Practica unidad 5. Seguridad

Parte 1.

Configuración de Dos Instancias de SQL Server en la Misma Máquina

Antes de realizar la práctica de administración de bases de datos, necesitamos configurar dos instancias de SQL Server en la misma máquina. Pasos a seguir:

Requisitos previos:

- Windows 10/11 o Windows Server 2016/2019/2022
- SQL Server (Developer o Enterprise Edition)
- Suficiente espacio en disco (mínimo 10GB recomendado)
- Al menos 8GB de RAM (recomendado para dos instancias)

Paso 1: Instalar la primera instancia (Default o con nombre)

1. Ejecuta el instalador de SQL Server
2. Selecciona "Nueva instalación independiente de SQL Server o agregar características a una instalación existente"
3. En la página "Configuración de instancia":
    - ✓ Para la primera instancia puedes seleccionar "Instancia predeterminada" (MSSQLSERVER)
    - ✓ O especificar un nombre de instancia (ej: SQLPRIMARY)
4. Completa el resto de la instalación con la configuración deseada

Paso 2: Instalar la segunda instancia (con nombre diferente)

1. Ejecuta nuevamente el instalador de SQL Server
2. Selecciona la misma opción de nueva instalación
3. En "Configuración de instancia":
    - ✓ Selecciona "Instancia con nombre"
    - ✓ Proporciona un nombre diferente (ej: SQLMIRROR)
4. Asegúrate de especificar diferentes directorios para los archivos de esta instancia
5. Completa la instalación

Paso 3: Configurar los servicios

1. Abre "SQL Server Configuration Manager"

2. Verifica que ambos servicios estén ejecutándose:
    - SQL Server (SQLPRIMARY)
    - SQL Server (SQLMIRROR)
3. Configura el inicio automático para ambos servicios

Paso 4: Configurar los puertos (opcional pero recomendado)

1. En SQL Server Configuration Manager, expande "Configuración de red de SQL Server"
2. Para cada instancia:
    - Selecciona "Protocolos para [nombre de instancia]"
    - Habilita TCP/IP
    - En propiedades de TCP/IP, en la pestaña "Direcciones IP", configura un puerto estático diferente para cada instancia (ej: 1433 para la primera, 1434 para la segunda)

Paso 5: Verificar la conectividad

1. Abre SQL Server Management Studio (SSMS)
2. Conéctate a cada instancia por separado:
    - Para la primera instancia: localhost o localhost\SQLPRIMARY
    - Para la segunda instancia: localhost\SQLMIRROR

Paso 6: Configurar alias (opcional pero útil)

1. En SQL Server Configuration Manager, selecciona "Alias de SQL Server"
2. Crea alias para cada instancia:
    - Alias: PRIMARY_SERVER, Servidor: localhost\SQLPRIMARY
    - Alias: MIRROR_SERVER, Servidor: localhost\SQLMIRROR

Comandos útiles para verificar las instancias:

-- Ver todas las instancias instaladas en el equipo

SELECT @@SERVERNAME AS 'Server Name';

-- O desde PowerShell:

Get-Service -Name "MSSQL" | Select-Object DisplayName, Status, StartType*

Notas importantes:

- Cada instancia tendrá sus propios servicios SQL Server y SQL Agent
- Cada instancia consume recursos independientes (CPU, memoria)
- Las instancias no comparten bases de datos del sistema (master, model, msdb, tempdb)
- Puedes detener/arrancar cada instancia independientemente

Una vez configuradas las dos instancias, puedes proceder con la práctica de administración de bases de datos (parte 2) utilizando:

- SQLPRIMARY como tu servidor principal
- SQLMIRROR como tu servidor espejo/réplica

Parte 2.

Práctica de Administración de Bases de Datos: Seguridad en SQL Server

Ejercicio A.1: Crear tablas con datos sensibles para pruebas de seguridad

```
-- En la base de datos principal (PrincipalServer)
-- crear y abrir la base DB_SeguridadPractica
USE DB_SeguridadPractica;
GO

-- Tabla de usuarios con información sensible
CREATE TABLE dbo.Usuarios (
    UsuarioID INT IDENTITY(1,1) PRIMARY KEY,
    NombreCompleto NVARCHAR(100) NOT NULL,
    NombreUsuario NVARCHAR(50) NOT NULL UNIQUE,
    Contraseña NVARCHAR(255) NOT NULL,  -- Se almacenará encriptada
    Email NVARCHAR(100) NOT NULL,
    FechaNacimiento DATE,
    Salario DECIMAL(10,2),
    Direccion NVARCHAR(200),
    Telefono NVARCHAR(20),
```

```
    EsAdministrador BIT DEFAULT 0,
    FechaCreacion DATETIME DEFAULT GETDATE(),
    UltimoLogin DATETIME
);
GO

-- Tabla de registros de auditoría
CREATE TABLE dbo.AuditoriaLog (
    LogID INT IDENTITY(1,1) PRIMARY KEY,
    UsuarioID INT NULL,
    Accion NVARCHAR(50) NOT NULL,
    TablaAfectada NVARCHAR(100) NOT NULL,
    RegistroID INT NULL,
    DetallesAnteriores NVARCHAR(MAX),
    DetallesNuevos NVARCHAR(MAX),
    FechaHora DATETIME DEFAULT GETDATE(),
    DireccionIP NVARCHAR(50),
    FOREIGN KEY (UsuarioID) REFERENCES dbo.Usuarios(UsuarioID)
);
GO

-- Tabla de información financiera
CREATE TABLE dbo.TransaccionesFinancieras (
    TransaccionID INT IDENTITY(1,1) PRIMARY KEY,
    UsuarioID INT NOT NULL,
    TipoTransaccion NVARCHAR(50) NOT NULL,
    Monto DECIMAL(12,2) NOT NULL,
    FechaTransaccion DATETIME DEFAULT GETDATE(),
    Descripcion NVARCHAR(255),
```

```
    CuentaDestino NVARCHAR(50),
    Aprobada BIT DEFAULT 0,
    FOREIGN KEY (UsuarioID) REFERENCES dbo.Usuarios(UsuarioID)
);
GO

-- Tabla de productos
CREATE TABLE dbo.Productos (
    ProductoID INT IDENTITY(1,1) PRIMARY KEY,
    Nombre NVARCHAR(100) NOT NULL,
    Descripcion NVARCHAR(500),
    Precio DECIMAL(10,2) NOT NULL,
    Costo DECIMAL(10,2) NOT NULL,
    Stock INT NOT NULL DEFAULT 0,
    Categoria NVARCHAR(50),
    FechaCreacion DATETIME DEFAULT GETDATE(),
    Activo BIT DEFAULT 1
);
GO
```

Ejercicio A.2: Insertar datos de prueba

```
-- Insertar usuarios (las contraseñas son el hash de 'Password123')
INSERT INTO dbo.Usuarios (NombreCompleto, NombreUsuario, Contraseña, Email,
FechaNacimiento, Salario, Direccion, Telefono, EsAdministrador)
VALUES
('Administrador del Sistema', 'admin',
'0x0200A89A828C3F0D8C1A3E2A3D8C1A3E2A3D8C1A3E2A3D8C1A3E2A3D8C1A3E2A3D8C1A3E',
'admin@empresa.com', '1980-01-15', 7500.00, 'Av. Principal 123', '+1234567890', 1),
```

('Juan Pérez', 'jperez',
'0x0200A89A828C3F0D8C1A3E2A3D8C1A3E2A3D8C1A3E2A3D8C1A3E2A3D8C1A3E', 'juan@empresa.com', '1990-05-22', 3500.00, 'Calle Secundaria 456', '+1234567891', 0),

('María Gómez', 'mgomez',
'0x0200A89A828C3F0D8C1A3E2A3D8C1A3E2A3D8C1A3E2A3D8C1A3E2A3D8C1A3E', 'maria@empresa.com', '1985-11-30', 4200.00, 'Av. Central 789', '+1234567892', 0),

('Carlos Ruiz', 'cruiz',
'0x0200A89A828C3F0D8C1A3E2A3D8C1A3E2A3D8C1A3E2A3D8C1A3E2A3D8C1A3E', 'carlos@empresa.com', '1992-03-10', 3800.00, 'Calle Norte 101', '+1234567893', 0);

GO

-- Insertar productos

INSERT INTO dbo.Productos (Nombre, Descripcion, Precio, Costo, Stock, Categoria)

VALUES

('Laptop Elite', 'Laptop de última generación con 16GB RAM y 1TB SSD', 1299.99, 850.00, 25, 'Tecnología'),

('Smartphone Pro', 'Teléfono inteligente con cámara de 48MP', 799.99, 450.00, 50, 'Tecnología'),

('Tablet Basic', 'Tablet con pantalla HD de 10 pulgadas', 299.99, 180.00, 30, 'Tecnología'),

('Auriculares Inalámbricos', 'Auriculares con cancelación de ruido', 199.99, 120.00, 40, 'Accesorios'),

('Monitor 27"', 'Monitor QHD de 27 pulgadas', 349.99, 220.00, 15, 'Tecnología');

GO

-- Insertar transacciones financieras

INSERT INTO dbo.TransaccionesFinancieras (UsuarioID, TipoTransaccion, Monto, Descripcion, CuentaDestino, Aprobada)

VALUES

(2, 'Salario', 3500.00, 'Pago de nómina', 'CUENTA-12345', 1),

(3, 'Salario', 4200.00, 'Pago de nómina', 'CUENTA-12346', 1),

(4, 'Salario', 3800.00, 'Pago de nómina', 'CUENTA-12347', 1),

(2, 'Reembolso', 150.00, 'Reembolso de gastos', 'CUENTA-12345', 1),

(3, 'Bonificación', 300.00, 'Bonificación por desempeño', 'CUENTA-12346', 1),

```
(2, 'Compra', -799.99, 'Compra de Smartphone Pro', 'CUENTA-98765', 1),
(3, 'Compra', -299.99, 'Compra de Tablet Basic', 'CUENTA-98766', 1);
GO

-- Insertar registros de auditoría de ejemplo
INSERT INTO dbo.AuditoriaLog (UsuarioID, Accion, TablaAfectada, RegistroID, DetallesAnteriores, DetallesNuevos, DireccionIP)
VALUES
(1, 'INSERT', 'Usuarios', 2, NULL, '{"NombreCompleto":"Juan Pérez","NombreUsuario":"jperez","Email":"juan@empresa.com"}', '192.168.1.100'),
(1, 'INSERT', 'Productos', 1, NULL, '{"Nombre":"Laptop Elite","Precio":1299.99}', '192.168.1.100'),
(2, 'UPDATE', 'Usuarios', 2, '{"Salario":3400.00}', '{"Salario":3500.00}', '192.168.1.101'),
(1, 'DELETE', 'Productos', 5, '{"Nombre":"Monitor 27\"","Precio":349.99}', NULL, '192.168.1.100');
GO
```

Ejercicio A.3: Crear procedimientos almacenados para operaciones comunes

```
-- Procedimiento para login de usuarios
CREATE PROCEDURE dbo.sp_LoginUsuario
    @NombreUsuario NVARCHAR(50),
    @Contraseña NVARCHAR(255)
AS
BEGIN
    SET NOCOUNT ON;

    DECLARE @UsuarioID INT;

    -- Verificar credenciales (en un caso real, compararías hashes)
    SELECT @UsuarioID = UsuarioID
    FROM dbo.Usuarios
```

```
    WHERE NombreUsuario = @NombreUsuario AND Contraseña = @Contraseña;

    -- Registrar intento de login
    INSERT INTO dbo.AuditoriaLog (UsuarioID, Accion, TablaAfectada, RegistroID, DetallesNuevos)
    VALUES (@UsuarioID, 'LOGIN', 'Usuarios', @UsuarioID,
        '{"IntentoLogin":"' + CASE WHEN @UsuarioID IS NULL THEN 'Fallido' ELSE 'Exitoso' END +
'"}');

    -- Actualizar último login si fue exitoso
    IF @UsuarioID IS NOT NULL
    BEGIN
        UPDATE dbo.Usuarios
        SET UltimoLogin = GETDATE()
        WHERE UsuarioID = @UsuarioID;

        SELECT 'Login exitoso' AS Resultado, u.*
        FROM dbo.Usuarios u
        WHERE u.UsuarioID = @UsuarioID;
    END
    ELSE
    BEGIN
        SELECT 'Credenciales inválidas' AS Resultado;
    END
END;
GO

-- Procedimiento para registrar transacciones financieras
CREATE PROCEDURE dbo.sp_RegistrarTransaccion
    @UsuarioID INT,
```

```
    @TipoTransaccion NVARCHAR(50),
    @Monto DECIMAL(12,2),
    @Descripcion NVARCHAR(255),
    @CuentaDestino NVARCHAR(50)
AS
BEGIN
    SET NOCOUNT ON;

    BEGIN TRY
        BEGIN TRANSACTION;

        -- Insertar la transacción
        INSERT INTO dbo.TransaccionesFinancieras (UsuarioID, TipoTransaccion, Monto, Descripcion, CuentaDestino)
        VALUES (@UsuarioID, @TipoTransaccion, @Monto, @Descripcion, @CuentaDestino);

        -- Registrar en auditoría
        DECLARE @TransaccionID INT = SCOPE_IDENTITY();

        INSERT INTO dbo.AuditoriaLog (UsuarioID, Accion, TablaAfectada, RegistroID, DetallesNuevos)
        VALUES (@UsuarioID, 'INSERT', 'TransaccionesFinancieras', @TransaccionID,
            '{"Tipo":"' + @TipoTransaccion + '","Monto":' + CAST(@Monto AS NVARCHAR(20)) + '}');

        COMMIT TRANSACTION;

        SELECT 'Transacción registrada exitosamente' AS Resultado, @TransaccionID AS TransaccionID;
    END TRY
    BEGIN CATCH
        IF @@TRANCOUNT > 0
```

```
        ROLLBACK TRANSACTION;

    INSERT INTO dbo.AuditoriaLog (UsuarioID, Accion, TablaAfectada, DetallesNuevos)
    VALUES (@UsuarioID, 'ERROR', 'TransaccionesFinancieras',
        '{"Error":"' + ERROR_MESSAGE() + '","TipoTransaccion":"' + @TipoTransaccion + '"}');

    SELECT 'Error al registrar transacción: ' + ERROR_MESSAGE() AS Resultado;
  END CATCH
END;
GO
```

1. Configuración de Database Mirroring (Espejo)

Objetivo: Establecer un espejo para alta disponibilidad y protección contra fallos.

1.1. Preparar la base de datos para mirroring

```
-- En el servidor principal (PRIMARYSERVER\SQLPRIMARY)
USE master;
GO

-- Asegurar que la BD usa modelo FULL RECOVERY
ALTER DATABASE DB_SeguridadPractica SET RECOVERY FULL;
GO

-- Crear backup completo y de logs
BACKUP DATABASE DB_SeguridadPractica
TO DISK = 'C:\Mirroring\DB_SeguridadPractica_Full.bak'
WITH COMPRESSION, CHECKSUM;
```

```
GO

BACKUP LOG DB_SeguridadPractica
TO DISK = 'C:\Mirroring\DB_SeguridadPractica_Log.trn'
WITH COMPRESSION, CHECKSUM;
GO
```

1.2. Restaurar la BD en el servidor espejo

```
-- En el servidor espejo (MIRRORSERVER\SQLMIRROR)
USE master;
GO

-- Restaurar con NORECOVERY
RESTORE DATABASE DB_SeguridadPractica
FROM DISK = '\\PRIMARYSERVER\Mirroring\DB_SeguridadPractica_Full.bak'
WITH NORECOVERY,
MOVE 'DB_SeguridadPractica' TO 'C:\Data\DB_SeguridadPractica_Mirror.mdf',
MOVE 'DB_SeguridadPractica_log' TO 'C:\Data\DB_SeguridadPractica_Mirror.ldf';
GO

-- Restaurar logs
RESTORE LOG DB_SeguridadPractica
FROM DISK = '\\PRIMARYSERVER\Mirroring\DB_SeguridadPractica_Log.trn'
WITH NORECOVERY;
GO
```

1.3. Configurar el mirroring

```
-- En ambos servidores (crear endpoint)
```

```
CREATE ENDPOINT EndpointMirroring
STATE = STARTED
AS TCP (LISTENER_PORT = 5022)
FOR DATABASE_MIRRORING (ROLE = PARTNER);
GO

-- En el principal, iniciar mirroring
ALTER DATABASE DB_SeguridadPractica
SET PARTNER = 'TCP://MIRRORSERVER:5022';
GO

-- En el espejo, completar configuración
ALTER DATABASE DB_SeguridadPractica
SET PARTNER = 'TCP://PRIMARYSERVER:5022';
GO
```

1.4. Verificar el estado

```
-- Consultar estado del mirroring
SELECT
   database_name,
   mirroring_state_desc,
   mirroring_role_desc
FROM sys.database_mirroring
WHERE database_id = DB_ID('DB_SeguridadPractica');
GO
```

2. Configuración de Replicación Transaccional

Objetivo: Distribuir datos para reporting sin afectar rendimiento.

2.1. Configurar el Publicador

```
-- En el servidor principal (PUBLISHER)
USE master;
GO

-- Habilitar distribución
EXEC sp_adddistributor @distributor = 'PUBLISHER';
EXEC sp_adddistributiondb @database = 'distribution';
GO

-- Configurar publicación
USE DB_SeguridadPractica;
EXEC sp_replicationdboption @dbname = 'DB_SeguridadPractica',
@optname = 'publish', @value = 'true';
GO

-- Crear publicación (excluir tablas sensibles como AuditoriaLog)
EXEC sp_addpublication
@publication = 'DB_SeguridadPractica_Pub',
@sync_method = 'native',
@repl_freq = 'continuous',
@status = 'active';
GO

-- Agregar artículos (tablas)
EXEC sp_addarticle
@publication = 'DB_SeguridadPractica_Pub',
```

```
@article = 'Productos',
@source_owner = 'dbo',
@source_object = 'Productos';
GO
```

2.2. Configurar el Suscriptor

```
-- En el servidor secundario (SUBSCRIBER)
USE master;
GO

-- Crear suscripción push
EXEC sp_addsubscription
@publication = 'DB_SeguridadPractica_Pub',
@subscriber = 'SUBSCRIBER',
@destination_db = 'DB_SeguridadPractica_Replica',
@subscription_type = 'Push';
GO
```

2.3. Monitorear replicación

```
-- Verificar estado
USE distribution;
GO

SELECT
   a.publisher_db,
   a.article,
   a.status
FROM dbo.MSarticles a
JOIN dbo.MSpublications p ON a.publication_id = p.publication_id;
```

```
GO
```

3. Estrategias de Respaldo y Recuperación

3.1. Backups completos, diferenciales y de logs

```
-- Backup completo semanal
BACKUP DATABASE DB_SeguridadPractica
TO DISK = 'C:\Backups\DB_SeguridadPractica_Full_$(date).bak'
WITH COMPRESSION, CHECKSUM, STATS = 10;
GO

-- Backup diferencial diario
BACKUP DATABASE DB_SeguridadPractica
TO DISK = 'C:\Backups\DB_SeguridadPractica_Diff_$(date).bak'
WITH DIFFERENTIAL, COMPRESSION, CHECKSUM;
GO

-- Backup de logs cada hora
BACKUP LOG DB_SeguridadPractica
TO DISK = 'C:\Backups\Logs\DB_SeguridadPractica_Log_$(time).trn'
WITH COMPRESSION, CHECKSUM;
GO
```

3.2. Recuperación a punto en el tiempo (PITR)

```
-- Restaurar full + diff + logs hasta un momento específico
RESTORE DATABASE DB_SeguridadPractica
FROM DISK = 'C:\Backups\DB_SeguridadPractica_Full.bak'
WITH NORECOVERY, REPLACE;
GO
```

```
RESTORE DATABASE DB_SeguridadPractica
FROM DISK = 'C:\Backups\DB_SeguridadPractica_Diff.bak'
WITH NORECOVERY;
GO

RESTORE LOG DB_SeguridadPractica
FROM DISK = 'C:\Backups\Logs\DB_SeguridadPractica_Log.trn'
WITH STOPAT = '2023-11-15 14:00:00', RECOVERY;
GO
```

4. Migración Segura de la Base de Datos

4.1. Migración mediante Backup/Restore con Encriptación

```
-- En el servidor origen (exportar certificado)
USE master;
GO

BACKUP CERTIFICATE CertificadoBackup
TO FILE = 'C:\Migracion\CertificadoBackup.cer'
WITH PRIVATE KEY (
   FILE = 'C:\Migracion\CertificadoBackup.pvk',
   ENCRYPTION BY PASSWORD = 'M1gr@cionP@ss!'
);
GO

-- En el servidor destino (importar certificado)
CREATE CERTIFICATE CertificadoBackup
FROM FILE = 'C:\Migracion\CertificadoBackup.cer'
WITH PRIVATE KEY (
   FILE = 'C:\Migracion\CertificadoBackup.pvk',
```

```
   DECRYPTION BY PASSWORD = 'M1gr@cionP@ss!'
);
GO

-- Restaurar BD encriptada
RESTORE DATABASE DB_SeguridadPractica
FROM DISK = 'C:\Migracion\DB_SeguridadPractica_Encriptado.bak'
WITH REPLACE;
GO
```

4.2. Validar integridad post-migración

```
-- Verificar datos críticos
USE DB_SeguridadPractica;
GO

SELECT
   (SELECT COUNT(*) FROM Usuarios) AS TotalUsuarios,
   (SELECT COUNT(*) FROM Productos) AS TotalProductos,
   (SELECT COUNT(*) FROM AuditoriaLog) AS TotalLogs;
GO

-- Probar funciones de seguridad
SELECT dbo.DesencriptarSalario(1);  -- Debe devolver el salario correcto
GO
```