

## Administración de Base de Datos

Implemente y explique un Script para crear una vista para crear utilizando tres tablas

The screenshot displays the Microsoft SQL Server Management Studio interface. The left pane shows the 'Object Explorer' with the 'gian DB' database selected. The central pane shows a SQL script with three queries, each preceded by a comment: '-- Ver la primera tabla', '-- Ver la segunda tabla', and '-- Ver la tercera tabla'. Red arrows point from these comments to the respective queries. The right pane shows the 'Results' tab with three tables of data. The first table has columns 'id', 'nombre', 'edad', and 'fecha\_registro'. The second table has columns 'id', 'nombre', 'edad', and 'sexo'. The third table has columns 'id', 'nombre', 'apellido', 'celularperuano', 'sexo', and 'edad'. Red arrows point from the comments to the corresponding result tables. The status bar at the bottom indicates 'Query executed successfully.'

```
SELECT * FROM tercera_tabla;
```

```
-- Ver la primera tabla
SELECT * FROM gian123;
```

```
-- Ver la segunda tabla
SELECT * FROM segunda_tabla;
```

```
-- Ver la tercera tabla
SELECT * FROM tercera_tabla;
```

id	nombre	edad	fecha_registro
1	Ana López	25	2024-11-01 00:00:00.000
2	Carlos Pérez	30	2024-11-02 00:00:00.000
3	María García	22	2024-11-03 00:00:00.000
4	Juan Rodríguez	45	2024-11-04 00:00:00.000
5	Laura Martínez	28	2024-11-05 00:00:00.000
6	José Sánchez	35	2024-11-06 00:00:00.000
7	Pedro González	40	2024-11-07 00:00:00.000
8	Sofía Rodríguez	26	2024-11-08 00:00:00.000

id	nombre	edad	sexo
1	Ana López	25	F
2	Carlos Pérez	30	M
3	María Gar...	22	F
4	Juan Rodr...	45	M
5	Laura Mart...	28	F
6	José Sánc...	35	M
7	Pedro Gon...	40	M
8	Sofía Rod...	26	F

id	nombre	apellido	celularperuano	sexo	edad
1	Ana	López	987654321	F	25
2	Carlos	Pérez	912345678	M	30
3	María	García	998877665	F	22
4	Juan	Rodrí...	999112233	M	45
5	Laura	Martí...	955334477	F	28
6	José	Sánc...	998877123	M	35
7	Pedro	Gonz...	987654999	M	40
8	Sofía	Rodrí...	951234567	F	26

Implemente y explique un Script para crear un procedimiento almacenado para insertar datos a su base de datos.

```

CREATE TABLE segunda_tabla (
    id INT PRIMARY KEY,          -- El identificador será un número entero, y será la clave primaria.
    nombre NVARCHAR(100),        -- El nombre será una cadena de texto (hasta 100 caracteres).
    edad INT,                    -- La edad será un número entero.
    sexo CHAR(1)                 -- El sexo será un carácter, 'M' para masculino o 'F' para femenino.
);

INSERT INTO segunda_tabla (id, nombre, edad, sexo) VALUES (1, 'Ana López', 25, 'F');
INSERT INTO segunda_tabla (id, nombre, edad, sexo) VALUES (2, 'Carlos Pérez', 30, 'M');
INSERT INTO segunda_tabla (id, nombre, edad, sexo) VALUES (3, 'María García', 22, 'F');
INSERT INTO segunda_tabla (id, nombre, edad, sexo) VALUES (4, 'Juan Rodríguez', 45, 'M');
INSERT INTO segunda_tabla (id, nombre, edad, sexo) VALUES (5, 'Laura Martínez', 28, 'F');

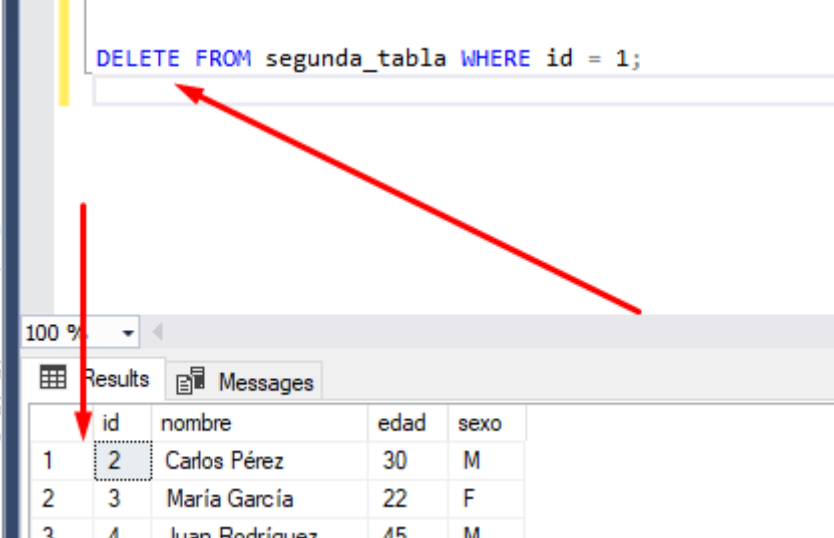
```

Implemente y explique un Script para crear un procedimiento almacenado para eliminar datos a su base de datos

```

DELETE FROM segunda_tabla WHERE id = 1;

```



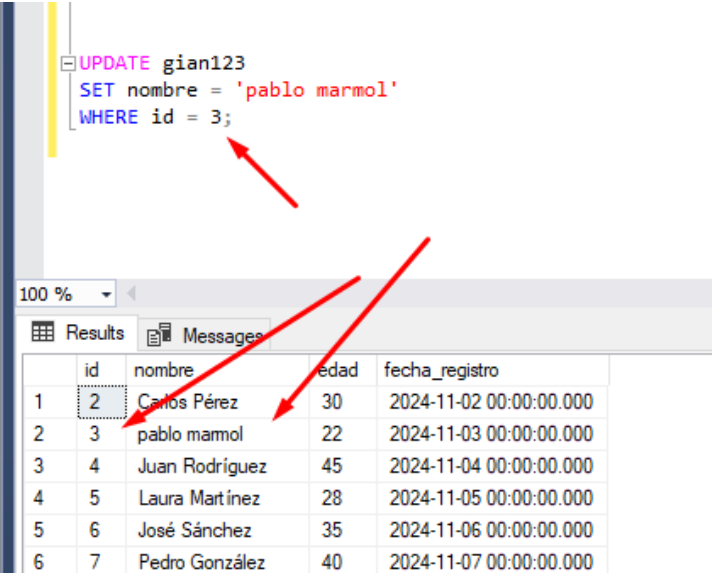
id	nombre	edad	sexo
1	Carlos Pérez	30	M
2	María García	22	F
3	Juan Rodríguez	45	M

Implemente y explique un Script para crear un procedimiento almacenado para actualizar datos a su base de datos

```

UPDATE gian123
SET nombre = 'pablo marmol'
WHERE id = 3;

```



id	nombre	edad	fecha_registro
1	Carlos Pérez	30	2024-11-02 00:00:00.000
2	pablo marmol	22	2024-11-03 00:00:00.000
3	Juan Rodríguez	45	2024-11-04 00:00:00.000
4	Laura Martínez	28	2024-11-05 00:00:00.000
5	José Sánchez	35	2024-11-06 00:00:00.000
6	Pedro González	40	2024-11-07 00:00:00.000

Implemente y explique un Script para crear un procedimiento almacenado para realizar cálculos matemáticos de una columna de su base de datos.

```
SELECT SUM(edad) AS suma_edad FROM tercera_tabla;
```

Results	
Messages	
suma_edad	
1	3148

Implemente y explique un Script para crear un disparador para ingresar un registro automáticamente en una tabla de su base de datos.

```
-- Crear el disparador
CREATE TRIGGER trg_insert_x3_tabla
ON x2_tabla
AFTER INSERT
AS
BEGIN
    -- Insertar un registro en tercera_tabla
    INSERT INTO tercera_tabla (id, nombre, apellido, celularperuano, sexo, edad)
    SELECT
        id,                -- Copia el 'id' del nuevo registro
        nombre,            -- Copia el 'nombre' del nuevo registro
        '',                -- Deja el campo 'apellido' vacío
        '000000000',       -- Proporciona un valor por defecto para 'celularperuano'
        sexo,              -- Copia el 'sexo' del nuevo registro
        edad               -- Copia la 'edad' del nuevo registro
    FROM inserted; -- 'inserted' es una tabla virtual que contiene los registros insertados
END;

SELECT name
FROM sys.triggers
WHERE name = 'trg_insert_x3_tabla';
```

Results	
Messages	
Client Statistics	
name	
1	trg_insert_x3_tabla

Implemente y explique un Script para crear un disparador para elimine un registro automáticamente en una tabla de su base de datos.

```
CREATE TRIGGER trg_delete_x3_tabla
ON x2_tabla
AFTER DELETE
AS
BEGIN
    -- Eliminar el registro correspondiente en tercera_tabla
    DELETE FROM x3_tabla
    WHERE id IN (SELECT id FROM deleted);

    PRINT 'El registro ha sido eliminado de tercera_tabla';
END;
```

Implemente y explique un Script para crear un disparador para actualice un registro automáticamente en una tabla de su base de datos.

```
CREATE TRIGGER trg_update_x3_tabla
ON x2_tabla
AFTER UPDATE
AS
BEGIN
    -- Actualizar los registros correspondientes en tercera_tabla
    UPDATE t
    SET t.nombre = i.nombre,
        t.edad = i.edad,
        t.sexo = i.sexo
    FROM tercera_tabla t
    INNER JOIN inserted i
        ON t.id = i.id;

    PRINT 'El registro en x3_tabla ha sido actualizado';
END;
```

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

```
CREATE TABLE calificaciones (
    id INT PRIMARY KEY,          -- Identificador único del registro
    estudiante NVARCHAR(100),    -- Nombre del estudiante
    nota INT                     -- Nota que obtuvo el estudiante
);
CREATE TRIGGER trg_verificar_nota
ON calificaciones
AFTER INSERT, UPDATE
AS
BEGIN
    -- Verificar si alguna nota no está en el rango válido (0 - 20)
    IF EXISTS (
        SELECT 1
        FROM inserted
        WHERE nota < 0 OR nota > 20
    )
    BEGIN
        -- Si se encuentra alguna nota inválida, se deshace la operación
        PRINT 'Error: La nota debe estar entre 0 y 20';

        -- Cancelar la operación (rollback)
        ROLLBACK TRANSACTION;
    END
    ELSE
    BEGIN
        -- Si las notas son válidas, permitir la operación
        PRINT 'La operación se realizó correctamente';
    END
END;
```

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

```
-- Crear login para Carlos Pérez
CREATE LOGIN CarlosPerez WITH PASSWORD = 'ContraseñaCarlos123';

-- Crear login para Ana López
CREATE LOGIN AnaLopez WITH PASSWORD = 'ContraseñaAna123';

-- Crear login para Luis Gómez
CREATE LOGIN LuisGomez WITH PASSWORD = 'ContraseñaLuis123';

-- Crear login para Juan Torres (tu hombre)
CREATE LOGIN JuanTorres WITH PASSWORD = 'ContraseñaJuan123';
```

133 %

Messages Client Statistics

Commands completed successfully.

Completion time: 2024-11-25T22:04:50.0107353-05:00

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

```
-- Otorgar acceso a los usuarios
USE mi_base_de_datos_copia;
```

```
CREATE USER CarlosPerez FOR LOGIN CarlosPerez;
CREATE USER AnaLopez FOR LOGIN AnaLopez;
CREATE USER LuisGomez FOR LOGIN LuisGomez;
CREATE USER JuanTorres FOR LOGIN JuanTorres;

ALTER ROLE db_datareader ADD MEMBER CarlosPerez;
ALTER ROLE db_datawriter ADD MEMBER CarlosPerez;

ALTER ROLE db_datareader ADD MEMBER AnaLopez;
ALTER ROLE db_datawriter ADD MEMBER AnaLopez;

ALTER ROLE db_datareader ADD MEMBER LuisGomez;
ALTER ROLE db_datawriter ADD MEMBER LuisGomez;

ALTER ROLE db_datareader ADD MEMBER JuanTorres;
ALTER ROLE db_datawriter ADD MEMBER JuanTorres;
```

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

```
-- Generar una copia de seguridad de la base de datos
BACKUP DATABASE mi_base_de_datos
TO DISK = 'C:\ruta\mi_base_de_datos.bak';

-- Otorgar permisos a los usuarios
GRANT BACKUP DATABASE TO CarlosPerez;
GRANT BACKUP DATABASE TO AnaLopez;
GRANT BACKUP DATABASE TO LuisGomez;
GRANT BACKUP DATABASE TO JuanTorres;
```

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

```
-- Otorgar permisos a los usuarios
GRANT BACKUP DATABASE TO CarlosPerez;
GRANT BACKUP DATABASE TO AnaLopez;
GRANT BACKUP DATABASE TO LuisGomez;
GRANT BACKUP DATABASE TO JuanTorres;
-- Crear una clave simétrica
CREATE SYMMETRIC KEY mi_clave_simetrica
WITH ALGORITHM = AES_256
ENCRYPTION BY PASSWORD = 'mi_contraseña_segura';

-- Abrir la clave simétrica
OPEN SYMMETRIC KEY mi_clave_simetrica DECRYPTION BY PASSWORD = 'mi_contraseña_segura';

-- Encriptar datos en una tabla
UPDATE mi_tabla
SET columna_encriptada = ENCRYPTBYKEY(KEY_GUID('mi_clave_simetrica'), columna_original);

-- Cerrar la clave simétrica
CLOSE SYMMETRIC KEY mi_clave_simetrica;
```

Utilizando un script, aplique la seguridad a nivel de columna, restringiendo el acceso a la columna que contiene la clave primaria de una de las tablas de su base de datos

```

-- Cerrar la clave simétrica
CLOSE SYMMETRIC KEY mi_clave_simetrica;

-- Crear un rol para restringir acceso
CREATE ROLE rol_restringido;

-- Otorgar acceso solo a las columnas que no sean la clave primaria
DENY SELECT ON mi_tabla (id) TO rol_restringido;
DENY INSERT ON mi_tabla (id) TO rol_restringido;
DENY UPDATE ON mi_tabla (id) TO rol_restringido;
DENY DELETE ON mi_tabla (id) TO rol_restringido;

-- Asignar usuarios al rol
ALTER ROLE rol_restringido ADD MEMBER CarlosPerez;
ALTER ROLE rol_restringido ADD MEMBER AnaLopez;

```

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

```

-- Denegar acceso a una columna específica
DENY SELECT ON mi_tabla (columna_secreta) TO CarlosPerez;
DENY INSERT ON mi_tabla (columna_secreta) TO CarlosPerez;
DENY UPDATE ON mi_tabla (columna_secreta) TO CarlosPerez;
DENY DELETE ON mi_tabla (columna_secreta) TO CarlosPerez;

```

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

```

-- Crear un certificado para TDE
CREATE CERTIFICATE mi_certificado
WITH SUBJECT = 'Cifrado Transparente de Datos';

-- Crear una clave de cifrado de base de datos
CREATE DATABASE ENCRYPTION KEY
WITH ALGORITHM = AES_256
ENCRYPTION BY CERTIFICATE mi_certificado;

-- Habilitar el cifrado de base de datos
ALTER DATABASE mi_base_de_datos
SET ENCRYPTION ON;

```

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.

```

-- Otorgar permisos de SELECT, INSERT, UPDATE y DELETE
GRANT SELECT, INSERT, UPDATE, DELETE ON mi_base_de_datos TO CarlosPerez;
GRANT SELECT, INSERT, UPDATE, DELETE ON mi_base_de_datos TO AnaLopez;
GRANT SELECT, INSERT, UPDATE, DELETE ON mi_base_de_datos TO LuisGomez;
GRANT SELECT, INSERT, UPDATE, DELETE ON mi_base_de_datos TO JuanTorres;

```

Utilizando un Scripts realice la validación y filtración de entradas del usuario para evitar caracteres maliciosos (Ejemplo: ', --, ;)

```
-- Validación de entradas para evitar caracteres maliciosos
DECLARE @input NVARCHAR(4000);
SET @input = 'entrada del usuario aquí'; -- Reemplaza esto con la entrada del usuario

-- Comprobamos si la entrada contiene caracteres maliciosos
IF @input LIKE '%\'' OR @input LIKE '%--' OR @input LIKE '%;'
BEGIN
    RAISERROR('Entrada maliciosa detectada. Por favor, revise los caracteres ingresados.', 16, 1);
    RETURN;
END

-- Si no se detectan caracteres maliciosos, se puede proceder con el insert
INSERT INTO mi_tabla (columnal) VALUES (@input);
```

Realice un script que verifiquen que los datos ingresados cumplan con formatos esperados (ej.: números en lugar de texto, longitud máxima).

```
-- Verificación de formato de entrada (números, longitud máxima)
DECLARE @edad INT = 'abc'; -- Ejemplo de valor incorrecto
DECLARE @telefono NVARCHAR(15) = '1234567890';

-- Verificar que la edad sea un número entero
IF ISNUMERIC(@edad) = 0
BEGIN
    RAISERROR('La edad debe ser un número válido.', 16, 1);
    RETURN;
END

-- Verificar que el teléfono tenga 9 dígitos
IF LEN(@telefono) != 9
BEGIN
    RAISERROR('El número de teléfono debe tener 9 dígitos.', 16, 1);
    RETURN;
END

-- Si los datos son válidos, se pueden insertar
INSERT INTO mi_tabla (edad, telefono) VALUES (@edad, @telefono);
```

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

```
-- Crear una especificación de auditoría para auditar eventos de acceso
CREATE SERVER AUDIT mi_auditoria
TO FILE (FILEPATH = 'C:\ruta\de\auditoria\')
WITH (ON_FAILURE = CONTINUE);

-- Crear una especificación de auditoría de base de datos
CREATE DATABASE AUDIT SPECIFICATION mi_auditoria_bd
FOR SERVER AUDIT mi_auditoria
ADD (SUCCESSFUL_LOGIN_GROUP),
ADD (FAILED_LOGIN_GROUP),
ADD (DATABASE_OBJECT_PERMISSION_CHANGE_GROUP)
WITH (STATE = ON);
```

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

```
-- Configuración de memoria
EXEC sp_configure 'max server memory (MB)', 4096; -- Asigna 4 GB de memoria
RECONFIGURE;

-- Configuración de disco (cambiar los archivos de datos y log)
ALTER DATABASE mi_base_de_datos
MODIFY FILE (NAME = mi_base_de_datos_data, FILENAME = 'C:\nuevo_ubicacion\mi_base_de_datos.mdf');
ALTER DATABASE mi_base_de_datos
MODIFY FILE (NAME = mi_base_de_datos_log, FILENAME = 'C:\nuevo_ubicacion\mi_base_de_datos_log.ldf');
```

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



```
-- Generación de una copia de seguridad de la base de datos
BACKUP DATABASE mi_base_de_datos
TO DISK = 'C:\ruta\mi_base_de_datos.bak'
WITH FORMAT, MEDIANAME = 'BackupMedio', NAME = 'Backup Completo';
```

Realice un script para programar backups automatizados de su base de datos

```
-- Crear un trabajo en SQL Server Agent para respaldar la base de datos
EXEC msdb.dbo.sp_add_job
    @job_name = 'Backup Diario';

EXEC msdb.dbo.sp_add_jobstep
    @job_name = 'Backup Diario',
    @step_name = 'Backup Completo',
    @subsystem = 'TSQL',
    @command = 'BACKUP DATABASE mi_base_de_datos TO DISK = ''C:\ruta\mi_base_de_datos.bak'';',
    @on_success_action = 1,
    @on_fail_action = 2;

EXEC msdb.dbo.sp_add_schedule
    @schedule_name = 'Backup Diario',
    @enabled = 1,
    @freq_type = 4, -- Frecuencia diaria
    @freq_interval = 1, -- Todos los días
    @active_start_time = 020000; -- 2:00 AM

EXEC msdb.dbo.sp_attach_schedule
    @job_name = 'Backup Diario',
    @schedule_name = 'Backup Diario';
```

Utilizando un script, genere la restauración de la base de datos

```
-- Restaurar la base de datos desde un archivo de respaldo
RESTORE DATABASE mi_base_de_datos
FROM DISK = 'C:\ruta\mi_base_de_datos.bak'
WITH REPLACE;
```

Utilizando un script, cree un espejo de la base de datos

```
-- Iniciar el servicio de espejo de base de datos
ALTER DATABASE mi_base_de_datos
SET PARTNER = 'TCP://ServidorEspejo:5022';
```

Utilizando un script, para enviar datos a la base de datos espejo creada

```
-- Iniciar el servicio de espejo de base de datos
ALTER DATABASE mi_base_de_datos
SET PARTNER = 'TCP://ServidorEspejo:5022';

-- Realizar operaciones en la base de datos espejo (los datos se sincronizarán automáticamente)
INSERT INTO mi_base_de_datos (columna1, columna2)
VALUES ('valor1', 'valor2');
```

Utilizando un script, de permiso a un usuario por un determinado tiempo

```

-- Crear un trabajo para otorgar permisos y luego revocarlos después de un periodo
GRANT SELECT ON mi_tabla TO CarlosPerez;

-- Crear un job para revocar permisos después de 1 hora
EXEC msdb.dbo.sp_add_job
    @job_name = 'Revocar Permisos CarlosPerez';

EXEC msdb.dbo.sp_add_jobstep
    @job_name = 'Revocar Permisos CarlosPerez',
    @step_name = 'Revocar Permisos',
    @subsystem = 'TSQL',
    @command = 'REVOKE SELECT ON mi_tabla TO CarlosPerez;',
    @on_success_action = 1,
    @on_fail_action = 2;

EXEC msdb.dbo.sp_add_schedule
    @schedule_name = 'Revocar Permisos CarlosPerez',
    @enabled = 1,
    @freq_type = 1, -- Frecuencia única
    @active_start_time = 060000; -- Ejecución después de 1 hora

EXEC msdb.dbo.sp_attach_schedule
    @job_name = 'Revocar Permisos CarlosPerez',
    @schedule_name = 'Revocar Permisos CarlosPerez';

```

Utilizando un script, realice la replicación de bases de datos

```

-- Configurar el servidor como publicador
EXEC sp_adddistributor @distributor = 'mi_servidor', @password = 'contraseña';

-- Configurar la publicación
EXEC sp_addpublication
    @publication = 'PublicaciónMiBase',
    @description = 'Publicación de la base de datos',
    @publication_type = 0, -- Publicación transaccional
    @sync_method = 'native',
    @retention = 60,
    @allow_push = true,
    @allow_pull = true;

-- Agregar la base de datos como publicadora
EXEC sp_addarticle
    @publication = 'PublicaciónMiBase',
    @article = 'mi_base_de_datos',
    @source_table = 'mi_tabla',
    @type = 'logbased';

```

## 29) Always On Availability Groups

**Always On Availability Groups (AG)** es una característica de alta disponibilidad y recuperación ante desastres disponible en SQL Server (desde la versión 2012 en adelante, en ediciones Enterprise) que permite a las bases de datos de una instancia de SQL Server estar disponibles de manera continua, incluso en situaciones de fallos. Su propósito principal es proporcionar alta disponibilidad y protección de datos en entornos de misión crítica.

### Características clave de Always On Availability Groups:

- **Grupos de Disponibilidad:** Permite agrupar varias bases de datos en un solo "grupo de disponibilidad". Esto significa que puedes manejar varias bases de datos a la vez como una unidad de alta disponibilidad.

- **Réplicas Activas:** Una característica fundamental de Always On AG es que permite tener **réplicas secundarias** (hasta 8 réplicas, con una réplica primaria) de una base de datos en servidores diferentes. Estas réplicas pueden estar **en modo de solo lectura** o en modo de **sincronización** para proporcionar redundancia y balanceo de carga.
- **Sincronización:** Las réplicas secundarias están sincronizadas con la base de datos primaria. La sincronización puede ser de dos tipos:
  - **Sincrónica (Synchronous):** La base de datos primaria y la réplica secundaria deben estar sincronizadas antes de que cualquier transacción se confirme.
  - **Asíncrona (Asynchronous):** Las transacciones pueden ser confirmadas en la base de datos primaria antes de que lleguen a las réplicas secundarias. Esto es útil para entornos con baja latencia, como una réplica geográficamente distante.
- **Failover Automático:** En caso de que la réplica primaria falle, Always On AG puede realizar un **failover automático** a una réplica secundaria configurada para tomar el control, minimizando el tiempo de inactividad.
- **Transparente para el usuario:** El cliente puede conectarse al grupo de disponibilidad utilizando un **listener**. Este es un punto de conexión virtual que siempre redirige las conexiones a la réplica activa disponible.
- **Alta disponibilidad:** Proporciona alta disponibilidad mediante la replicación continua de datos a través de una red entre servidores, asegurando que, incluso en caso de fallo de un servidor, las bases de datos permanezcan accesibles a través de las réplicas secundarias.

#### Escenarios de uso:

- **Entornos de misión crítica:** Empresas que no pueden permitirse un tiempo de inactividad significativo, como los servicios financieros o las plataformas de comercio electrónico.
- **Recuperación ante desastres:** Permite recuperar los datos de las réplicas secundarias en caso de fallo de la base de datos primaria.

#### Ventajas:

- Alta disponibilidad y protección de datos en tiempo real.
- Failover automático o manual.
- Soporta bases de datos de solo lectura en réplicas secundarias, lo que ayuda a distribuir la carga de trabajo.
- Gran escalabilidad al agregar más réplicas.

## 30) Log Shipping

**Log Shipping** es una técnica de recuperación ante desastres que permite mantener una copia secundaria de una base de datos, replicando los registros de transacciones (log de

transacciones) de una base de datos primaria a una base de datos secundaria en un servidor diferente. Es una solución más sencilla que **Always On Availability Groups** y se usa comúnmente en escenarios donde no es necesaria la alta disponibilidad en tiempo real, pero aún se quiere protección ante desastres.

### Características clave de Log Shipping:

- **Base de Datos Primaria y Secundaria:** Log Shipping consiste en un par de bases de datos: una **primaria** y una o más **secundarias**. La base de datos primaria es la que recibe las actualizaciones de los usuarios, mientras que la secundaria sirve como una copia de respaldo.
- **Transacción Log Backup:** Log Shipping realiza un proceso de respaldo del log de transacciones (transaction log) de la base de datos primaria. Estos archivos de log contienen todas las transacciones realizadas en la base de datos, permitiendo que se reproduzcan en una base de datos secundaria.
- **Copia y Restauración:**
  1. **Backup del log de transacciones:** El primer paso es hacer un respaldo de los logs de transacciones desde la base de datos primaria.
  2. **Copia:** Luego, estos archivos de log son copiados al servidor secundario.
  3. **Restauración en la base de datos secundaria:** Finalmente, los registros de log copiados se restauran en la base de datos secundaria. Este proceso puede hacerse en modo **NORECOVERY** para que la base de datos secundaria no se quede en un estado recuperable y continúe recibiendo los log de transacciones.
- **Sincronización periódica:** A diferencia de Always On, Log Shipping no mantiene las réplicas sincronizadas en tiempo real. El proceso de backup, copia y restauración es un proceso **periódico** que se ejecuta en intervalos programados (por ejemplo, cada 15 minutos, cada hora).
- **Failover Manual:** Si ocurre un fallo en la base de datos primaria, el failover a la secundaria debe ser realizado manualmente. La base de datos secundaria debe ser restaurada en modo **RECOVERY** para que se vuelva accesible, pero no puede haber transacciones pendientes en los archivos de log.

### Escenarios de uso:

- **Recuperación ante desastres:** Log Shipping es una solución ideal para empresas que desean tener una copia de seguridad física de sus bases de datos en otro servidor para su recuperación rápida, pero que no requieren alta disponibilidad en tiempo real.
- **Entornos de menor carga:** Log Shipping es adecuado para entornos donde el tiempo de inactividad tolerable no es crítico, y no se requiere un failover automático.

### Ventajas:

- Relativamente sencillo de configurar y administrar.
- Menor costo comparado con soluciones como Always On.
- Proporciona una copia de seguridad en un servidor remoto.
- Adecuado para recuperación ante desastres.

### Desventajas:

- No es una solución de alta disponibilidad en tiempo real, ya que las réplicas no están siempre sincronizadas con la base de datos primaria.
  - El failover debe ser manual.
  - No soporta cargas de trabajo de solo lectura como las réplicas de Always On.
- 

### Comparativa entre Always On y Log Shipping:

Característica	Always On Availability Groups	Log Shipping
<b>Alta disponibilidad en tiempo real</b>	Sí, con failover automático	No, el failover es manual
<b>Sincronización de datos</b>	Sincrónica o asíncrona (dependiendo de la configuración)	No en tiempo real; hay un desfase entre el backup y la restauración
<b>Escalabilidad</b>	Soporta múltiples réplicas secundarias	No soporta muchas réplicas
<b>Costo</b>	Requiere SQL Server Enterprise Edition	Más económico, se puede usar en ediciones Standard
<b>Uso de bases de datos de solo lectura</b>	Sí, en réplicas secundarias	No disponible de forma directa
<b>Complejidad de configuración</b>	Más complejo debido a las funcionalidades avanzadas	Relativamente sencillo de configurar

Ambas son soluciones valiosas dependiendo de las necesidades de alta disponibilidad, recuperación ante desastres y la carga de trabajo. **Always On Availability Groups** es más adecuado para entornos de alta disponibilidad en tiempo real, mientras que **Log Shipping** es una solución más sencilla para la protección de datos y recuperación ante desastres.