Seguridad y control de acceso

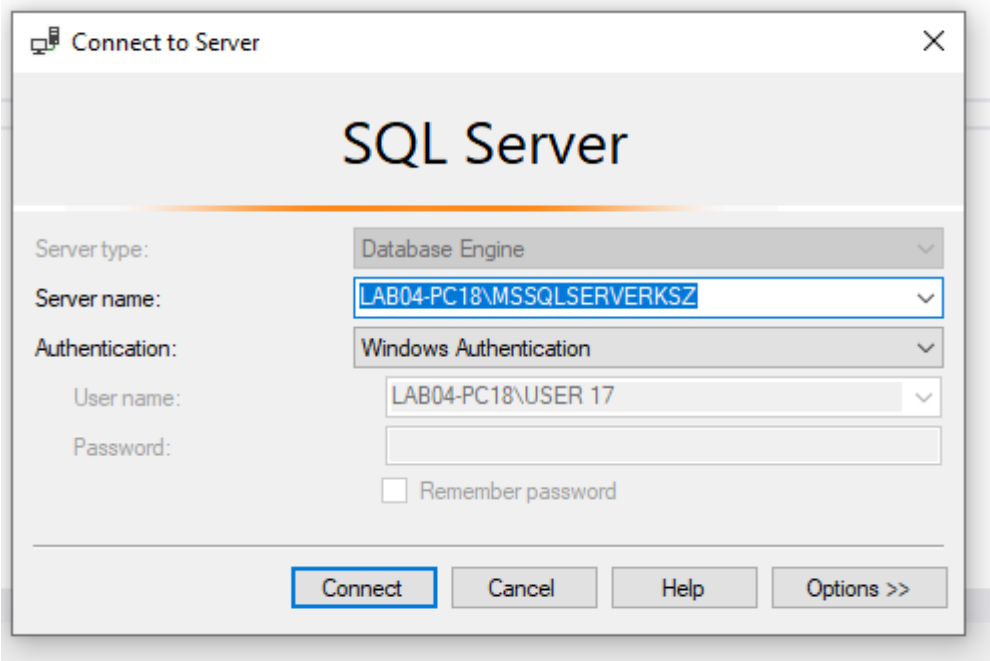
Objetivo: Implementar políticas de seguridad robustas y administración de roles.
Temas:

1. Autenticación SQL y Windows: diferencias y prácticas seguras.

Definición:

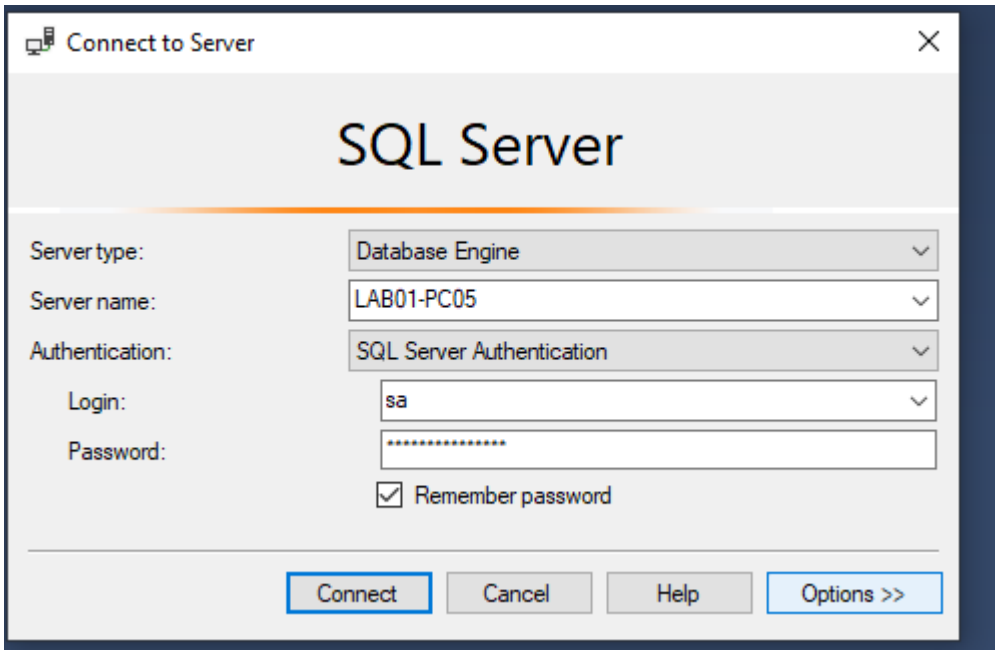
SQL Server ofrece dos modos de autenticación:

1. **Autenticación de Windows:**
   Utiliza las credenciales del sistema operativo (Active Directory o usuarios locales). Es el método **más seguro**, porque se apoya en la infraestructura de seguridad de Windows (políticas de contraseñas, bloqueo de cuentas, Kerberos, etc.).



   a. Ventajas:
      i. No se transmiten contraseñas en texto plano.
      ii. Integración con políticas de dominio.
      iii. Administración centralizada.
   b. Desventajas:
      i. Dependencia del Entorno de Windows (Active Directory)
      ii. Complejidad en Conexiones Externas.
      iii. Sin Cuenta de Emergencia Integrada

2. **Autenticación de SQL Server:**
   Las credenciales (usuario y contraseña) se almacenan dentro de SQL Server. Se usa cuando no hay integración con Active Directory o para accesos externos.



   a. Ventajas:
      i. Cuenta de "Super-Administrador" (sa).
      ii. Acceso a Través de Internet o Redes No Confiables:
      iii. Flexibilidad en el Desarrollo.
   b. Desventajas:
      i. Se deben manejar contraseñas dentro de SQL Server.
      ii. Menor seguridad si no se aplican políticas robustas.

Prácticas seguras:

- Preferir **Autenticación de Windows** siempre que sea posible.
- Deshabilitar la cuenta `sa` o cambiarle el nombre.
- Implementar **políticas de contraseñas fuertes** y expiración.
- Utilizar **roles** en lugar de asignar permisos directos a usuarios.

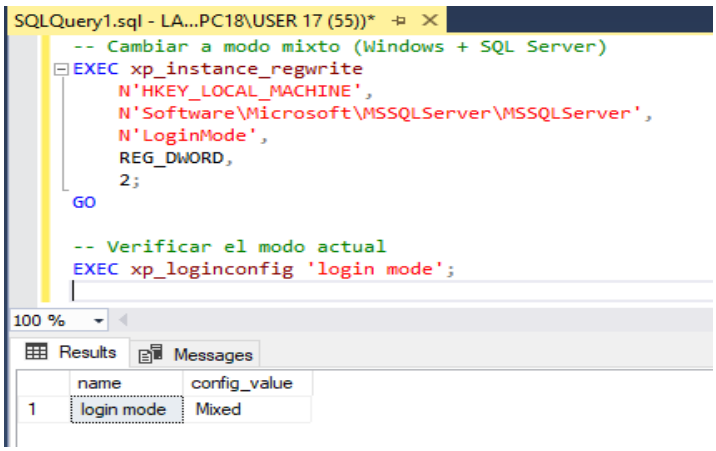
Diferencias:

Característica	Autenticación de Windows	Autenticación de SQL Server

Seguridad	**Mayor**. Se beneficia de Kerberos, políticas de dominio, bloqueo de cuentas.	**Menor** por defecto. Depende de las políticas internas de SQL Server.
Administración	**Centralizada**. Se maneja a nivel de dominio/SO.	**Descentralizada**. Se gestiona dentro de SQL Server.
Transmisión de Contraseña	**No** se transmite en texto plano. Solo se transmiten tokens.	**Puede** transmitirse si no se usa cifrado (aunque lo ideal es usar cifrado TLS/SSL).
Uso Típico	Ambientes corporativos con Active Directory, conexiones internas y locales.	Aplicaciones que no se integran con Active Directory, accesos externos, entornos de trabajo en grupo.
Soporte de Contraseña	Impuesto por el SO/Dominio (fuerza, expiración, historial).	Implementado y gestionado **dentro** de SQL Server.

Script para cambiar el modo de autenticación:

```
-- Cambiar a modo mixto (Windows + SQL Server)
EXEC xp_instance_regwrite N'HKEY_LOCAL_MACHINE',
N'Software\Microsoft\MSSQLServer\MSSQLServer', N'LoginMode', REG_DWORD, 2;
GO
-- Verificar el modo actual
EXEC xp_loginconfig 'login mode';
```



2. Cuentas de servicio y configuración del servidor.

Definición y Fundamentación

Las cuentas de servicio son identidades bajo las cuales se ejecutan los servicios de SQL Server (Motor de Base de Datos, Agente SQL Server, Integration Services (SSIS), Reporting Services (SSRS), Analysis Services (SSAS), etc.).

Importancia:

1. **Seguridad y Aislamiento:** Al ejecutar cada servicio con una cuenta separada y con privilegios mínimos, se logra el **aislamiento de fallas**. Si un atacante compromete un servicio (por ejemplo, el Agente), solo obtendrá los permisos de esa cuenta, limitando el daño potencial a otros servicios o al sistema operativo.
2. **Principio de Mínimo Privilegio (PoLP):** La cuenta solo debe tener los permisos necesarios para realizar su trabajo (acceso a archivos de base de datos, logs, red). Esto reduce drásticamente la superficie de ataque.
3. **Auditoría Clara:** Permite una auditoría sencilla, ya que los registros de eventos del sistema operativo y de red identifican claramente qué servicio está realizando una acción específica.

Tipos de Cuentas de Servicio

SQL Server ofrece varios tipos de cuentas. La elección se basa en el nivel de seguridad y la necesidad de acceso a recursos de red.

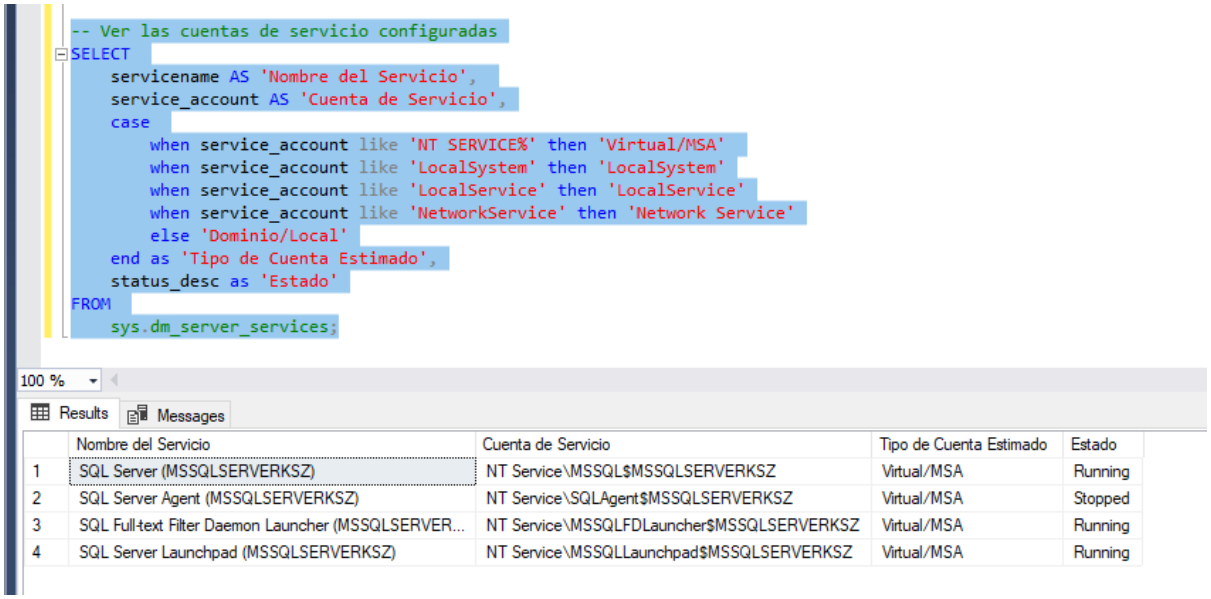
Tipo de Cuenta	Descripción	Uso Recomendado
Cuenta de Usuario de Dominio	Cuentas estándar de Active Directory.	**Recomendado y Preferido.** Proporciona el mejor equilibrio entre seguridad y funcionalidad.
Cuentas de Servicio Administradas de Grupo (gMSA)	Cuentas de dominio que son gestionadas automáticamente por Windows (cambio de contraseña, etc.).	**Más Seguro.** Ideal para entornos de Active Directory. Ofrece gestión de contraseñas automatizada, reduciendo la carga administrativa.
Cuentas de Servicio Administradas (MSA)	Similar a gMSA, pero solo se puede asignar a un único servidor.	Para servicios que solo residen en un servidor y se desea la gestión automatizada de la contraseña.
Sistema Local (`LocalSystem`)	Cuenta de alto privilegio dentro del servidor.	**Evitar a toda costa.** Tiene permisos excesivos, permitiendo el control total del SO. **Violación del PoLP.**

Servicio de Red (`Network Service`)	Permite el acceso a recursos de red utilizando las credenciales del equipo.	**Evitar.** Aunque tiene menos privilegios que `LocalSystem`, no es la práctica recomendada para el motor de SQL Server.
Servicio Local (`Local Service`)	Cuenta con privilegios mínimos en el equipo local.	**No recomendado.** No tiene acceso a recursos de red y es demasiado restrictivo para el motor de SQL Server.

Script para Ver Cuentas de Servicio:

-- Ver las cuentas de servicio configuradas

SELECT
servicename AS 'Nombre del Servicio', service_account AS 'Cuenta de Servicio', case when service_account like 'NT SERVICE%' then 'Virtual/MSA' when service_account like 'LocalSystem' then 'LocalSystem' when service_account like 'LocalService' then 'LocalService' when service_account like 'NetworkService' then 'Network Service' else 'Dominio/Local' end as 'Tipo de Cuenta Estimado', status_desc as 'Estado'
FROM
sys.dm_server_services;



3. Creación de roles fijos y personalizados.

Definición

Un **rol** en SQL Server es una agrupación lógica de permisos. Permite administrar la seguridad de forma más eficiente, asignando permisos a roles y luego agregando usuarios a dichos roles (en lugar de dar permisos individualmente).

SQL Server maneja **dos niveles principales de roles**:

- **Roles de servidor (server roles):** aplican a nivel de instancia.
- **Roles de base de datos (database roles):** aplican dentro de una base específica.

Tipos de Roles Fijos del Servidor

Estos roles son predeterminados por SQL Server. Ejemplos y sus funciones:

Rol de Servidor	Descripción
`sysadmin`	Control total sobre la instancia de SQL Server.
`serveradmin`	Configura opciones del servidor y controla el servicio.
`securityadmin`	Administra inicios de sesión, contraseñas y permisos de servidor.
`dbcreator`	Crea, altera, borra y restaura bases de datos.
`diskadmin`	Administra archivos de disco.
`processadmin`	Maneja procesos en ejecución.
`setupadmin`	Administra servidores vinculados.
`bulkadmin`	Permite ejecutar operaciones de importación masiva (BULK INSERT).

Roles Fijos de Base de Datos

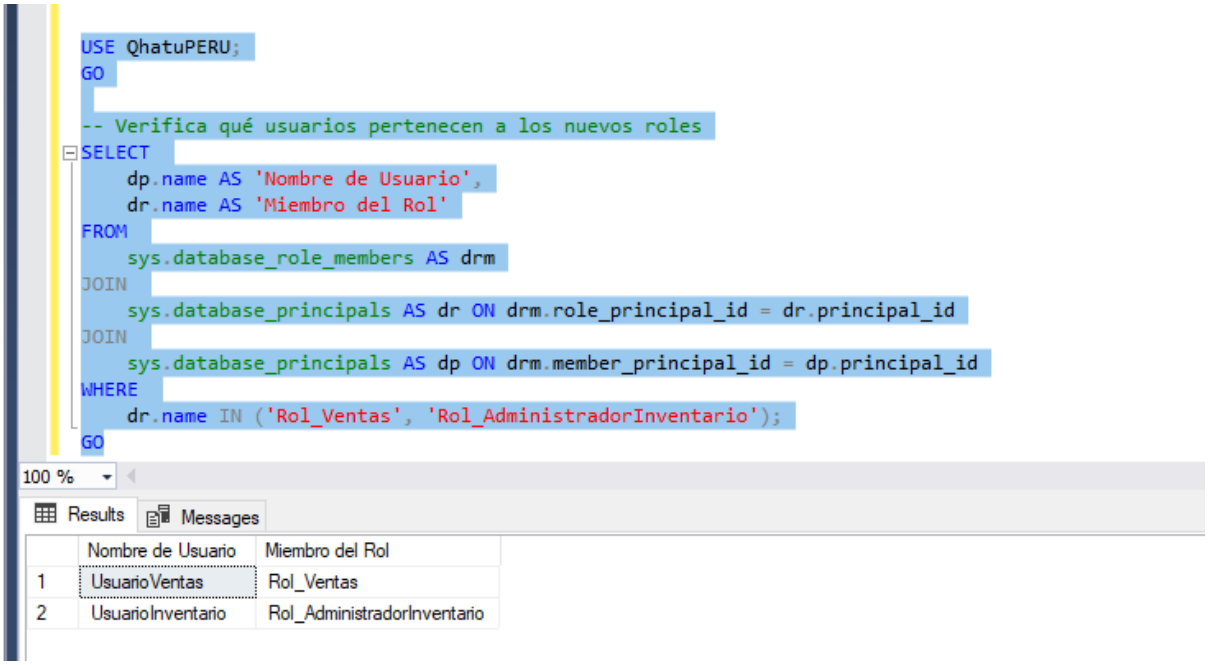
Aplican dentro de cada base de datos, como `QhatuPERU`.

Rol de Base de Datos	Descripción
`db_owner`	Control total sobre la base de datos.
`db_securityadmin`	Administra roles y permisos de la base.
`db_accessadmin`	Administra acceso de usuarios.
`db_backupoperator`	Permite hacer copias de seguridad.
`db_datareader`	Permite seleccionar (SELECT) sobre todas las tablas.
`db_datawriter`	Permite insertar, actualizar y borrar (INSERT, UPDATE, DELETE).

db_ddladmin	Puede ejecutar comandos DDL (CREATE, ALTER, DROP).
db_denydatareader	Deniega SELECT.
db_denydatawriter	Deniega INSERT, UPDATE, DELETE.

SCRIPT Verificación de Miembros de Roles (Comprueba la Asignación)

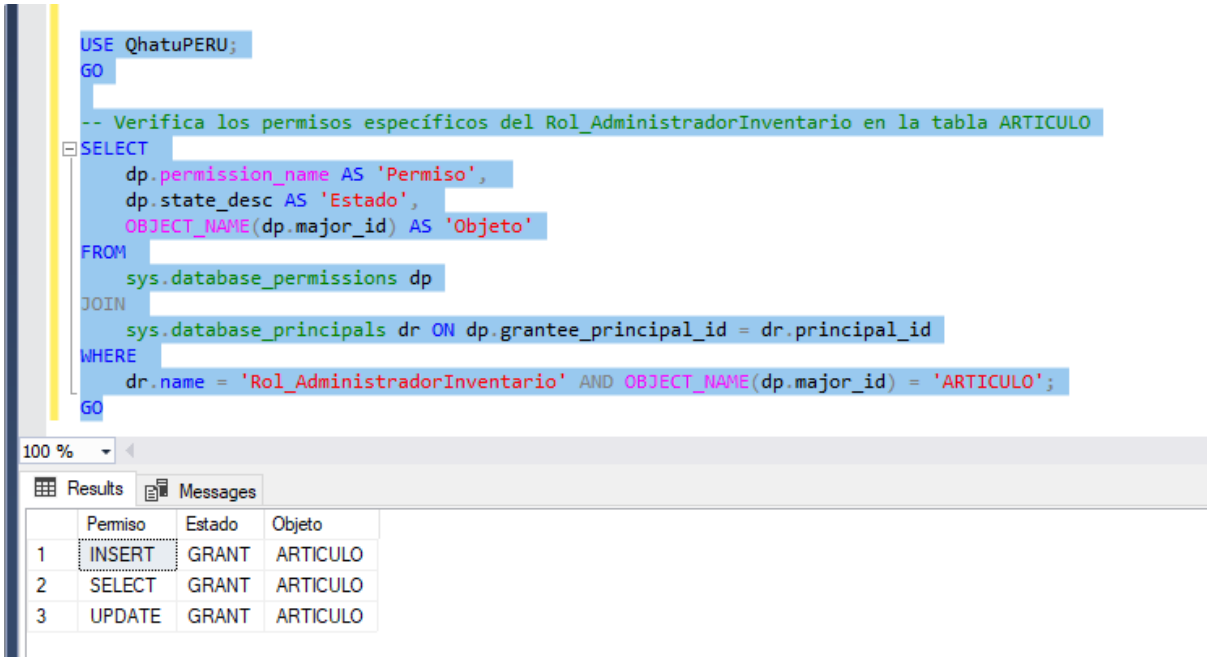
-- Verifica qué usuarios pertenecen a los nuevos roles
SELECT dp.name AS 'Nombre de Usuario', dr.name AS 'Miembro del Rol' FROM sys.database_role_members AS drm JOIN sys.database_principals AS dr ON drm.role_principal_id = dr.principal_id JOIN sys.database_principals AS dp ON drm.member_principal_id = dp.principal_id
WHERE dr.name IN ('Rol_Ventas', 'Rol_AdministradorInventario');
GO



Verificación de Permisos Otorgados (Comprueba las Sentencias GRANT)

USE QhatuPERU; GO
-- Verifica los permisos específicos del Rol_AdministradorInventario en la tabla ARTICULO
SELECT dp.permission_name AS 'Permiso', dp.state_desc AS 'Estado', OBJECT_NAME(dp.major_id) AS 'Objeto' FROM sys.database_permissions

dp JOIN sys.database_principals dr ON dp.grantee_principal_id = dr.principal_id
WHERE dr.name = 'Rol_AdministradorInventario' AND OBJECT_NAME(dp.major_id) = 'ARTICULO';
GO



```
USE QhatuPERU;
GO

-- Verifica los permisos específicos del Rol_AdministradorInventario en la tabla ARTICULO
SELECT
    dp.permission_name AS 'Permiso',
    dp.state_desc AS 'Estado',
    OBJECT_NAME(dp.major_id) AS 'Objeto'
FROM
    sys.database_permissions dp
JOIN
    sys.database_principals dr ON dp.grantee_principal_id = dr.principal_id
WHERE
    dr.name = 'Rol_AdministradorInventario' AND OBJECT_NAME(dp.major_id) = 'ARTICULO';
GO
```

100 %

Results | Messages

	Permiso	Estado	Objeto
1	INSERT	GRANT	ARTICULO
2	SELECT	GRANT	ARTICULO
3	UPDATE	GRANT	ARTICULO

4. Control de acceso mediante GRANT, DENY y REVOKE.

El **control de acceso** define quién puede hacer qué sobre los objetos de la base de datos.

SQL Server utiliza tres comandos clave:

Comando	Función
GRANT	Otorga un permiso.

DENY	Niega un permiso (tiene prioridad sobre GRANT).
REVOKE	Elimina un permiso otorgado o denegado previamente.

Tipos de permisos comunes:

Tipo de Permiso	Descripción	Aplicable a
SELECT	Lectura de datos	Tablas, vistas
INSERT	Inserción de datos	Tablas
UPDATE	Modificación	Tablas
DELETE	Eliminación	Tablas
ALTER	Cambios de estructura	Esquemas, tablas
CONTROL	Control total	Cualquier objeto
EXECUTE	Ejecución	Procedimientos, funciones

Script de control de acceso:

```
--GRANT Otorgar permisos
 GRANT SELECT, INSERT ON dbo.PROVEEDOR TO UsuarioInventario;
--DENY Negar permisos
-- Negar acceso de eliminación a UsuarioInventario
DENY DELETE ON dbo.ARTICULO TO UsuarioInventario;
--REVOKE Quitar permisos
-- Quitar permisos de inserción en PROVEEDOR
 REVOKE INSERT ON dbo.PROVEEDOR FROM UsuarioInventario;
--Ver permisos actuales
 EXEC sp_helprotect NULL, 'UsuarioInventario';
```

SQLQuery4.sql - LA...PC18\USER 17 (79))* | SQLQuery2.sql - LA...PC18\USER 17 (76))* | SQLQuery1.s

```
USE QhatuPERU;
--GRANT - Otorgar permisos
GRANT SELECT, INSERT ON dbo.PROVEEDOR TO UsuarioInventario;

--DENY - Negar permisos
-- Negar acceso de eliminación a UsuarioInventario
DENY DELETE ON dbo.ARTICULO TO UsuarioInventario;

--REVOKE - Quitar permisos
-- Quitar permisos de inserción en PROVEEDOR
REVOKE INSERT ON dbo.PROVEEDOR FROM UsuarioInventario;

--Ver permisos actuales
EXEC sp_helprotect NULL, 'UsuarioInventario';
```

100 %

Results | Messages

	Owner	Object	Grantee	Grantor	ProtectType	Action	Column
1	dbo	ARTICULO	UsuarioInventario	dbo	Deny	Delete	.
2	dbo	PROVEEDOR	UsuarioInventario	dbo	Grant	Select	(All+New)
3	.	.	UsuarioInventario	dbo	Grant	CONNECT	.

5. Cifrado y protección de datos (TDE, Always Encrypted).

Definición

El cifrado es una capa de defensa crítica en profundidad (Defense-in-Depth) . Su objetivo principal es garantizar que, incluso si un atacante obtiene acceso a los datos, estos sean ilegibles e inutilizables sin las claves de descifrado. SQL Server ofrece herramientas para cifrar datos **en reposo** (almacenados en el disco) y **en uso** (cuando son procesados).

Mecanismo de Cifrado	Nivel de Protección	¿Quién tiene la clave?
TDE (Transparente)	Archivos Físicos (MDF, LDF, Backups)	SQL Server
Always Encrypted	Columnas Específicas de la Tabla	La Aplicación Cliente (Control del Usuario)

A. Transparent Data Encryption (TDE)

Fundamentación

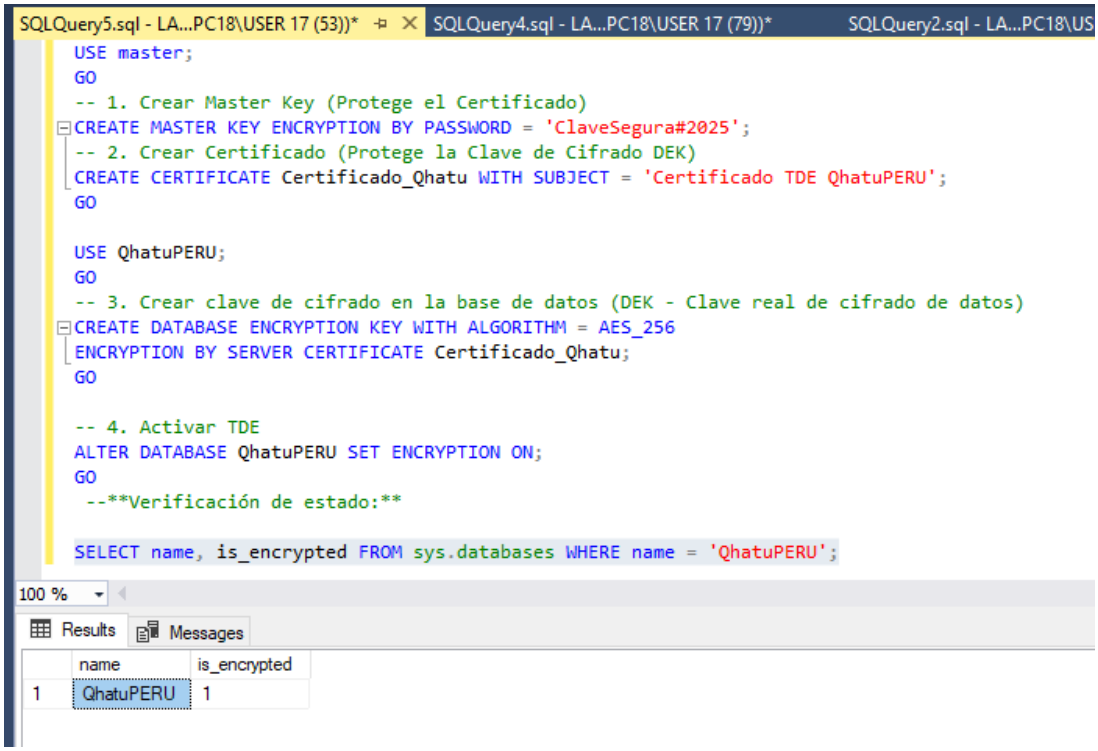
TDE aborda la amenaza del **robo físico de datos**. Su propósito es proteger los archivos de la base de datos (datos `.mdf`, logs `.ldf` y copias de seguridad) si un atacante los roba, por ejemplo, llevándose un disco duro o una cinta de *backup*.

Características Clave:

- **Cifrado a Nivel de Almacenamiento:** El cifrado ocurre antes de que los datos se escriban en el disco y el descifrado ocurre al leerse en la memoria.
- **Transparente para Aplicaciones:** Las aplicaciones cliente y los usuarios no necesitan modificar su código ni manejar claves, de ahí el término "transparente".
- **Jerarquía de Claves:** Utiliza una jerarquía robusta: **Service Master Key $\rightarrow$ Database Master Key $\rightarrow$ Certificado de Servidor $\rightarrow$ Database Encryption Key (DEK)** (Clave de Cifrado de Base de Datos). La DEK es la clave simétrica real que cifra la base de datos.
- **Protección de Datos en Reposo:** Su principal función es inutilizar los archivos si son extraídos fuera del entorno del servidor SQL.

Uso:

- Se usa para proteger los datos en reposo (data at rest).
- Ideal en entornos donde se almacenan datos sensibles: finanzas, salud, universidades, etc.



6. Auditoría y monitoreo de eventos con SQL Server Audit

Diferencia entre Auditoría y Monitoreo

Concepto	Descripción	Ejemplo
Auditoría	Es el **registro histórico de eventos** que ocurren en SQL Server. Permite **rastrear actividades** para saber qué pasó.	Saber quién ejecutó un DELETE o ALTER TABLE.

Monitoreo	Es la **observación continua del comportamiento del sistema** en tiempo real o casi real, para detectar anomalías o problemas.	Detectar inicios de sesión fallidos repetidos, lentitud en consultas o cambios inesperados.

En otras palabras:

- **Auditoría = Registro (log)**
- **Monitoreo = Supervisión (alertas y análisis en vivo)**

¿Qué es SQL Server Audit?

La **auditoría** en SQL Server es una herramienta que permite **registrar y monitorear** lo que ocurre dentro del servidor o una base de datos.
Su propósito es saber **quién hizo qué, cuándo y dónde**.

Por ejemplo:

- ¿Quién modificó una tabla?
- ¿Cuándo se eliminó un registro?
- ¿Quién intentó conectarse sin permiso?
- ¿Qué consultas se ejecutaron sobre datos sensibles?

Cómo funciona el monitoreo con SQL Server Audit

Una vez activada la auditoría:

- SQL Server **graba los eventos en un archivo** o en el registro de Windows.
- Estos registros se pueden **consultar o analizar** en SQL Server Management Studio (SSMS) o herramientas externas.
- Se pueden configurar **alertas automáticas** si se detectan ciertos tipos de eventos (usando SQL Agent o Power BI para visualización).

Componentes de SQL Server Audit

Componente	Qué hace	Nivel

Audit (Auditoría del Servidor)	Es el contenedor principal. Define **dónde** se guardarán los registros (archivo, log de Windows, etc.)	Servidor
Server Audit Specification	Define **qué eventos** del servidor auditar (por ejemplo, inicios de sesión, cambios de configuración).	Servidor
Database Audit Specification	Define **qué acciones dentro de una base de datos** auditar (por ejemplo, `SELECT`, `UPDATE`, `DELETE` sobre tablas).	Base de Datos

Configuración paso a paso (ejemplo práctico)

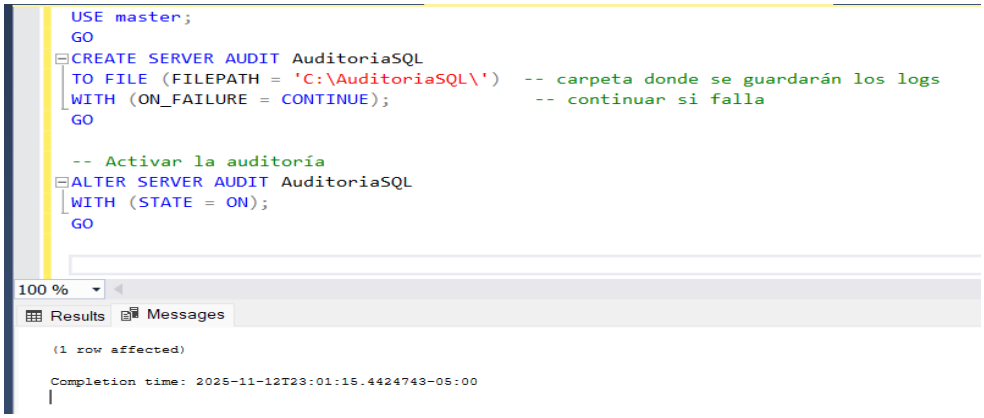
A continuación, un ejemplo que audita todos los **SELECT e INSERT** en la tabla `Alumno` de la base `UniversidadDB`.

1.Crear una Auditoría del Servidor

Esto define **dónde guardarás los registros**.

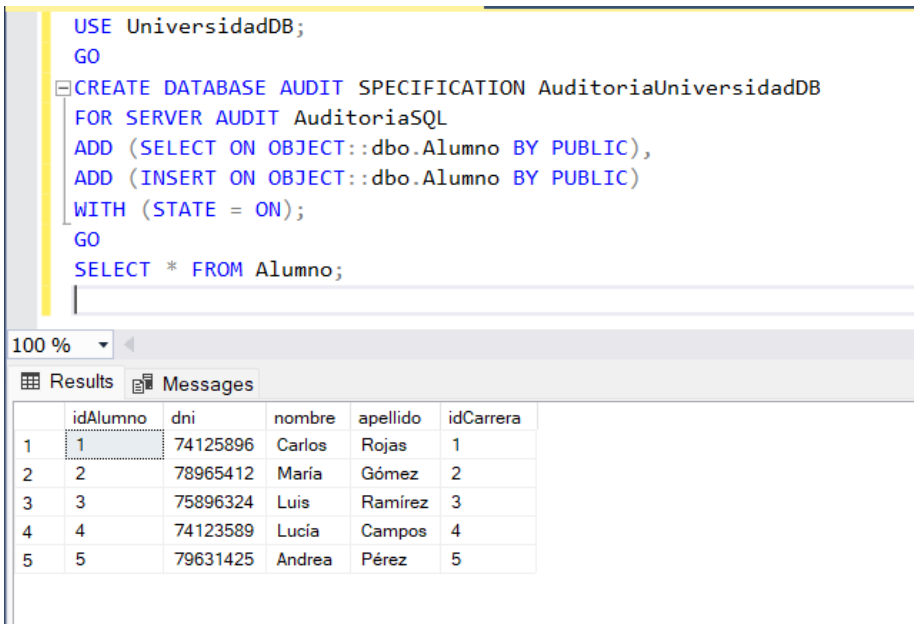
```
USE master;
GO
CREATE SERVER AUDIT AuditoriaSQL
TO FILE (FILEPATH = 'C:\AuditoriaSQL\')  -- carpeta donde se guardarán los logs
WITH (ON_FAILURE = CONTINUE);          -- continuar si falla
GO

-- Activar la auditoría
ALTER SERVER AUDIT AuditoriaSQL
WITH (STATE = ON);
GO
```



2. Crear una Especificación de Auditoría a Nivel de Base de Datos

USE UniversidadDB;
GO
CREATE DATABASE AUDIT SPECIFICATION AuditoriaUniversidadDB
FOR SERVER AUDIT AuditoriaSQL
ADD (SELECT ON OBJECT::dbo.Alumno BY PUBLIC),
ADD (INSERT ON OBJECT::dbo.Alumno BY PUBLIC)
WITH (STATE = ON);
GO

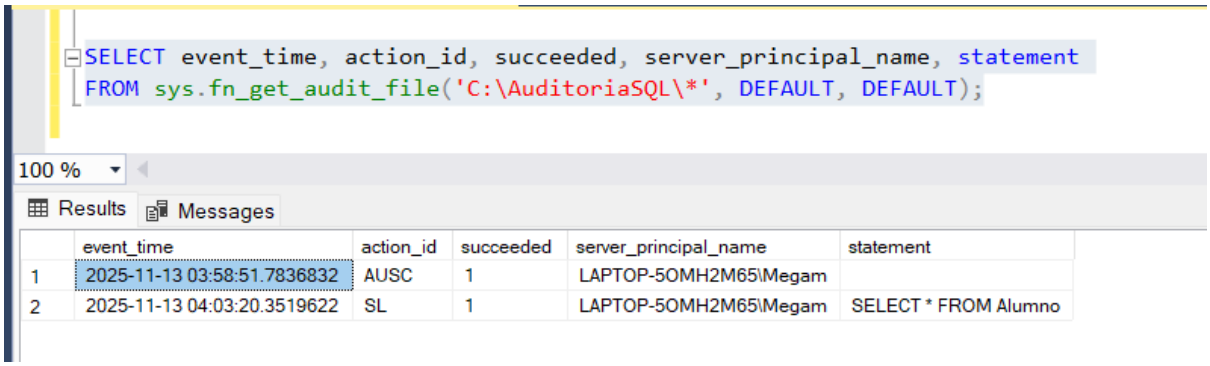


	idAlumno	dni	nombre	apellido	idCarrera
1	1	74125896	Carlos	Rojas	1
2	2	78965412	María	Gómez	2
3	3	75896324	Luis	Ramírez	3
4	4	74123589	Lucía	Campos	4
5	5	79631425	Andrea	Pérez	5

3.Consultar los registros de auditoría

Ahora puedes leer lo que se registró:

SELECT event_time, action_id, succeeded, server_principal_name, statement
FROM sys.fn_get_audit_file('C:\AuditoriaSQL*', DEFAULT, DEFAULT);



Ventajas de usar SQL Server Audit

Ventaja	Descripción
Seguridad y cumplimiento	Permite cumplir con leyes y normativas (por ejemplo, GDPR o ISO).
Detección de incidentes	Puedes descubrir quién cambió datos o intentó acceder sin permiso.
Monitoreo en tiempo real	Permite observar actividad sospechosa en tablas sensibles.
Historial detallado	Guarda cada acción con fecha, usuario y tipo de comando.
Fácil integración	Compatible con alertas de SQL Agent, Power BI o scripts automatizados.