre VARCHAR(100)

AS

BEGIN

SELECT \* FROM Cliente

WHERE Nombre LIKE '%' + @Nombre + '%'

END

CREATE PROCEDURE EliminarCliente

@ClienteID INT

AS

BEGIN

DELETE FROM Cliente WHERE ClienteID = @ClienteID

END

CREATE PROCEDURE ObtenerProducto

@ProductoID INT = NULL

AS

BEGIN

IF @ProductoID IS NULL

SELECT \* FROM Producto

ELSE

SELECT \* FROM Producto WHERE ProductoID = @ProductoID

END

CREATE PROCEDURE BuscarProducto

@Nombre VARCHAR(100)

AS

BEGIN

SELECT \* FROM Producto

WHERE Nombre LIKE '%' + @Nombre + '%'

END

CREATE PROCEDURE EliminarProducto

@ProductoID INT

AS

BEGIN

DELETE FROM Producto WHERE ProductoID = @ProductoID

END

CREATE PROCEDURE ObtenerOrden

@OrdenID INT = NULL

AS

BEGIN

IF @OrdenID IS NULL

SELECT \* FROM Orden

ELSE

SELECT \* FROM Orden WHERE OrdenID = @OrdenID

END

CREATE PROCEDURE EliminarOrden

@OrdenID INT

AS

BEGIN

DELETE FROM DetalleOrden WHERE OrdenID = @OrdenID

DELETE FROM Orden WHERE OrdenID = @OrdenID

END

CREATE PROCEDURE ObtenerDetalleOrden

@DetalleID INT = NULL

AS

BEGIN

IF @DetalleID IS NULL

SELECT \* FROM DetalleOrden

ELSE

SELECT \* FROM DetalleOrden WHERE DetalleID = @DetalleID

END

CREATE PROCEDURE EliminarDetalleOrden

@DetalleID INT

AS

BEGIN

DELETE FROM DetalleOrden WHERE DetalleID = @DetalleID

END

CREATE PROCEDURE ActualizarTotalOrden

@OrdenID INT

AS

BEGIN

UPDATE Orden

SET Total = (

SELECT ISNULL(SUM(SubTotal), 0)

FROM DetalleOrden

WHERE OrdenID = @OrdenID

)

WHERE OrdenID = @OrdenID

END

## Ejecutables

EXEC ObtenerCliente -- Obtiene todos

EXEC ObtenerCliente @ClienteID = 1 -- Obtiene uno específico

EXEC BuscarCliente @Nombre = 'Kev' -- Buscará cualquier nombre que contenga 'Kev'

EXEC BuscarProducto @Nombre = 'Lap' -- Buscará cualquier producto que contenga 'Lap'

EXEC EliminarCliente @ClienteID = 1

EXEC EliminarProducto @ProductoID = 1

EXEC EliminarOrden @OrdenID = 1

EXEC EliminarDetalleOrden @DetalleID = 1

EXEC ActualizarTotalOrden @OrdenID = 1

## Insertar datos

-- Procedimiento para insertar un producto

CREATE PROCEDURE InsertarProducto

@Nombre VARCHAR(100),

@Precio DECIMAL(10,2),

@Stock INT

AS

BEGIN

INSERT INTO Producto (Nombre, Precio, Stock)

VALUES (@Nombre, @Precio, @Stock)

END

-- Procedimiento para insertar una orden

CREATE PROCEDURE InsertarOrden

@ClienteID INT

AS

BEGIN

INSERT INTO Orden (ClienteID, Fecha, Total)

VALUES (@ClienteID, GETDATE(), 0)

SELECT SCOPE\_IDENTITY() AS OrdenID

END

-- Procedimiento para insertar detalle de orden

CREATE PROCEDURE InsertarDetalleOrden

@OrdenID INT,

@ProductoID INT,

@Cantidad INT,

@PrecioUnitario DECIMAL(10,2)

AS

BEGIN

INSERT INTO DetalleOrden (OrdenID, ProductoID, Cantidad, PrecioUnitario)

VALUES (@OrdenID, @ProductoID, @Cantidad, @PrecioUnitario)

EXEC ActualizarTotalOrden @OrdenID

END

-- Insertar 10 Clientes

EXEC InsertarCliente @Nombre='Juan Pérez', @Telefono='99887761', @Correo='juan@email.com';

EXEC InsertarCliente @Nombre='María García', @Telefono='99887762', @Correo='maria@email.com';

EXEC InsertarCliente @Nombre='Pedro López', @Telefono='99887763', @Correo='pedro@email.com';

EXEC InsertarCliente @Nombre='Ana Martínez', @Telefono='99887764', @Correo='ana@email.com';

EXEC InsertarCliente @Nombre='Luis Torres', @Telefono='99887765', @Correo='luis@email.com';

EXEC InsertarCliente @Nombre='Carmen Ruiz', @Telefono='99887766', @Correo='carmen@email.com';

EXEC InsertarCliente @Nombre='José Sánchez', @Telefono='99887767', @Correo='jose@email.com';

EXEC InsertarCliente @Nombre='Sofia Castro', @Telefono='99887768', @Correo='sofia@email.com';

EXEC InsertarCliente @Nombre='Diego Mora', @Telefono='99887769', @Correo='diego@email.com';

EXEC InsertarCliente @Nombre='Laura Flores', @Telefono='99887770', @Correo='laura@email.com';

-- Insertar 10 Productos

EXEC InsertarProducto 'Laptop HP', 1200.00, 50;

EXEC InsertarProducto 'Monitor Dell', 300.00, 75;

EXEC InsertarProducto 'Teclado Mecánico', 80.00, 100;

EXEC InsertarProducto 'Mouse Gaming', 45.00, 150;

EXEC InsertarProducto 'Impresora Canon', 250.00, 30;

EXEC InsertarProducto 'Disco Duro 1TB', 70.00, 80;

EXEC InsertarProducto 'Memoria RAM 8GB', 60.00, 120;

EXEC InsertarProducto 'Webcam HD', 40.00, 90;

EXEC InsertarProducto 'Router WiFi', 55.00, 60;

EXEC InsertarProducto 'Auriculares BT', 85.00, 70;

-- Insertar 10 Órdenes con sus Detalles

DECLARE @OrdenID INT;

EXEC InsertarOrden 1;

SET @OrdenID = SCOPE\_IDENTITY();

EXEC InsertarDetalleOrden @OrdenID, 1, 2, 1200.00;

EXEC InsertarDetalleOrden @OrdenID, 3, 1, 80.00;

EXEC InsertarOrden 2;

SET @OrdenID = SCOPE\_IDENTITY();

EXEC InsertarDetalleOrden @OrdenID, 2, 3, 300.00;

EXEC InsertarOrden 3;

SET @OrdenID = SCOPE\_IDENTITY();

EXEC InsertarDetalleOrden @OrdenID, 4, 2, 45.00;

EXEC InsertarDetalleOrden @OrdenID, 6, 1, 70.00;

EXEC InsertarOrden 4;

SET @OrdenID = SCOPE\_IDENTITY();

EXEC InsertarDetalleOrden @OrdenID, 5, 1, 250.00;

EXEC InsertarOrden 5;

SET @OrdenID = SCOPE\_IDENTITY();

EXEC InsertarDetalleOrden @OrdenID, 7, 4, 60.00;

EXEC InsertarOrden 6;

SET @OrdenID = SCOPE\_IDENTITY();

EXEC InsertarDetalleOrden @OrdenID, 8, 2, 40.00;

EXEC InsertarDetalleOrden @OrdenID, 9, 1, 55.00;

EXEC InsertarOrden 7;

SET @OrdenID = SCOPE\_IDENTITY();

EXEC InsertarDetalleOrden @OrdenID, 10, 3, 85.00;

EXEC InsertarOrden 8;

SET @OrdenID = SCOPE\_IDENTITY();

EXEC InsertarDetalleOrden @OrdenID, 1, 1, 1200.00;

EXEC InsertarOrden 9;

SET @OrdenID = SCOPE\_IDENTITY();

EXEC InsertarDetalleOrden @OrdenID, 3, 2, 80.00;

EXEC InsertarDetalleOrden @OrdenID, 4, 3, 45.00;

EXEC InsertarOrden 10;

SET @OrdenID = SCOPE\_IDENTITY();

EXEC InsertarDetalleOrden @OrdenID, 2, 2, 300.00;

EXEC InsertarDetalleOrden @OrdenID, 7, 3, 60.00;

# Tarea 2

-- Ejercicio 1: Insertar un nuevo registro de cliente

CREATE PROCEDURE InsertarCliente

@Nombre VARCHAR(100),

@Telefono VARCHAR(15) = NULL,

@Correo VARCHAR(100) = NULL

AS

BEGIN

INSERT INTO Cliente (Nombre, Telefono, Correo)

VALUES (@Nombre, @Telefono, @Correo);

END

-- Ejercicio 2: Actualizar el precio de un producto

CREATE PROCEDURE ActualizarPrecioProducto

@ProductoID INT,

@NuevoPrecio DECIMAL(10,2)

AS

BEGIN

UPDATE Producto

SET Precio = @NuevoPrecio

WHERE ProductoID = @ProductoID;

END

-- Ejercicio 3: Eliminar un registro de orden y sus detalles

CREATE PROCEDURE EliminarOrden

@OrdenID INT

AS

BEGIN

DELETE FROM DetalleOrden WHERE OrdenID = @OrdenID;

DELETE FROM Orden WHERE OrdenID = @OrdenID;

END

-- Ejercicio 4: Obtener los 10 productos más vendidos

CREATE PROCEDURE ObtenerProductosMasVendidos

AS

BEGIN

SELECT TOP 10 p.ProductoID, p.Nombre, SUM(d.Cantidad) AS TotalVendido

FROM DetalleOrden d

JOIN Producto p ON d.ProductoID = p.ProductoID

GROUP BY p.ProductoID, p.Nombre

ORDER BY TotalVendido DESC;

END

-- Ejercicio 5: Calcular el total de ventas por cliente

CREATE PROCEDURE CalcularTotalVentasPorCliente

AS

BEGIN

SELECT o.ClienteID, c.Nombre, SUM(o.Total) AS TotalVentas

FROM Orden o

JOIN Cliente c ON o.ClienteID = c.ClienteID

GROUP BY o.ClienteID, c.Nombre

ORDER BY TotalVentas DESC;

END

-- Ejercicio 6: Insertar un nuevo producto y su stock inicial

CREATE PROCEDURE InsertarProducto

@Nombre VARCHAR(100),

@Precio DECIMAL(10,2),

@Stock INT

AS

BEGIN

INSERT INTO Producto (Nombre, Precio, Stock)

VALUES (@Nombre, @Precio, @Stock);

END

-- Ejercicio 7: Actualizar el estado de una orden (se necesita agregar un campo de estado en la tabla Orden)

ALTER TABLE Orden ADD Estado VARCHAR(50) DEFAULT 'Pendiente';

CREATE PROCEDURE ActualizarEstadoOrden

@OrdenID INT,

@NuevoEstado VARCHAR(50)

AS

BEGIN

UPDATE Orden

SET Estado = @NuevoEstado

WHERE OrdenID = @OrdenID;

END

-- Ejercicio 8: Obtener el promedio de ventas por año

CREATE PROCEDURE ObtenerPromedioVentasPorAño

AS

BEGIN

SELECT YEAR(Fecha) AS Año, AVG(Total) AS PromedioVentas

FROM Orden

GROUP BY YEAR(Fecha)

ORDER BY Año DESC;

END

-- Ejercicio 9: Obtener los 5 clientes con más órdenes pendientes

CREATE PROCEDURE ObtenerClientesConMasOrdenesPendientes

AS

BEGIN

SELECT TOP 5 c.ClienteID, c.Nombre, COUNT(o.OrdenID) AS OrdenesPendientes

FROM Orden o

JOIN Cliente c ON o.ClienteID = c.ClienteID

WHERE o.Estado <> 'Shipped'

GROUP BY c.ClienteID, c.Nombre

ORDER BY OrdenesPendientes DESC;

END

-- Ejercicio 10: Eliminar un cliente y sus órdenes

CREATE PROCEDURE EliminarCliente

@ClienteID INT

AS

BEGIN

DELETE FROM DetalleOrden WHERE OrdenID IN (SELECT OrdenID FROM Orden WHERE ClienteID = @ClienteID);

DELETE FROM Orden WHERE ClienteID = @ClienteID;

DELETE FROM Cliente WHERE ClienteID = @ClienteID;

END

# TRIGERS

Es un bloque de código que se ejecuta automáticamente cuando ocurre un evento en una tabla.

Create table Log (

ID int identitty (1,1) Primary key

Table varchar (50)

tipoAcción varchar (50)

ID\_registro int,

Usuario varchar (100),

fechaHora Datetime Default GetDate()

)

Tarea

Ejecutar un trigger que actualize el stock del producto desde el detalle

Investigar el tigger instead para hacer validacionesy crear un trigger que no permita consumir el producto si la cantidad consumida es mayor a la existente en el stock

-- Trigger para verificar stock antes de insertar y actualizar stock automáticamente

CREATE TRIGGER trg\_ActualizarStock

ON DetalleOrden

INSTEAD OF INSERT

AS

BEGIN

SET NOCOUNT ON;

-- Verificar si hay suficiente stock para los productos

IF EXISTS (

SELECT 1

FROM inserted i

JOIN Producto p ON i.ProductoID = p.ProductoID

WHERE i.Cantidad > p.Stock

)

BEGIN

DECLARE @ProductoSinStock VARCHAR(100), @StockDisponible INT, @CantidadSolicitada INT;

SELECT TOP 1

@ProductoSinStock = p.Nombre,

@StockDisponible = p.Stock,

@CantidadSolicitada = i.Cantidad

FROM inserted i

JOIN Producto p ON i.ProductoID = p.ProductoID

WHERE i.Cantidad > p.Stock;

DECLARE @ErrorMsg NVARCHAR(255) = 'No hay suficiente stock del producto "' + @ProductoSinStock +

'". Stock disponible: ' + CAST(@StockDisponible AS VARCHAR) +

', Cantidad solicitada: ' + CAST(@CantidadSolicitada AS VARCHAR);

RAISERROR(@ErrorMsg, 16, 1);

RETURN;

END

-- Si hay suficiente stock, insertar el registro en DetalleOrden

-- Usamos SET IDENTITY\_INSERT porque estamos en un trigger INSTEAD OF

BEGIN TRY

-- Insertar los registros en DetalleOrden

INSERT INTO DetalleOrden (OrdenID, ProductoID, Cantidad, PrecioUnitario)

SELECT OrdenID, ProductoID, Cantidad, PrecioUnitario

FROM inserted;

-- Actualizar el stock del producto restando la cantidad vendida

UPDATE p

SET p.Stock = p.Stock - i.Cantidad

FROM Producto p

JOIN inserted i ON p.ProductoID = i.ProductoID;

-- Actualizar el total de la orden

DECLARE @OrdenID INT;

SELECT @OrdenID = OrdenID FROM inserted;

-- Utilizar el procedimiento almacenado existente

EXEC ActualizarTotalOrden @OrdenID;

END TRY

BEGIN CATCH

DECLARE @ErrorMessage NVARCHAR(4000) = ERROR\_MESSAGE();

RAISERROR(@ErrorMessage, 16, 1);

END CATCH

END;

GO