**Universidad Peruana Los Andes**
**Facultad de Ingeniería**
**Escuela Profesional de Ingeniería de Sistemas y Computación**

# Examen Final

## Administración de Base de Datos



Apellidos y Nombres: **Mosquera Zevallos Valerio** Código: **R03398D**
Ciclo: **5° Ciclo** Salón: **A1**

## Enunciado 01:

De acuerdo con la **base de datos** desarrollada en **Microsoft SQL Server**, responda las siguientes preguntas:

1) **No se encuentran entradas de índice.**

**- La base de datos que elabore resuelve la difícil manera de gestionar una variedad de productos sin una estructura clara tambien los productos sin categorías pueden estar desorganizados, dificultando la navegación para los clientes**
**Es necesario almacenar información de los clientes de forma segura y accesible**

2) **Implemente un Script para crear una vista para crear utilizando tres tablas**

**- En mi caso utilizare 3 tablas en mi bd "Clientes, Pedidos y Detalle_Pedido"**

```
-- Verificar si la vista ya existe y eliminarla si es necesario
IF OBJECT_ID('ResumenPedidos', 'V') IS NOT NULL
    DROP VIEW ResumenPedidos;
GO
-- Crear la vista ResumenPedidos
CREATE VIEW ResumenPedidos AS
SELECT
    c.nombre AS NombreCliente,
    c.correo AS CorreoCliente,
    p.id_pedido AS PedidoID,
    p.fecha_pedido AS FechaPedido,
    p.estado AS EstadoPedido,
    dp.id_producto AS ProductoID,
    dp.cantidad AS Cantidad,
    dp.precio_unitario AS PrecioUnitario,
    (dp.cantidad * dp.precio_unitario) AS TotalProducto
FROM
    Clientes c
INNER JOIN
    Pedidos p ON c.id_cliente = p.id_cliente
INNER JOIN
    Detalle_Pedido dp ON p.id_pedido = dp.id_pedido;
GO


-- Consultar la vista para verificar su contenido
SELECT * FROM ResumenPedidos;
```

| NombreCliente | CorreoCliente | Pedido ID | Fecha Pedido | Estado Pedido | Producto ID | Cantidad | Precio Unitario | Total Producto |
|---|---|---|---|---|---|---|---|---|
| Valerio Mosquera | valerio.mosquera@mail.com | 1 | 2024-12-20 10:00:00 | Pendiente | 1 | 2 | 599.99 | 1199.98 |
| María López | maria.lopez@mail.com | 2 | 2024-12-20 11:00:00 | Enviado | 2 | 1 | 799.99 | 799.99 |





## 3) Implemente un Script para crear un procedimiento almacenado para modificar el ingreso de datos en forma secuencial

**Script para Crear el Procedimiento Almacenado**

```
-- Verificar si el procedimiento ya existe y eliminarlo si es necesario
IF OBJECT_ID('ModificarStockSecuencial', 'P') IS NOT NULL
   DROP PROCEDURE ModificarStockSecuencial;
GO

-- Crear el procedimiento almacenado
CREATE PROCEDURE ModificarStockSecuencial
   @CantidadReduccion INT
AS
BEGIN
   SET NOCOUNT ON;

   -- Variable para controlar el ciclo
   DECLARE @ProductoID INT;

   -- Cursor para recorrer secuencialmente los productos
   DECLARE cursorProductos CURSOR FOR
   SELECT id_producto FROM Productos;

   -- Abrir el cursor
   OPEN cursorProductos;

   -- Obtener el primer producto
   FETCH NEXT FROM cursorProductos INTO @ProductoID;

   -- Ciclo para recorrer los productos
   WHILE @@FETCH_STATUS = 0
   BEGIN
      -- Actualizar el stock del producto actual
      UPDATE Productos
      SET stock = CASE
            WHEN stock >= @CantidadReduccion THEN stock -
@CantidadReduccion
            ELSE 0 -- Si el stock es menor a la cantidad de reducción, lo deja en 0
         END
```

```
        WHERE id_producto = @ProductoID;

        -- Obtener el siguiente producto
        FETCH NEXT FROM cursorProductos INTO @ProductoID;
    END

    -- Cerrar y liberar el cursor
    CLOSE cursorProductos;
    DEALLOCATE cursorProductos;

    PRINT 'Stock modificado secuencialmente.';
END;
GO
```

**Ejecutar el Procedimiento**

```
EXEC ModificarStockSecuencial @CantidadReduccion = 5;
```

| id_producto | stock |
|---|---|
| **1** | 100 |
| **2** | 50 |
| **3** | 200 |

| id_producto | stock |
|---|---|
| **1** | 95 |
| **2** | 45 |
| **3** | 195 |

## 4) Implemente un Script para crear un disparador para verificar el control de datos (Ejemplo: que la nota ingresada este entre 0 y 20)

### Script para Crear el Disparador

```
-- Crear la tabla Notas para el ejemplo
IF OBJECT_ID('Notas', 'U') IS NOT NULL
    DROP TABLE Notas;
GO

CREATE TABLE Notas (
    id_nota INT IDENTITY(1,1) PRIMARY KEY,
    id_estudiante INT NOT NULL,
    nota DECIMAL(5,2) NOT NULL
);
GO

-- Crear el disparador para verificar que la nota esté entre 0 y 20
CREATE TRIGGER trg_VerificarNota
ON Notas
```

```
AFTER INSERT, UPDATE
AS
BEGIN
    SET NOCOUNT ON;

    -- Verificar si alguna nota está fuera del rango permitido
    IF EXISTS (
        SELECT 1
        FROM inserted
        WHERE nota < 0 OR nota > 20
    )
    BEGIN
        -- Lanza un error si alguna nota está fuera del rango
        RAISERROR ('La nota debe estar entre 0 y 20.', 16, 1);
        ROLLBACK TRANSACTION; -- Revierte la transacción
    END
END;
GO
```

**Explicación del Script**

1. **Tabla Notas**:
   Contiene las columnas id_nota, id_estudiante y nota.
2. **Disparador trg_VerificarNota**:
   Se activa **después** de una operación INSERT o UPDATE.
   Utiliza la tabla inserted, que contiene los registros que se intentan insertar o actualizar.
3. **Verificación del Rango**:
   Realiza una consulta en la tabla inserted para verificar si alguna nota está fuera del rango permitido (menor a 0 o mayor a 20).
4. Si encuentra un valor fuera del rango, usa RAISERROR para mostrar un mensaje y cancela la operación con ROLLBACK TRANSACTION.

## 5) Utilizando Script Crear 03 usuarios con nombres de sus compañeros y uno suyo

# Script para Crear Usuarios

```
-- Asegúrate de estar en la base de datos TiendaOnline
USE TiendaOnline;
GO

-- Crear usuarios para la base de datos
CREATE LOGIN UsuarioValerio WITH PASSWORD = 'Valerio123!';
CREATE LOGIN UsuarioJuan WITH PASSWORD = 'Juan123!';
CREATE LOGIN UsuarioMaria WITH PASSWORD = 'Maria123!';
```

```
CREATE LOGIN UsuarioCarlos WITH PASSWORD = 'Carlos123!';
GO

-- Asociar los usuarios de base de datos
CREATE USER Valerio FOR LOGIN UsuarioValerio;
CREATE USER Juan FOR LOGIN UsuarioJuan;
CREATE USER Maria FOR LOGIN UsuarioMaria;
CREATE USER Carlos FOR LOGIN UsuarioCarlos;
GO

-- Asignar roles básicos a los usuarios
-- Rol db_datareader: Permite leer datos
-- Rol db_datawriter: Permite escribir datos
ALTER ROLE db_datareader ADD MEMBER Valerio;
ALTER ROLE db_datawriter ADD MEMBER Valerio;

ALTER ROLE db_datareader ADD MEMBER Juan;
ALTER ROLE db_datawriter ADD MEMBER Juan;

ALTER ROLE db_datareader ADD MEMBER Maria;
ALTER ROLE db_datawriter ADD MEMBER Maria;

ALTER ROLE db_datareader ADD MEMBER Carlos;
ALTER ROLE db_datawriter ADD MEMBER Carlos;
GO


-- Confirmar que los usuarios han sido creados y tienen roles asignados
SELECT name, type_desc
FROM sys.database_principals
WHERE type IN ('S', 'U') AND name NOT LIKE '##%';
GO
```



**Explicación del Script**

1. **Creación de Logins**:
   Se crean inicios de sesión para cada usuario (UsuarioValerio, UsuarioJuan, UsuarioMaria, UsuarioCarlos).
   Cada login tiene una contraseña asignada. Asegúrate de seguir las políticas de contraseñas del servidor.
2. **Creación de Usuarios**:
   Cada login se asocia con un usuario de base de datos (ejemplo: Valerio asociado con UsuarioValerio).
3. **Asignación de Roles**:
   **db_datareader**: Permite leer datos en todas las tablas de la base de datos.
   **db_datawriter**: Permite insertar, actualizar y eliminar datos en todas las tablas.
4. **Consulta de Verificación**:
   Se ejecuta una consulta para listar los usuarios de la base de datos y confirmar su creación.

## 6) Utilizando un script, copiar la base de datos (creada anteriormente) y compartir en cada uno de los usuarios

**Script para Copiar la Base de Datos y Compartir con Usuarios**

1. **Crear una copia de seguridad de la base de datos original:**

```
-- Crear un backup de la base de datos original
BACKUP DATABASE TiendaOnline
TO DISK = 'C:\Backups\TiendaOnline.bak'
WITH FORMAT, NAME = 'Backup de TiendaOnline';
GO
```



2. **Restaurar la base de datos con un nuevo nombre:**

```
-- Restaurar la base de datos con un nuevo nombre
RESTORE DATABASE TiendaOnlineCopia
FROM DISK = 'C:\Backups\TiendaOnline.bak'
WITH MOVE 'TiendaOnline' TO 'C:\SQLData\TiendaOnlineCopia.mdf',
    MOVE 'TiendaOnline_log' TO 'C:\SQLData\TiendaOnlineCopia_log.ldf';
GO
```

3. **Asignar permisos a los usuarios en la base de datos copiada:**

```
-- Asignar usuarios a la base de datos copiada
USE TiendaOnlineCopia;
GO

CREATE USER Valerio FOR LOGIN UsuarioValerio;
CREATE USER Juan FOR LOGIN UsuarioJuan;
CREATE USER Maria FOR LOGIN UsuarioMaria;
CREATE USER Carlos FOR LOGIN UsuarioCarlos;
GO

-- Asignar roles a los usuarios
ALTER ROLE db_datareader ADD MEMBER Valerio;
ALTER ROLE db_datawriter ADD MEMBER Valerio;

ALTER ROLE db_datareader ADD MEMBER Juan;
ALTER ROLE db_datawriter ADD MEMBER Juan;

ALTER ROLE db_datareader ADD MEMBER Maria;
ALTER ROLE db_datawriter ADD MEMBER Maria;

ALTER ROLE db_datareader ADD MEMBER Carlos;
ALTER ROLE db_datawriter ADD MEMBER Carlos;
GO

-- Confirmar que los usuarios están creados y tienen permisos
SELECT name, type_desc
FROM sys.database_principals
WHERE type IN ('S', 'U') AND name NOT LIKE '##%';
GO
```

**Explicación del Script**

**Backup**:

Realiza una copia de seguridad de la base de datos original (TiendaOnline).

El archivo de copia se guarda en una ruta específica (C:\Backups\TiendaOnline.bak).
**Restore**:
Restaura la base de datos con un nuevo nombre (TiendaOnlineCopia).
Se mueven los archivos físicos (.mdf y .ldf) a nuevas rutas.
**Asignación de Usuarios**:
Los mismos usuarios (Valerio, Juan, Maria, Carlos) se crean en la base de datos restaurada.
Se asignan roles básicos de lectura y escritura.
**Verificación**:
Se ejecuta una consulta para confirmar la creación de usuarios en la nueva base de datos.



## 7) Utilizando un script, generar una copia de seguridad de la base de datos y compartir a cada uno de los usuarios

**Script para Generar una Copia de Seguridad de la Base de Datos y Compartir con Usuarios**

1. **Generar el Respaldo de la Base de Datos (TiendaOnline)**

```
-- Generar un respaldo completo de la base de datos
BACKUP DATABASE TiendaOnline
TO DISK = 'C:\Backups\TiendaOnline_FullBackup.bak'
WITH FORMAT, NAME = 'Respaldo Completo de TiendaOnline',
    DESCRIPTION = 'Copia de seguridad completa de la base de datos
TiendaOnline',
    STATS = 10;
GO
```

**2. Asignar Permisos a los Usuarios para Acceder al Respaldo**
Una vez generado el respaldo, puedes compartirlo mediante permisos en el sistema operativo del servidor. Sin embargo, en SQL Server, puedes asegurarte de que cada usuario tenga acceso al archivo utilizando su login.

1. **Crear Logins y Usuarios (si no existen):**

```
-- Crear logins en el servidor (si no están creados previamente)
CREATE LOGIN UsuarioValerio WITH PASSWORD = 'Password123';
CREATE LOGIN UsuarioJuan WITH PASSWORD = 'Password123';
CREATE LOGIN UsuarioMaria WITH PASSWORD = 'Password123';
GO

-- Crear usuarios dentro de la base de datos
USE TiendaOnline;
GO

CREATE USER Valerio FOR LOGIN UsuarioValerio;
CREATE USER Juan FOR LOGIN UsuarioJuan;
CREATE USER Maria FOR LOGIN UsuarioMaria;
GO
```

2. **Dar Permisos a los Usuarios en la Base de Datos:**

```
-- Asignar permisos de lectura para consultar información de respaldo
```

```
USE TiendaOnline;
GO

GRANT VIEW DATABASE STATE TO Valerio;
GRANT VIEW DATABASE STATE TO Juan;
GRANT VIEW DATABASE STATE TO Maria;
GO
```



**3. Compartir el Respaldo con los Usuarios**

Los usuarios pueden acceder al respaldo si tienen acceso al servidor donde se guarda el archivo. Asegúrate de:

1. **Compartir el Directorio de Backup (Windows):**
   - Navega al directorio del respaldo (C:\Backups).
   - Haz clic derecho en la carpeta y selecciona "Compartir con...".
   - Agrega a los usuarios correspondientes y asigna permisos de solo lectura.
2. **Informar a los Usuarios sobre la Ubicación del Respaldo:**
   - Proporciónales la ruta completa del archivo de respaldo: \\ServidorSQL\Backups\TiendaOnline_FullBackup.bak.

**4. Restaurar el Respaldo (si es necesario)**

Cada usuario puede restaurar el respaldo en su instancia de SQL Server ejecutando el siguiente script:

```
-- Restaurar la base de datos desde el respaldo compartido
RESTORE DATABASE TiendaOnline_Restaurada
FROM DISK = '\\ServidorSQL\Backups\TiendaOnline_FullBackup.bak'
WITH MOVE 'TiendaOnline' TO 'C:\SQLData\TiendaOnline_Restaurada.mdf',
    MOVE 'TiendaOnline_log' TO 'C:\SQLData\TiendaOnline_Restaurada_log.ldf',
    STATS = 10;
GO
```

## 8) Utilizando un script, encriptar una de las tablas para que no se puedan ver los datos

**Script para Encriptar los Datos de una Tabla**

**1. Configurar un Entorno de Encriptación**

```
-- Crear una clave maestra en la base de datos si no existe
USE TiendaOnline;
GO

CREATE MASTER KEY ENCRYPTION BY PASSWORD = 'PasswordFuerte123!';
GO

-- Crear un certificado para encriptación
CREATE CERTIFICATE CertificadoClientes
WITH SUBJECT = 'Encriptación de datos sensibles en la tabla Clientes';
GO

-- Crear una clave simétrica protegida por el certificado
CREATE SYMMETRIC KEY ClaveSimetricaClientes
WITH ALGORITHM = AES_256
```

```
ENCRYPTION BY CERTIFICATE CertificadoClientes;
GO
```

**2. Encriptar los Datos Sensibles**

```
-- Abrir la clave simétrica
OPEN SYMMETRIC KEY ClaveSimetricaClientes
DECRYPTION BY CERTIFICATE CertificadoClientes;
GO

-- Actualizar los datos encriptando la columna "direccion"
UPDATE Clientes
SET direccion = EncryptByKey(Key_GUID('ClaveSimetricaClientes'),
CAST(direccion AS NVARCHAR(MAX)));
GO

-- Cerrar la clave simétrica
CLOSE SYMMETRIC KEY ClaveSimetricaClientes;
GO
```

**3. Verificar que los Datos Están Encriptados**

```
-- Consultar la tabla para verificar los datos encriptados
SELECT id_cliente, nombre, correo, direccion
FROM Clientes;
GO
```

La columna direccion mostrará datos ilegibles (encriptados).

**4. Desencriptar los Datos (Solo si es Necesario)**

```
-- Abrir la clave simétrica
OPEN SYMMETRIC KEY ClaveSimetricaClientes
DECRYPTION BY CERTIFICATE CertificadoClientes;
GO

-- Consultar los datos desencriptados
SELECT id_cliente,
    nombre,
    correo,
    CAST(DecryptByKey(direccion) AS NVARCHAR(MAX)) AS
direccion_desencriptada
FROM Clientes;
GO

-- Cerrar la clave simétrica
CLOSE SYMMETRIC KEY ClaveSimetricaClientes;
GO
```

## 9) Utilizando un script, aplique la seguridad a nivel de columna, restringiendo el acceso a la columna DNI de la tabla empleado en el usuario con nombre de su compañero.



**1. Crear el Usuario para Ivan Andy Aimituma Garcia**

Si el usuario no existe, se crea en la base de datos:

```
-- Crear un usuario para Ivan Andy Aimituma Garcia
CREATE USER IvanAndy FOR LOGIN IvanAndy;
GO
```

**2. Denegar Acceso a la Columna DNI**

Se deniega específicamente el acceso a la columna DNI de la tabla Empleado:

```
-- Denegar SELECT en la columna DNI para el usuario IvanAndy
DENY SELECT ON OBJECT::Empleado(DNI) TO IvanAndy;
GO
```

**3. Verificar el Acceso**

Como Administrador o Usuario con Permisos Completo:

```
-- Verificar que el administrador puede acceder a todos los datos
SELECT * FROM Empleado;
GO
```

**Como el Usuario IvanAndy:**

Cuando Ivan Andy intente acceder a la columna DNI, verá un error:

```
-- Consulta denegada para Ivan Andy
SELECT DNI FROM Empleado;
GO
```

**Ivan Andy puede consultar las otras columnas, pero no la columna DNI:**

```
-- Consulta permitida para otras columnas
SELECT id_empleado, nombre, cargo FROM Empleado;
GO
```

## 10) Utilizando un script, implementé seguridad a nivel de columna restringiendo el acceso a una de las columnas de una tabla.

**1. Crear un Usuario**

Primero, asegúrate de que el usuario exista. En este caso, usamos un ejemplo genérico, pero puedes personalizar el nombre del usuario.

```
-- Crear un usuario para ejemplo
CREATE USER UsuarioEjemplo FOR LOGIN UsuarioEjemploLogin;
GO
```

**2. Denegar Acceso a la Columna**

El acceso a la columna correo se deniega específicamente para el usuario creado:

```
-- Denegar SELECT en la columna correo para el usuario UsuarioEjemplo
DENY SELECT ON OBJECT::Clientes(correo) TO UsuarioEjemplo;
GO
```

**3. Verificar la Configuración**

Como Administrador:

El administrador sigue teniendo acceso completo a la tabla:

```
-- Verificar que el administrador puede ver todos los datos
SELECT * FROM Clientes;
GO
```



**Como el Usuario Restringido:**

El usuario con acceso restringido no podrá consultar la columna correo:

```
-- Esto generará un error para UsuarioEjemplo
SELECT correo FROM Clientes;
GO
```

**Sin embargo, puede consultar las demás columnas:**

```
-- Esto será permitido para UsuarioEjemplo
SELECT id_cliente, nombre, telefono, direccion FROM Clientes;
GO
```

## 11) Utilizando un script, realice el cifrado transparente de datos (TDE) para una las tablas.

**1. Configurar TDE para la Base de Datos**

**Paso 1. Crear una Llave Maestra en la Base de Datos Maestra**

La llave maestra protege la jerarquía de cifrado.

```
USE master;
GO

-- Crear una llave maestra
CREATE MASTER KEY ENCRYPTION BY PASSWORD =
'ContraseñaSegura2023!';
GO
```

**Paso 2. Crear un Certificado para el Cifrado**

El certificado se usará para proteger la clave de cifrado de la base de datos.

```
-- Crear un certificado en la base de datos maestra
CREATE CERTIFICATE CertificadoTiendaOnline
WITH SUBJECT = 'Certificado para cifrado de TiendaOnline';
GO
```

**Paso 3. Crear la Clave de Cifrado de la Base de Datos**

La clave de cifrado se crea utilizando el certificado.



```
USE TiendaOnline;
GO

-- Crear una clave de cifrado de la base de datos
CREATE DATABASE ENCRYPTION KEY
WITH ALGORITHM = AES_256
ENCRYPTION BY SERVER CERTIFICATE CertificadoTiendaOnline;
GO
```



**Paso 4. Habilitar TDE en la Base de Datos**

Activa TDE para cifrar la base de datos.

```
-- Habilitar cifrado en la base de datos
ALTER DATABASE TiendaOnline
SET ENCRYPTION ON;
GO
```

**2. Verificar el Estado del Cifrado**

Puedes verificar el estado del cifrado con la siguiente consulta:

```
-- Consultar el estado de cifrado de la base de datos
SELECT
    name AS DatabaseName,
    is_encrypted AS EncryptionStatus
FROM sys.databases
WHERE name = 'TiendaOnline';
GO
```

Si EncryptionStatus es 1, el cifrado está activo.

## 12) Utilizando un script, configure el usuario con el nombre de su compañero para otorgar permisos de SELECT, INSERT, UPDATE y DELETE en la base de datos.

**Script para otorgar permisos en la base de datos**

```
-- Cambiar al contexto de la base de datos TiendaOnline
USE TiendaOnline;
GO

-- Otorgar permisos al usuario para realizar SELECT, INSERT, UPDATE y DELETE
en todas las tablas
GRANT SELECT, INSERT, UPDATE, DELETE ON SCHEMA::dbo TO [Ivan Andy
Aimituma Garcia];
GO
```

```
-- Verificar los permisos otorgados
SELECT
    dp.name AS UserName,
    dp.type_desc AS UserType,
    o.name AS ObjectName,
    p.permission_name AS Permission,
    p.state_desc AS PermissionState
FROM sys.database_permissions p
JOIN sys.database_principals dp ON p.grantee_principal_id = dp.principal_id
LEFT JOIN sys.objects o ON p.major_id = o.object_id
WHERE dp.name = 'Ivan Andy Aimituma Garcia';
GO
```



**Explicación del Script**

Cambio al contexto de la base de datos:
Se usa **USE TiendaOnline** para asegurar que los permisos se configuren en la base de datos correcta.

Permisos a nivel de esquema (dbo):
La instrucción **GRANT SELECT, INSERT, UPDATE, DELETE ON SCHEMA::dbo TO [Ivan Andy Aimituma Garcia]** otorga permisos a todas las tablas del esquema dbo.

Verificación de permisos:
La consulta final lista los permisos otorgados al usuario Ivan Andy Aimituma Garcia en la base de datos.

## 13) Utilizando un script, configure la auditoría para el seguimiento y registro de acciones en la base de datos

**Script para Configurar la Auditoría**
**-- 1. Crear un archivo de auditoría a nivel de servidor**

```
USE master;
GO

CREATE SERVER AUDIT TiendaOnline_Audit
TO FILE (
    FILEPATH = 'C:\SQLAudit\' -- Cambiar a la ruta deseada
)
WITH (
    QUEUE_DELAY = 1000, -- Tiempo de espera en milisegundos antes de escribir en el archivo
    ON_FAILURE = CONTINUE -- Continuar en caso de falla
);
GO

-- 2. Habilitar el archivo de auditoría
ALTER SERVER AUDIT TiendaOnline_Audit WITH (STATE = ON);
```

```
GO

-- Crear una especificación de auditoría para la base de datos TiendaOnline
USE TiendaOnline;
GO

CREATE DATABASE AUDIT SPECIFICATION TiendaOnline_DB_AuditSpec
FOR SERVER AUDIT TiendaOnline_Audit
ADD (SELECT ON SCHEMA::dbo BY PUBLIC), -- Auditar consultas SELECT en el esquema dbo
ADD (INSERT ON SCHEMA::dbo BY PUBLIC), -- Auditar inserciones en el esquema dbo
ADD (UPDATE ON SCHEMA::dbo BY PUBLIC), -- Auditar actualizaciones en el esquema dbo
ADD (DELETE ON SCHEMA::dbo BY PUBLIC)  -- Auditar eliminaciones en el esquema dbo
WITH (STATE = ON); -- Habilitar la especificación de auditoría
GO

-- 4. Verificar las auditorías configuradas
SELECT *
FROM sys.server_audits;
GO

SELECT *
FROM sys.database_audit_specifications;
GO
```



**Descripción del Script**

**Crear un archivo de auditoría:**
Se utiliza CREATE SERVER AUDIT para definir un archivo donde se almacenarán los registros de auditoría.
Cambia FILEPATH a una ruta válida en tu sistema.

**Habilitar el archivo de auditoría:**
ALTER SERVER AUDIT se usa para activar la auditoría a nivel de servidor.

**Especificación de auditoría a nivel de base de datos:**
Con CREATE DATABASE AUDIT SPECIFICATION, se configura qué acciones serán auditadas en la base de datos

**TiendaOnline.**

**Habilitar la especificación:**
La opción WITH (STATE = ON) asegura que la auditoría comience a registrar las acciones inmediatamente.

**Verificación:**
Las consultas finales muestran las auditorías y especificaciones configuradas en el servidor y la base de datos.

## 14) Utilizando un script, configure de la memoria y el disco duro

**Script para Configurar Memoria**

```
-- Configurar el mínimo y máximo de memoria para SQL Server
EXEC sp_configure 'show advanced options', 1; -- Habilitar opciones avanzadas
RECONFIGURE;

EXEC sp_configure 'min server memory (MB)', 1024; -- Configurar memoria mínima
en MB
RECONFIGURE;

EXEC sp_configure 'max server memory (MB)', 8192; -- Configurar memoria máxima
en MB
RECONFIGURE;

-- Verificar la configuración actual de memoria
SELECT
   name AS 'Configuración',
   value AS 'Valor actual',
   value_in_use AS 'Valor en uso'
FROM sys.configurations
WHERE name IN ('min server memory (MB)', 'max server memory (MB)');
GO
```

**Script para Configurar Ubicación de Archivos en Disco**

```
-- Mover o configurar la ubicación de los archivos de datos y logs
USE master;
GO

-- Detener la base de datos antes de mover los archivos
ALTER DATABASE TiendaOnline SET OFFLINE;
GO

-- Cambiar la ubicación de los archivos de datos y logs
ALTER DATABASE TiendaOnline
MODIFY FILE (NAME = 'TiendaOnline', FILENAME =
'D:\SQLData\TiendaOnline.mdf'); -- Cambiar la ubicación del archivo MDF
GO

ALTER DATABASE TiendaOnline
MODIFY FILE (NAME = 'TiendaOnline_log', FILENAME =
'D:\SQLData\TiendaOnline_log.ldf'); -- Cambiar la ubicación del archivo LDF
GO

-- Volver a poner la base de datos en línea
ALTER DATABASE TiendaOnline SET ONLINE;
GO
```

```
-- Verificar la ubicación actual de los archivos
SELECT
    name AS 'Nombre del archivo',
    physical_name AS 'Ubicación física',
    type_desc AS 'Tipo'
FROM sys.master_files
WHERE database_id = DB_ID('TiendaOnline');
GO
```



**Descripción del Script**

**Configuración de memoria:**
min server memory (MB): Define la cantidad mínima de memoria que puede usar SQL Server.
max server memory (MB): Define la cantidad máxima de memoria asignada a SQL Server.
Configuración de ubicación de archivos:

ALTER DATABASE ... MODIFY FILE: Cambia la ubicación de los archivos .mdf (datos) y .ldf (logs).
Es importante detener la base de datos con SET OFFLINE antes de mover los archivos.
Verificación de configuración:
Las consultas finales permiten confirmar que los cambios se hayan aplicado correctamente.

## 15) Utilizando un script, genere una copia de seguridad de la base de datos

**Script para Generar una Copia de Seguridad**

```
-- Generar una copia de seguridad completa de la base de datos TiendaOnline
BACKUP DATABASE TiendaOnline
TO DISK = 'D:\SQLBackups\TiendaOnline_FullBackup.bak'
WITH
    FORMAT, -- Sobrescribe el archivo de respaldo si ya existe
    INIT,   -- Inicializa el medio de respaldo
    NAME = 'Copia de Seguridad Completa de TiendaOnline', -- Nombre del respaldo
    DESCRIPTION = 'Respaldo completo de la base de datos TiendaOnline', --
Descripción del respaldo
    STATS = 10; -- Mostrar progreso del respaldo cada 10%
GO
```

**Descripción del Script**

**Directiva TO DISK:**

Especifica la ruta del archivo de respaldo. Modifica D:\SQLBackups\ según la ubicación en tu servidor.

Opciones:



FORMAT: Sobrescribe el archivo si ya existe.

INIT: Inicializa el archivo para el nuevo respaldo.

NAME: Proporciona un nombre descriptivo al respaldo.

DESCRIPTION: Añade detalles sobre el respaldo.

STATS = 10: Muestra el progreso del respaldo

**Nombre del archivo:**

**TiendaOnline_FullBackup.bak: El archivo que contendrá el respaldo.**

**Verificar el Respaldo**

Para comprobar que el respaldo se creó correctamente, puedes revisar la carpeta especificada (D:\SQLBackups\) para confirmar que el archivo .bak existe.

**Restaurar la Base de Datos (en caso necesario)**

Si necesitas restaurar la base de datos a partir del respaldo generado, usa el siguiente script:

```
-- Restaurar la base de datos TiendaOnline desde un respaldo
RESTORE DATABASE TiendaOnline
FROM DISK = 'D:\SQLBackups\TiendaOnline_FullBackup.bak'
WITH
   REPLACE, -- Sobrescribe la base de datos existente
   STATS = 10; -- Mostrar progreso de la restauración cada 10%
GO
```

## 16) Utilizando un script, genere la restauración de la base de datos

**Script para Restaurar la Base de Datos**

```
-- Restaurar la base de datos TiendaOnline desde un archivo de respaldo
RESTORE DATABASE TiendaOnline
FROM DISK = 'D:\SQLBackups\TiendaOnline_FullBackup.bak' -- Ruta del archivo de respaldo
```

```
WITH
    MOVE 'TiendaOnline' TO 'D:\SQLData\TiendaOnline.mdf', -- Ubicación del archivo de datos
    MOVE 'TiendaOnline_Log' TO 'D:\SQLData\TiendaOnline_Log.ldf', -- Ubicación del archivo de registro
    REPLACE, -- Sobrescribe la base de datos si ya existe
    RECOVERY, -- Finaliza el proceso de restauración y deja la base de datos operativa
    STATS = 10; -- Muestra el progreso cada 10%
GO
```



**Detalles del Script**

**FROM DISK:**

Especifica la ubicación del archivo de respaldo (.bak). Cambia D:\SQLBackups\TiendaOnline_FullBackup.bak por la ruta real en tu sistema.

**MOVE:**

Define las ubicaciones de los archivos físicos:

TiendaOnline.mdf (archivo de datos).

TiendaOnline_Log.ldf (archivo de registro).

Ajusta las rutas (D:\SQLData\) según tu configuración.

**REPLACE:**

Sobrescribe la base de datos existente si ya está creada.

**RECOVERY:**

Deja la base de datos lista para uso inmediato después de la restauración.

**STATS:**

Muestra el progreso de la restauración cada 10%.

Pasos Previos a la Restauración

**Verificar Usuarios Conectados:**

Si la base de datos está en uso, desconecta a los usuarios:

```
ALTER DATABASE TiendaOnline SET SINGLE_USER WITH ROLLBACK IMMEDIATE;
GO
```

Estado de la Base de Datos:

**Si es necesario, la base de datos debe estar en estado de restauración:**

```
ALTER DATABASE TiendaOnline SET OFFLINE WITH ROLLBACK IMMEDIATE;
```

```
GO
```

Comprobaciones Posteriores a la Restauración

**Asegúrate de que la base de datos esté en modo MULTI_USER después de restaurar:**

```
ALTER DATABASE TiendaOnline SET MULTI_USER;
GO
```



## 17) Utilizando un script, cree un espejo de la base de datos

**Pasos para Crear un Espejo de Base de Datos**

**1. Realizar una Copia de Seguridad en el Servidor Principal**

En el servidor principal, realiza un respaldo completo de la base de datos y del log de transacciones.

```
-- Copia de seguridad completa de la base de datos
BACKUP DATABASE TiendaOnline
TO DISK = 'D:\SQLBackups\TiendaOnline_FullBackup.bak';
GO

-- Copia de seguridad del log de transacciones
BACKUP LOG TiendaOnline
TO DISK = 'D:\SQLBackups\TiendaOnline_LogBackup.trn';
GO
```

**2. Restaurar la Base de Datos en el Servidor Espejo**

En el servidor espejo, restaura la base de datos en modo NORECOVERY.

```
-- Restaurar la base de datos en el servidor espejo
RESTORE DATABASE TiendaOnline
FROM DISK = 'D:\SQLBackups\TiendaOnline_FullBackup.bak'
WITH MOVE 'TiendaOnline' TO 'D:\SQLData\TiendaOnline.mdf',
    MOVE 'TiendaOnline_Log' TO 'D:\SQLData\TiendaOnline_Log.ldf',
    NORECOVERY;
GO

-- Restaurar el log de transacciones en el servidor espejo
RESTORE LOG TiendaOnline
FROM DISK = 'D:\SQLBackups\TiendaOnline_LogBackup.trn'
```

```
WITH NORECOVERY;
GO
```

**3. Configurar los Endpoints de Mirroring**

Crea los endpoints para permitir la comunicación entre los servidores. Esto debe realizarse tanto en el servidor principal como en el espejo.



**En el Servidor Principal**

```
CREATE ENDPOINT MirroringEndpoint
STATE = STARTED
AS TCP (LISTENER_PORT = 5022)
FOR DATABASE_MIRRORING (
   ROLE = PARTNER,
   AUTHENTICATION = WINDOWS NEGOTIATE,
   ENCRYPTION = REQUIRED ALGORITHM AES
);
GO
En el Servidor Espejo
sql
Copiar código
CREATE ENDPOINT MirroringEndpoint
STATE = STARTED
AS TCP (LISTENER_PORT = 5022)
FOR DATABASE_MIRRORING (
   ROLE = PARTNER,
   AUTHENTICATION = WINDOWS NEGOTIATE,
   ENCRYPTION = REQUIRED ALGORITHM AES
);
GO
```

**4. Configurar el Mirroring**

Establece la asociación entre el servidor principal y el espejo.

**En el Servidor Principal**

```
ALTER DATABASE TiendaOnline
SET PARTNER = 'TCP://<NombreServidorEspejo>:5022';
```

```
GO
```

**En el Servidor Espejo**

```
ALTER DATABASE TiendaOnline
SET PARTNER = 'TCP://<NombreServidorPrincipal>:5022';
GO
```



**5. Configurar un Testigo (Opcional)**

Si deseas agregar un servidor testigo para el modo de alta disponibilidad, realiza los siguientes pasos.

**Crear un Endpoint en el Testigo**

```
CREATE ENDPOINT MirroringWitness
STATE = STARTED
AS TCP (LISTENER_PORT = 5023)
FOR DATABASE_MIRRORING (
   ROLE = WITNESS,
   AUTHENTICATION = WINDOWS NEGOTIATE,
   ENCRYPTION = REQUIRED ALGORITHM AES
);
GO
```

**Configurar el Testigo desde el Servidor Principal**

```
ALTER DATABASE TiendaOnline
SET WITNESS = 'TCP://<NombreServidorTestigo>:5023';
GO
```

**5. Verificar el Estado del Mirroring**

Puedes comprobar el estado del mirroring ejecutando:

```
SELECT
   database_id,
   name,
   mirroring_state_desc,
```

```
    mirroring_role_desc,
    mirroring_partner_name
FROM sys.databases
WHERE name = 'TiendaOnline';
GO
```



## 18) Utilizando un script, realice la replicación de bases de datos

**Pasos para Configurar la Replicación**

**1. Configurar el Distribuidor**

El distribuidor es responsable de gestionar el proceso de replicación.

```
-- Configurar el distribuidor
EXEC sp_adddistributor
    @distributor = '<NombreServidorDistribuidor>',
    @password = 'DistribuidorContraseña';

-- Configurar la base de distribución
EXEC sp_adddistributiondb
    @database = 'Distribution',
    @data_folder = 'C:\SQLData',
    @log_folder = 'C:\SQLLogs',
    @min_distretention = 0,
    @max_distretention = 72,
    @history_retention = 48;

-- Asociar el distribuidor con el servidor
EXEC sp_adddistpublisher
    @publisher = '<NombreServidorDistribuidor>',
    @distribution_db = 'Distribution',
    @security_mode = 1;
GO
```

**2. Configurar el Publicador**

El publicador define qué datos o tablas se van a replicar.

```
-- Configurar el publicador
EXEC sp_replicationdboption
    @dbname = 'TiendaOnline',
    @optname = 'publish',
    @value = 'true';

-- Crear una publicación
EXEC sp_addpublication
    @publication = 'Publicacion_TiendaOnline',
    @status = N'active',
    @allow_push = N'true',
    @allow_pull = N'true',
    @retention = 0,
    @sync_method = N'concurrent',
    @repl_freq = N'continuous',
    @description = N'Publicación para TiendaOnline',
    @allow_anonymous = N'false',
    @immediate_sync = N'false',
    @enabled_for_internet = N'false';

-- Agregar un artículo (tabla) a la publicación
EXEC sp_addarticle
    @publication = 'Publicacion_TiendaOnline',
    @article = 'Clientes',
    @source_owner = 'dbo',
    @source_object = 'Clientes',
    @type = N'logbased',
    @description = N'Clientes Table',
    @creation_script = null,
    @pre_creation_cmd = N'drop',
    @schema_option = 0x000000000803509F,
```

```
    @identityrangemanagementoption = N'manual',
    @destination_table = 'Clientes',
    @destination_owner = 'dbo';
GO
```

**3. Configurar el Suscriptor**



El suscriptor recibe los datos replicados del publicador.

```
-- Configurar el suscriptor
EXEC sp_addsubscription
   @publication = 'Publicacion_TiendaOnline',
   @subscriber = '<NombreServidorSuscriptor>',
   @destination_db = 'TiendaOnline_Replica',
   @subscription_type = N'Push',
   @sync_type = N'automatic',
   @article = N'all',
   @update_mode = N'read only',
   @subscriber_type = 0;
GO
```

## 19) Explique que es Always On Availability Groups

# ¿Qué son los Always On Availability Groups?

**Always On Availability Groups** es una funcionalidad de alta disponibilidad y recuperación ante desastres (HADR, por sus siglas en inglés) introducida en **Microsoft SQL Server 2012** y disponible en versiones posteriores. Es una solución avanzada que permite agrupar una o más bases de datos (llamadas **grupos de disponibilidad**) y mantenerlas replicadas en múltiples réplicas de bases de datos alojadas en diferentes instancias de SQL Server.

**Arquitectura y Componentes Principales**

1. **Grupo de Disponibilidad (Availability Group)**:
   - Un **grupo de disponibilidad** consiste en un conjunto de bases de datos específicas que deben ser gestionadas y protegidas como una unidad lógica.
   - Las bases de datos dentro del grupo pueden residir en diferentes réplicas de SQL Server, ya sea en el mismo datacenter o en ubicaciones geográficas remotas.
2. **Réplicas de Disponibilidad (Availability Replicas)**:
   - **Réplica Principal (Primary Replica)**:

     -Es la instancia que administra las transacciones de lectura/escritura.

-Es responsable de mantener la consistencia de los datos y sincronizarlos con las réplicas secundarias.

- **Réplicas Secundarias (Secondary Replicas)**:

  -Sirven como copias de solo lectura, permitiendo consultas y reportes sin afectar el rendimiento de la réplica principal.

  -Pueden configurarse como: **Sincrónicas**: Garantizan que los datos se reflejen en tiempo real en la réplica secundaria. **Asincrónicas**: Requieren menos latencia, pero pueden tener un retraso en la replicación.



3. **Listener (Escucha de Red)**:

   -Proporciona un nombre de red único que las aplicaciones pueden usar para conectarse al grupo de disponibilidad.

   -Los listeners redirigen automáticamente las solicitudes a la réplica principal activa, sin necesidad de reconfigurar las aplicaciones en caso de failover.

4. **Failover (Conmutación por Error)**:
   - Proceso mediante el cual una réplica secundaria se convierte en principal.
   - Tipos de failover:
     - **Automático**: Solo disponible para réplicas configuradas en modo sincrónico.
     - **Manual**: Requiere intervención del administrador.
5. **Integración con WSFC (Windows Server Failover Clustering)**:
   - Always On AG utiliza el clúster de conmutación por error de Windows para coordinar el failover y el monitoreo de las réplicas.
   - WSFC actúa como el sistema subyacente que asegura la alta disponibilidad a nivel del sistema operativo.

**Características Destacadas**

1. **Alta Disponibilidad y Recuperación Ante Desastres (HADR)**:
   - Always On permite replicar bases de datos críticas a varias réplicas, asegurando que los datos estén protegidos incluso en caso de fallos catastróficos.
   - La replicación puede extenderse a diferentes regiones geográficas.
2. **Escalabilidad para Lecturas**:
   - Las réplicas secundarias pueden configurarse para manejar cargas de lectura, como reportes y análisis, reduciendo el impacto en la réplica principal.
3. **Integridad y Consistencia**:
   - Los modos de sincronización garantizan que los datos replicados sean consistentes entre las réplicas, lo que es crucial en aplicaciones críticas.
4. **Capacidades de Monitoreo**:
   - SQL Server Management Studio (SSMS) proporciona herramientas para supervisar el estado del grupo de disponibilidad, las réplicas y las sesiones de sincronización.

5. **Compatibilidad con TDE (Transparent Data Encryption)**:
    - Las bases de datos cifradas con TDE son totalmente compatibles con Always On.
6. **Automatización de Failover**:
    - Permite una transición automática y rápida a una réplica secundaria en caso de fallos, minimizando los tiempos de inactividad.



**Beneficios Empresariales**

1. **Minimización del Tiempo de Inactividad**:
    - Las empresas que no pueden permitirse interrupciones prolongadas (bancos, comercio electrónico, hospitales) encuentran en Always On una solución confiable.
2. **Protección de Datos**:
    - En caso de pérdida de datos o fallos del sistema, las réplicas secundarias aseguran que haya copias actualizadas disponibles para restaurar.
3. **Ahorro en Costos de Infraestructura**:
    - Al utilizar réplicas secundarias para reportes, se optimiza el uso de recursos y se evita la necesidad de crear sistemas independientes para análisis de datos.
4. **Flexibilidad para Configuraciones Híbridas**:
    - Always On permite replicar datos entre servidores locales y en la nube, facilitando estrategias híbridas.

**Requisitos para Configurar Always On Availability Groups**

1. **Ediciones de SQL Server**:
    - Always On AG está disponible en la edición **Enterprise**.
    - En la edición **Standard**, solo se soportan Grupos de Disponibilidad Básicos (máximo una réplica secundaria y sin listeners).
2. **Clúster de Conmutación por Error (WSFC)**:
    - Es obligatorio configurar un clúster de Windows Server para habilitar Always On.
3. **Red de Alta Velocidad**:
    - Para réplicas sincrónicas, se requiere una red rápida y estable para garantizar la replicación en tiempo real.
4. **Licenciamiento**:
    - Cada instancia de SQL Server debe estar debidamente licenciada.

**Pasos Generales para Implementar Always On**

1. Configurar un clúster de conmutación por error en Windows.
2. Crear bases de datos en SQL Server y habilitarlas para Always On.

3. Configurar el grupo de disponibilidad y las réplicas.
4. Crear el listener para la conectividad de aplicaciones.
5. Probar el failover para asegurarse de que las réplicas secundarias puedan asumir el rol principal.

**Casos de Uso**



1. **Banca y Finanzas**:
    - Gestión de datos críticos con alta disponibilidad garantizada.
2. **Comercio Electrónico**:
    - Evitar interrupciones durante promociones y periodos de alta demanda.
3. **Sectores Sanitarios**:
    - Proteger datos médicos y asegurar disponibilidad en tiempo real.

**Limitaciones**

1. No todas las ediciones de SQL Server soportan todas las capacidades.
2. Requiere una configuración inicial compleja.
3. Las réplicas asincrónicas no garantizan consistencia en tiempo real.

**Diferencias con la Replicación Tradicional**

| Característica | Always On Availability Groups | Replicación Tradicional |
|---|---|---|
| **Unidad de Replicación** | Conjunto de bases de datos | Tablas específicas o bases de datos |
| **Objetivo Principal** | Alta disponibilidad y recuperación | Distribución de datos |
| **Requerimientos** | Clúster de conmutación por error | No requiere clúster |

**Resumen**

Always On Availability Groups es una tecnología avanzada que combina alta disponibilidad, recuperación ante desastres y escalabilidad en una solución única. Aunque su implementación puede requerir conocimientos avanzados y una infraestructura adecuada, los beneficios en términos de continuidad operativa y protección de datos lo convierten en una herramienta esencial para las empresas que operan en entornos críticos.

## 20) Explique que es Log Shipping



# ¿Qué es Log Shipping?



**Log Shipping** es una técnica de alta disponibilidad y recuperación ante desastres (HADR) disponible en **Microsoft SQL Server**. Se utiliza para mantener una copia de una base de datos en otro servidor (o múltiples servidores) al copiar periódicamente los archivos de registro de transacciones (log files) desde el servidor principal (primario) hacia uno o más servidores secundarios, donde se restauran para mantener la base de datos secundaria sincronizada.

Esta metodología asegura que los datos estén disponibles en caso de fallos en el servidor principal, aunque con un pequeño retraso en comparación con otros métodos como los Always On Availability Groups.

**Componentes Clave del Log Shipping**

1. **Servidor Primario**:
    - Es el servidor principal que contiene la base de datos de origen.
    - Genera los archivos de registro de transacciones (transaction log backups) de manera programada.
2. **Servidor Secundario**:
    - Es el servidor que contiene la copia de la base de datos principal.
    - Recibe y restaura los archivos de registro de transacciones enviados desde el servidor primario.
3. **Servidor de Monitoreo (Opcional)**:
    - Supervisa y registra el estado de las operaciones de log shipping.
    - Envía alertas en caso de fallos en la copia o restauración de los archivos.
4. **Archivos de Registro de Transacciones (Transaction Log Files)**:
    - Son los archivos generados en el servidor principal que contienen los cambios realizados en la base de datos.
5. **Jobs del Agente SQL Server**:
    - Tres trabajos automatizados controlan el proceso:
        1. **Copia de seguridad** (en el servidor primario): Crea los backups del log de transacciones.
        2. **Copia** (en el servidor secundario): Transfiere los archivos de registro desde el servidor primario al secundario.
        3. **Restauración** (en el servidor secundario): Aplica los archivos de registro para actualizar la base de datos secundaria.

**Cómo Funciona Log Shipping**

1. **Copia de Seguridad**:

   - El servidor primario realiza un backup incremental del log de transacciones en intervalos regulares y almacena los archivos en una carpeta compartida.

2. **Transferencia**:
   - El servidor secundario accede a la carpeta compartida y copia los archivos de log de transacciones.

3. **Restauración**:
   - Los archivos de log copiados se restauran en la base de datos secundaria para mantenerla sincronizada con la base de datos primaria.

4. **Monitoreo**:
   - El servidor de monitoreo supervisa el proceso para asegurar que la copia de seguridad, la transferencia y la restauración se realicen correctamente.

**Ventajas de Log Shipping**

1. **Alta Disponibilidad**:
   - Proporciona un método básico para mantener una copia actualizada de la base de datos en un servidor secundario, que puede ser utilizada en caso de fallos.

2. **Recuperación Ante Desastres**:
   - Los datos están disponibles en un servidor diferente, lo que facilita la recuperación en caso de que el servidor principal se dañe.

3. **Escalabilidad**:
   - Permite configurar múltiples servidores secundarios que actúan como copias de la base de datos principal.

4. **Facilidad de Configuración**:
   - Comparado con otras técnicas de alta disponibilidad, como Always On, Log Shipping es más fácil de configurar.

5. **Menor Impacto en el Servidor Principal**:
   - No afecta significativamente el rendimiento del servidor principal, ya que opera de manera programada.

**Limitaciones de Log Shipping**

1. **Retraso en la Sincronización**:
   - Existe un retraso inherente entre el momento en que se realiza un cambio en la base de datos primaria y el momento en que se refleja en la base de datos secundaria.

2. **Solo Soporte de Modo Manual para Failover**:
   - En caso de fallo del servidor principal, se requiere intervención manual para activar el servidor secundario como primario.

3. **Base de Datos en Solo Lectura**:
   - La base de datos secundaria permanece en estado de "restauración continua" y no puede ser utilizada para operaciones de lectura/escritura estándar.

4. **No Replica Configuraciones de Base de Datos**:
    - Configuraciones como logins o jobs no se transfieren automáticamente entre el servidor primario y los secundarios.
5. **No Proporciona Escalabilidad de Lecturas**:
    - A diferencia de Always On, la base de datos secundaria no puede ser utilizada para lecturas mientras está en modo restauración.



**Casos de Uso Común**

1. **Recuperación ante Desastres**:
    - Mantener una copia de seguridad actualizada en una ubicación física diferente.
2. **Migración de Datos**:
    - Ayuda a mover bases de datos de un servidor a otro con mínima interrupción.
3. **Ambientes de Pruebas**:
    - Configurar una copia actualizada de la base de datos principal para pruebas en un servidor secundario.

**Pasos Básicos para Configurar Log Shipping**

1. **Configurar el Servidor Primario**:

   -Realizar un backup completo de la base de datos inicial.

   -Crear un plan de respaldo periódico para los logs de transacciones.
2. **Configurar el Servidor Secundario**:

   -Restaurar el backup inicial de la base de datos en el servidor secundario utilizando la opción RESTORE WITH NORECOVERY.

   -Configurar las carpetas compartidas para que el servidor secundario pueda acceder a los logs.
3. **Habilitar Log Shipping**:
    - Usar SQL Server Management Studio (SSMS) para configurar el proceso de copia, transferencia y restauración.
4. **Probar la Configuración**:
    - Validar que los backups, la transferencia y la restauración funcionan correctamente.
5. **Configurar el Monitoreo**:
    - Agregar un servidor de monitoreo para alertas y reportes.

**Comparación con Otras Soluciones**

| Característica | Log Shipping | Always On Availability Groups | Replication |
|---|---|---|---|

| | | | |
|---|---|---|---|
| **Sincronización en Tiempo Real** | No | Sí | Sí (para transaccional) |
| **Escalabilidad de Lecturas** | No | Sí | Sí |
| **Intervención Manual en Failover** | Sí | No (si es configurado automático) | Sí |
| **Complejidad de Configuración** | Baja | Alta | Media |
| **Uso Principal** | HADR Básico | Alta Disponibilidad | Distribución de Datos |



**Conclusión**

Log Shipping es una solución sencilla y efectiva para garantizar la recuperación ante desastres y mantener una copia de la base de datos en un servidor secundario. Aunque tiene ciertas limitaciones, como la falta de sincronización en tiempo real y la necesidad de failover manual, sigue siendo una opción viable para entornos donde la simplicidad y el bajo costo son prioritarios.