UNIVERSIDAD PERUANA LOS ANDES

FACULTAD INGENIERIA INGENIERÍA DE SISTEMAS Y COMPUTACIÓN

MANUAL

SEGURIDAD Y CONTROL DE ACCESO

ASIGNATURA: BASE DE DATOS II

DOCENTE: FERNÁNDEZ BEJARANO RAUL

ESTUDIANTE: Bonifacio Hilario Erick

CÓDIGO: S01238F

HUANCAYO-2025

1. AUTENTICACIÓN SQL Y WINDOWS

Definición

La autenticación es el proceso mediante el cual SQL Server verifica la identidad del usuario antes de permitirle el acceso.
Existen dos tipos principales:

- **Autenticación de Windows:** utiliza las credenciales del sistema operativo, es más segura porque no almacena contraseñas dentro de SQL Server.
- **Autenticación de SQL Server:** requiere un usuario y contraseña propios del servidor, útil cuando no se maneja un dominio de Windows.

Diferencias

- **Windows:** usa la cuenta del sistema operativo y permite inicio de sesión sin necesidad de volver a escribir contraseña.
- **SQL Server:** requiere usuario y contraseña definidos dentro del motor SQL.
- **Windows:** más segura y fácil de administrar en redes corporativas.
- **SQL Server:** más flexible para conexiones externas o aplicaciones web.

Buenas prácticas

- Utilizar preferentemente autenticación de Windows en entornos empresariales.
- Si se usa autenticación SQL, establecer contraseñas seguras.
- Deshabilitar el inicio de sesión “sa” o cambiarle el nombre para evitar ataques.
- Restringir el número de inicios de sesión con permisos administrativos.

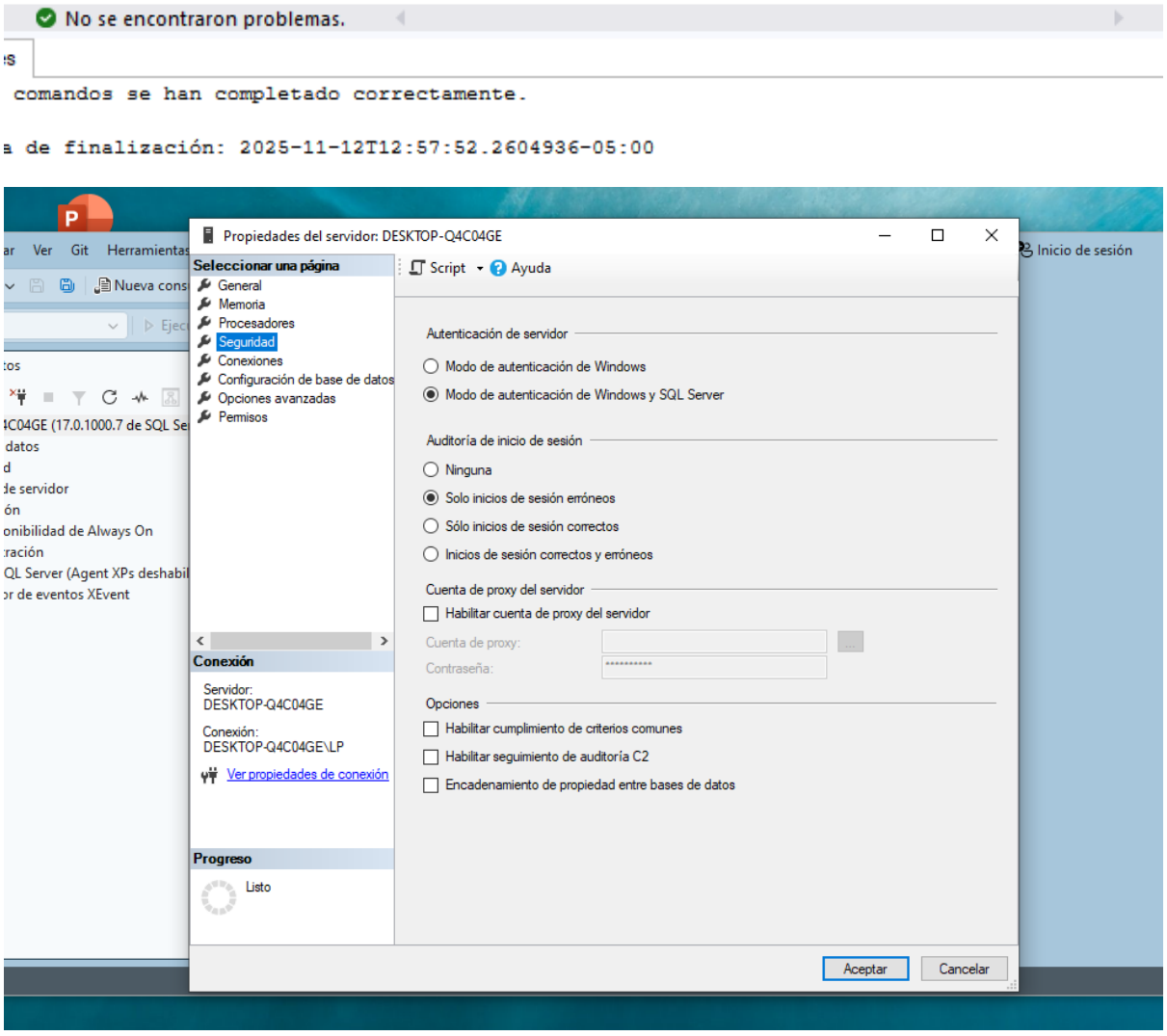
Paso a paso

1. Abrir **SQL Server Management Studio (SSMS)**.

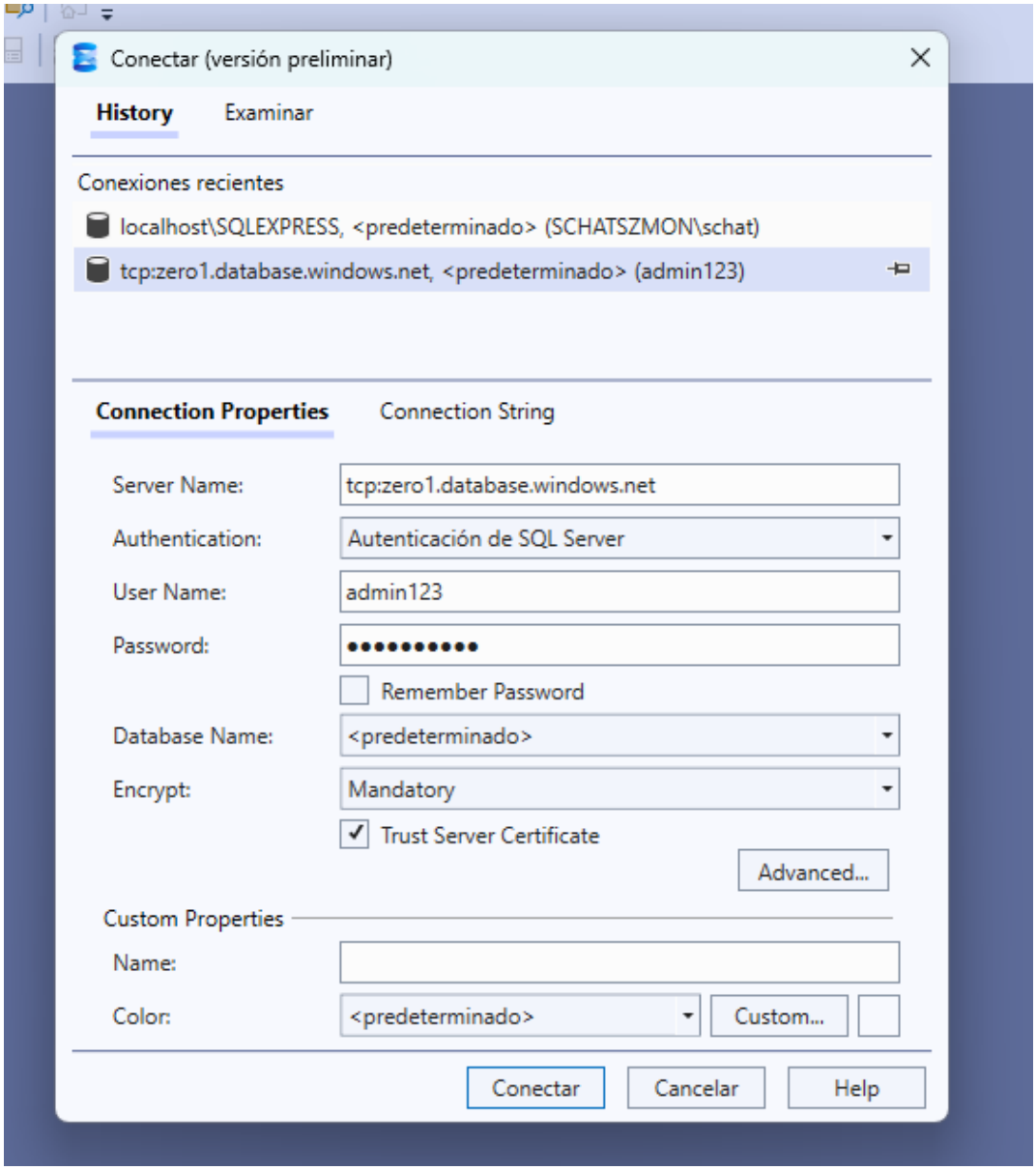
2. Conectarse al servidor con autenticación de Windows.
3. Para cambiar el modo de autenticación:
    a. Clic derecho sobre el servidor → *Propiedades*.
    b. Ir a la pestaña **Seguridad**.
    c. Seleccionar “Autenticación de Windows” o “SQL Server y Windows”.
4. Crear un nuevo inicio de sesión SQL:

```
CREATE LOGIN admin123 WITH PASSWORD = 'Erick2024$',
CHECK_POLICY = ON;
```

```
CREATE LOGIN admin123 WITH PASSWORD = 'Erick2024$', CHECK_POLICY = ON;
```



5. Probar conexión usando ese nuevo usuario.



Explicación

SQL Server autentica usuarios para evitar accesos no autorizados. Configurar correctamente este modo es clave para la seguridad del sistema. El paso a paso permite practicar ambos métodos y entender sus diferencias en escenarios reales.

2. CUENTAS DE SERVICIO Y CONFIGURACIÓN DEL SERVIDOR

Definición

Las cuentas de servicio son las que utiliza SQL Server para ejecutar sus procesos internos en el sistema operativo. Controlan la forma en que el motor accede a archivos, redes y otros servicios.

Buenas prácticas

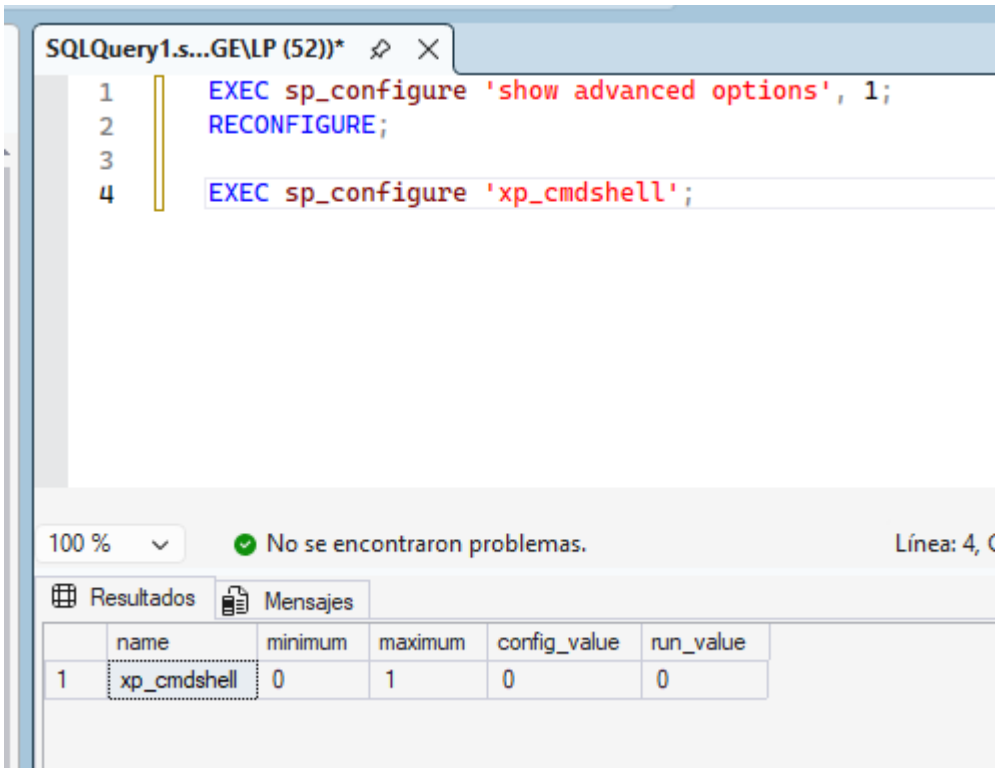
- Usar cuentas de servicio **dedicadas** (no personales).
- No usar cuentas de administrador local.
- Aplicar el principio de **mínimo privilegio**.
- Cambiar contraseñas periódicamente.

Paso a paso

1. Deshabilitar xp_cmdshell

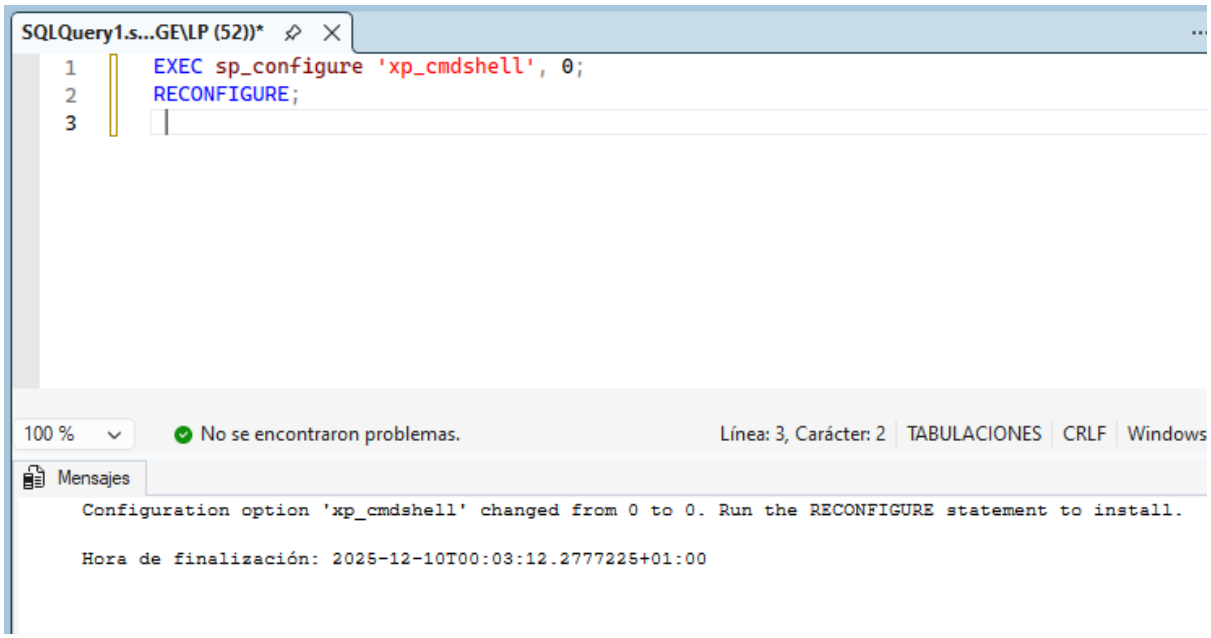
1. Abrir SQL Server Management Studio (SSMS).
2. Ejecutar lo siguiente para verificar su estado:

```
EXEC sp_configure 'show advanced options', 1;
RECONFIGURE;

EXEC sp_configure 'xp_cmdshell';
```



3. Para deshabilitarla, ejecutar:

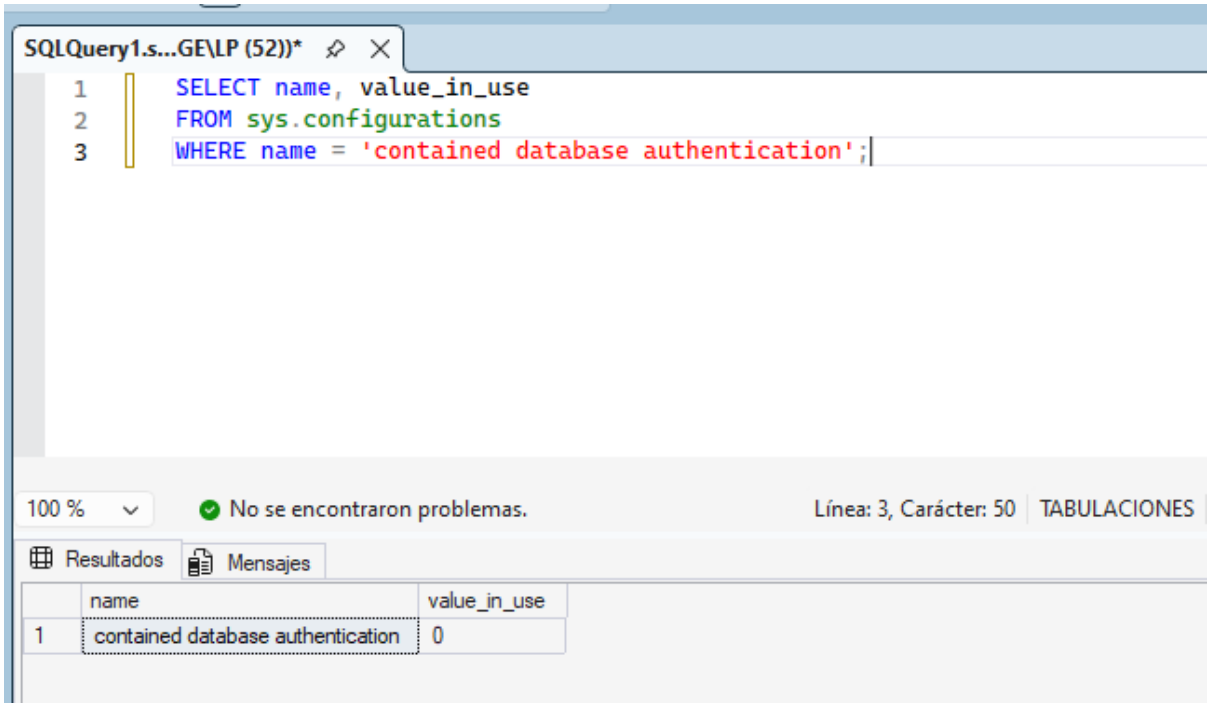
```
EXEC sp_configure 'xp_cmdshell', 0;
RECONFIGURE;
```



2. Revisar Contained Database Authentication

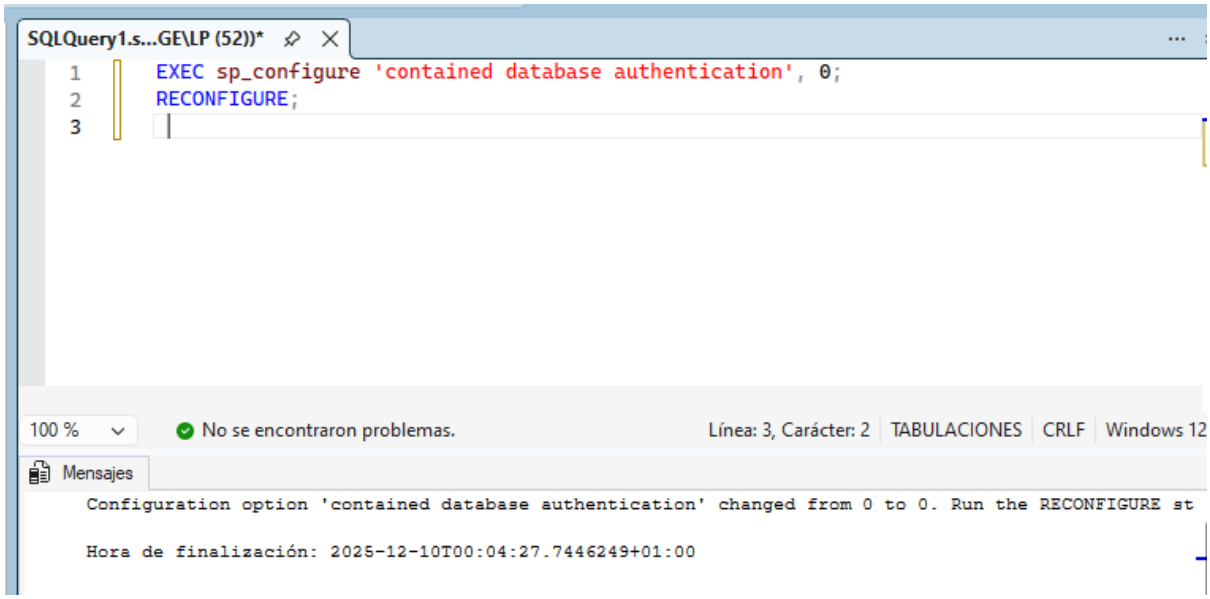
1. En SSMS, abrir una nueva consulta.
2. Verificar si está activa:

```
SELECT name, value_in_use
FROM sys.configurations
WHERE name = 'contained database authentication';
```



3. Si se desea desactivarla (por seguridad):

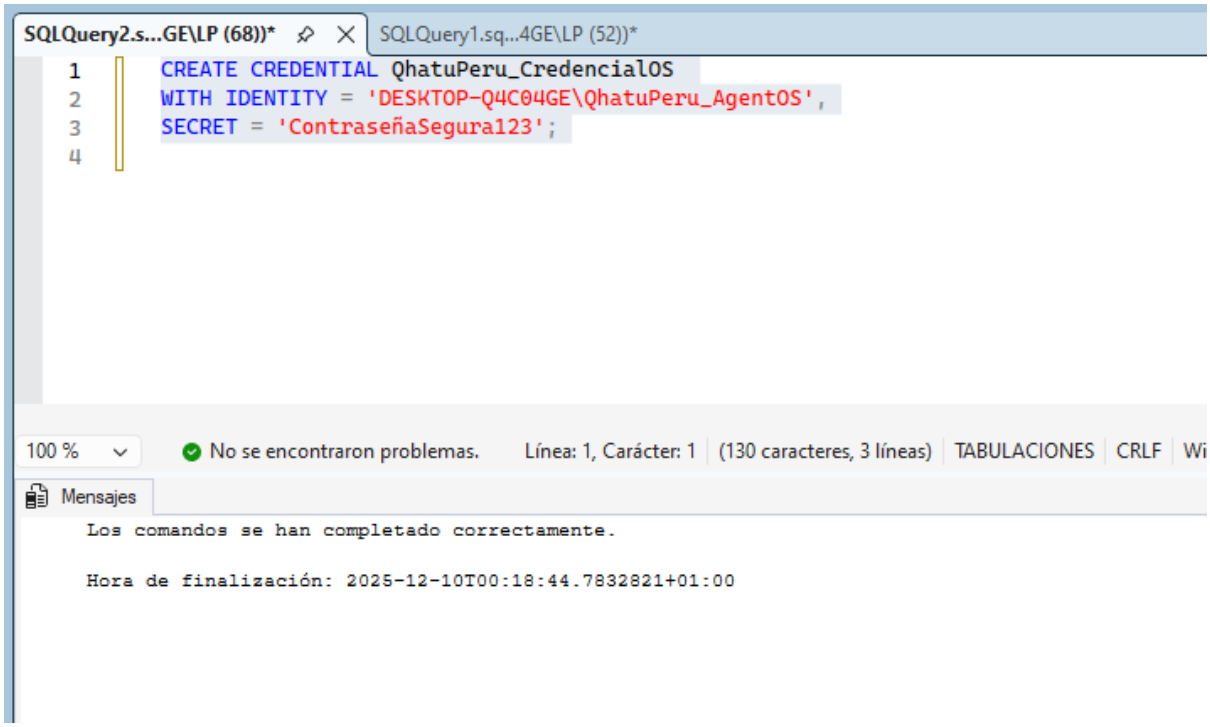
```
EXEC sp_configure 'contained database authentication', 0;
RECONFIGURE;
```



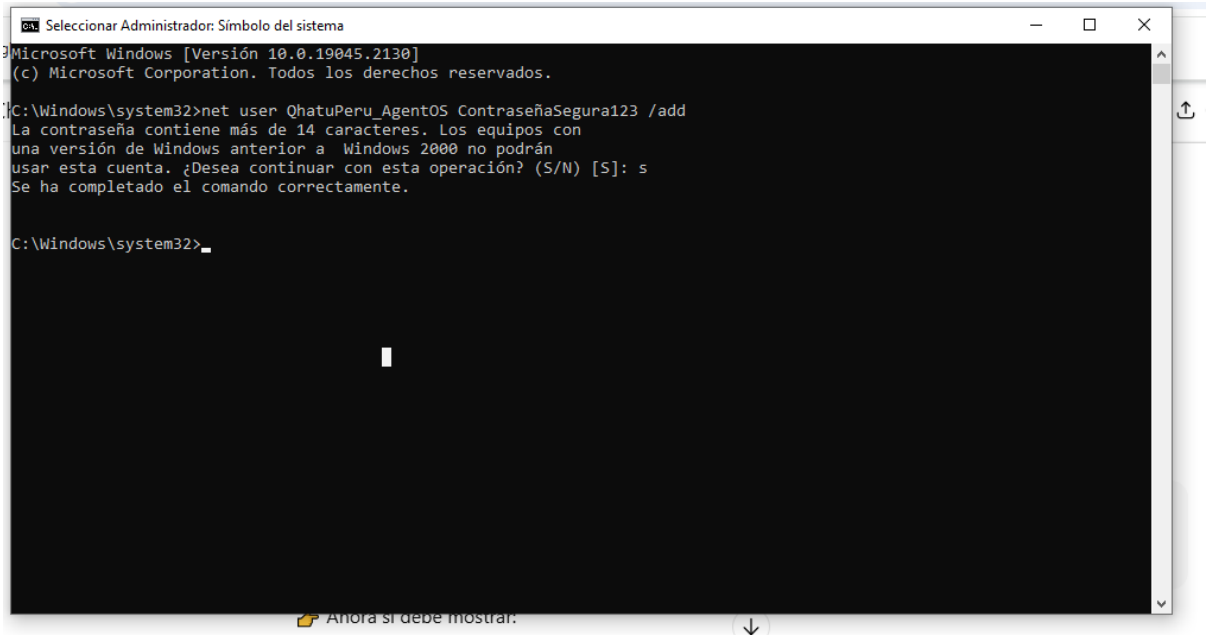
3. Crear una Credencial y un Proxy para SQL Agent

1. Crear una cuenta en Windows con permisos mínimos (por ejemplo QhatuPeru_AgentOS).
2. En SSMS, crear la credencial:

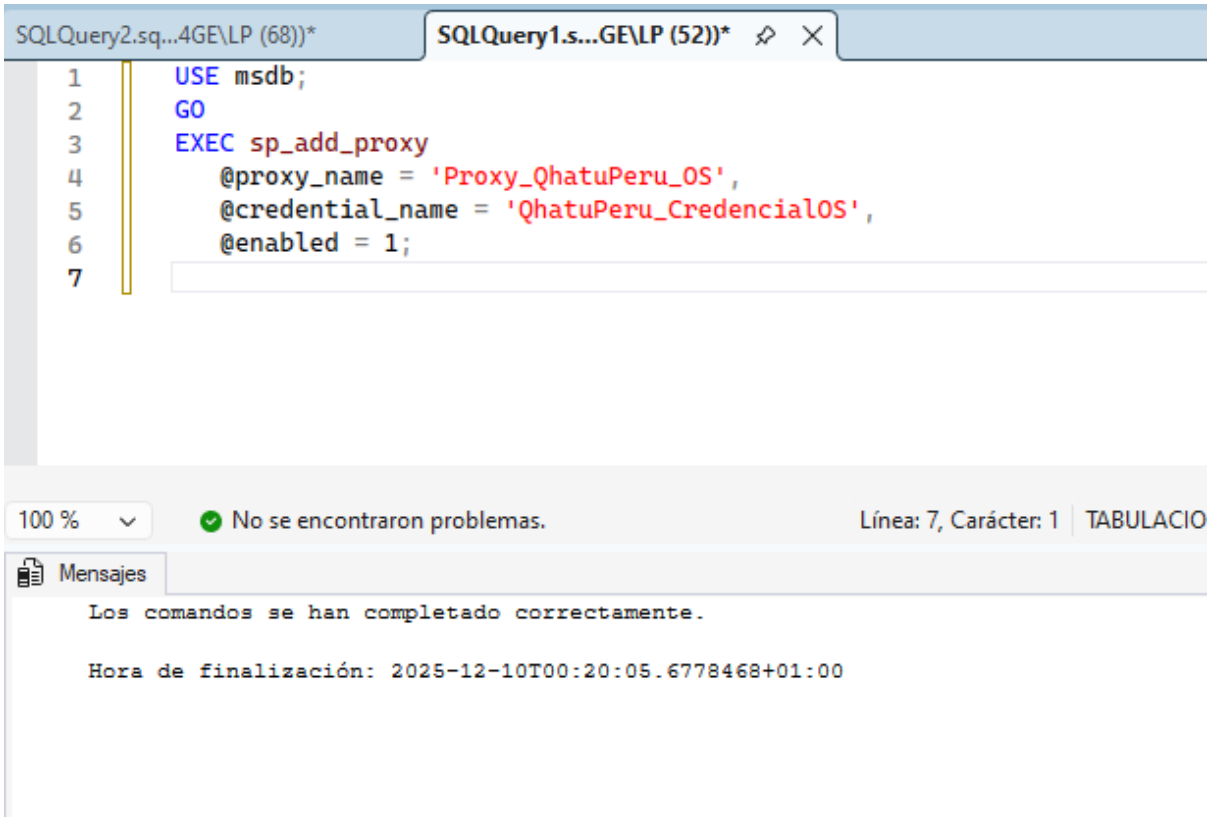
```
C CREATE CREDENTIAL QhatuPeru_CredencialOS
WITH IDENTITY = 'DESKTOP-Q4C04GE\QhatuPeru_AgentOS',
SECRET = 'ContraseñaSegura123';
```



3. Crear el proxy en SQL Agent:



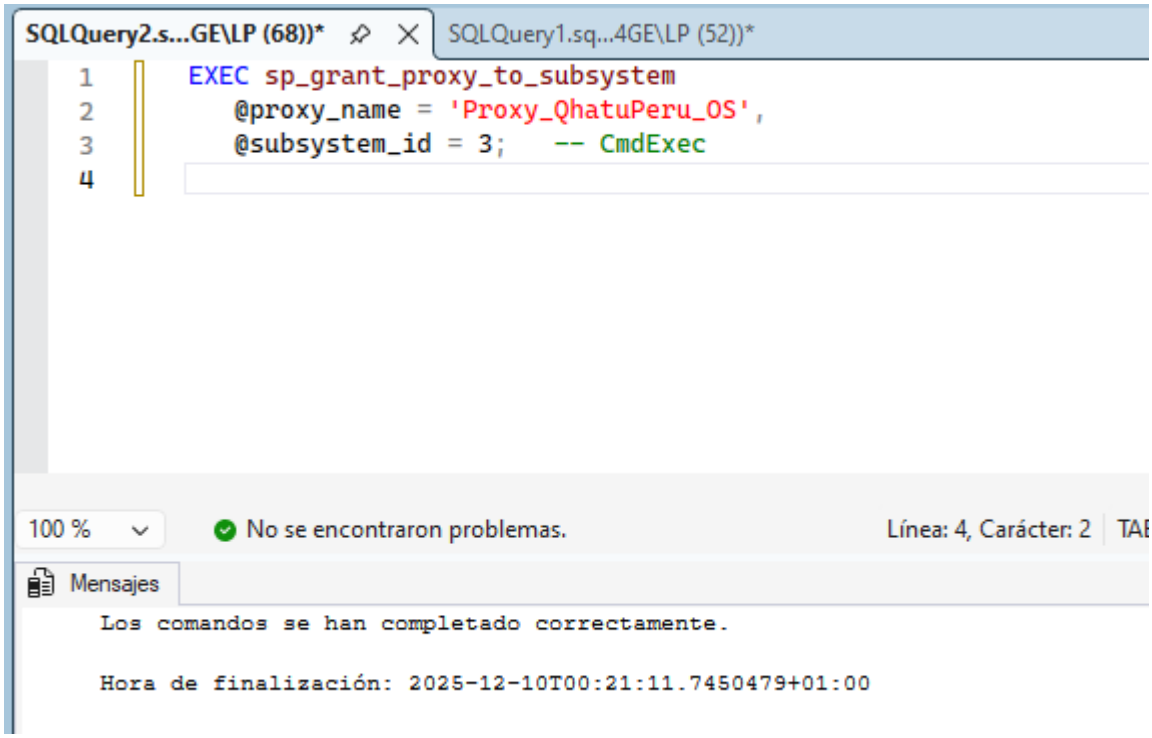
```
USE msdb;
GO
EXEC sp_add_proxy
  @proxy_name = 'Proxy_QhatuPeru_OS',
  @credential_name = 'QhatuPeru_CredencialOS',
  @enabled = 1;
```



4. Habilitarlo para CmdExec u otros subsistemas:

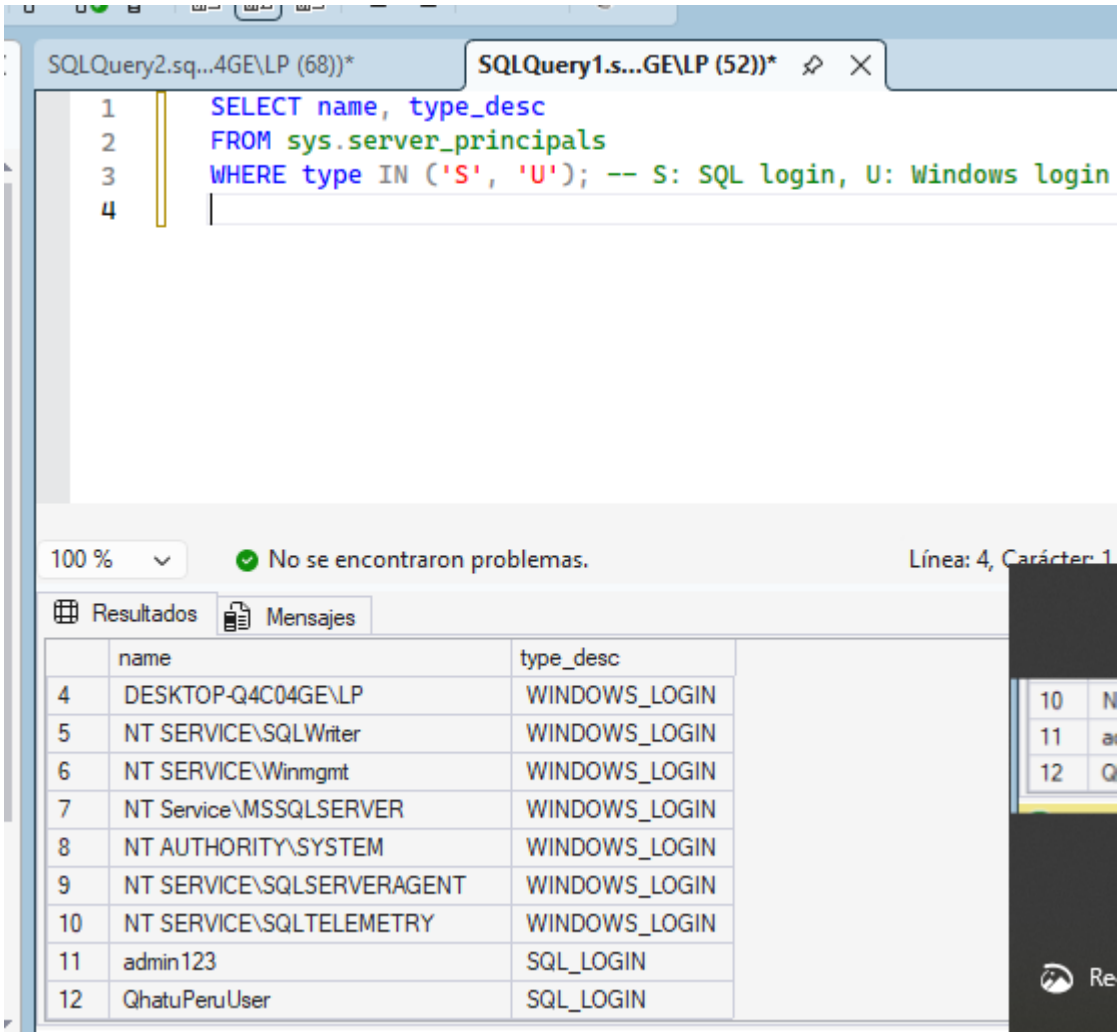
```
EXEC sp_grant_proxy_to_subsystem
    @proxy_name = 'Proxy_QhatuPeru_OS',
```

```
    @subsystem_id = 3;    -- CmdExec
```

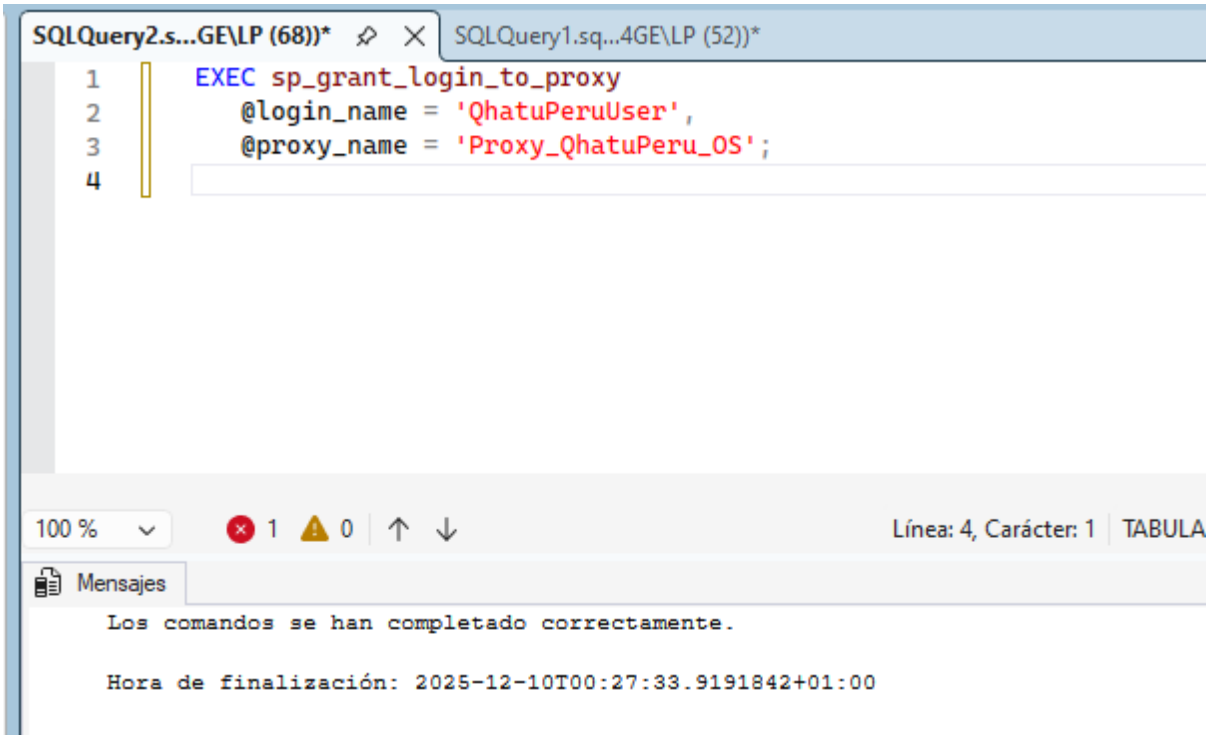


5. Asignar permiso al login que ejecutará el job:
   Ojo(CREATE LOGIN QhatuPeruUser
   WITH PASSWORD = 'ClaveTemporal123',
   CHECK_POLICY = OFF;

)




```
EXEC sp_grant_login_to_proxy
    @login_name = 'QhatuPeruUser',
    @proxy_name = 'Proxy_QhatuPeru_OS';
```



para ver login:

SELECT name, type_desc

FROM sys.server_principals

WHERE type IN ('S', 'U'); -- S: SQL login, U: Windows login

Explicación

Definir correctamente la cuenta de servicio protege la base de datos de accesos indebidos al sistema operativo. Este control básico reduce los riesgos de intrusión o daño a los archivos de datos.

3. CREACIÓN DE ROLES FIJOS Y PERSONALIZADOS

Definición

Los roles son agrupaciones de permisos que simplifican la administración de usuarios. SQL Server incluye roles fijos (como `db_datareader`, `db_datawriter`, `db_owner`) y permite crear roles personalizados según las necesidades.

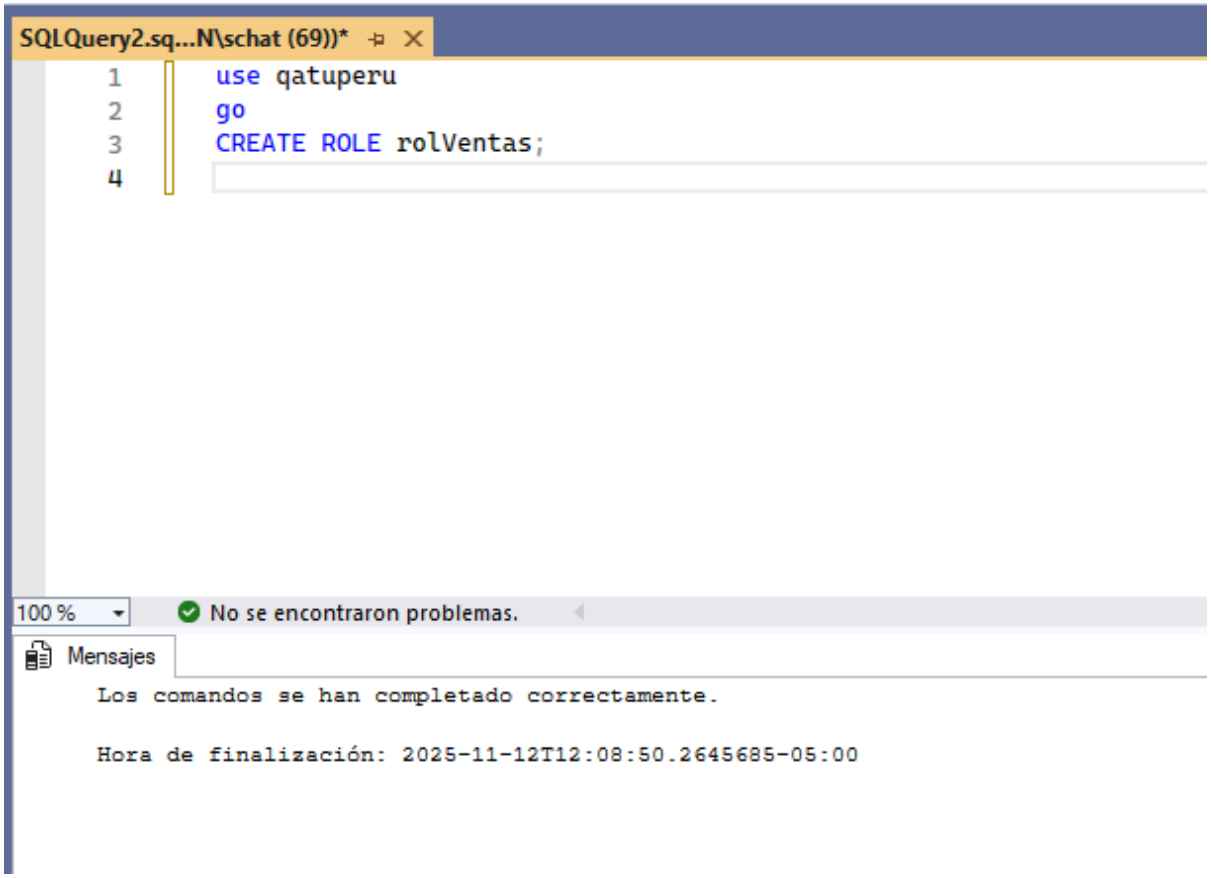
Buenas prácticas

- No asignar permisos directamente a usuarios; usar roles.
- Evitar usar `db_owner` salvo en casos de administración.
- Revisar periódicamente los permisos de cada rol.

Paso a paso

1. Abrir **SSMS** y conectarse a la base de datos deseada.
2. Crear un rol personalizado:

```
CREATE ROLE rolVentas;
```

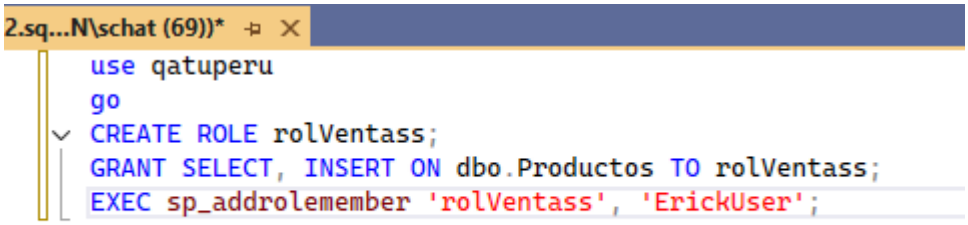


3. Otorgarle permisos:

```
GRANT SELECT, INSERT ON dbo.Productos TO rolVentas;
```

4. Asignar el rol a un usuario:

```
EXEC sp_addrolemember 'rolVentas', 'ErickUser';
```



Explicación

Usar roles facilita la administración de permisos al manejar grupos de usuarios con funciones similares. Esto evita errores y mejora la trazabilidad del control de acceso.

4 Control de acceso con GRANT / DENY / REVOKE

Enunciado

Simular un caso donde un analista necesita ver el inventario (artículos, stock) pero **no** los precios. Crear roles/usuarios y usar **DENY** para impedir `SELECT` sobre `PrecioProveedor` en `ARTICULO` y `PrecioVenta` en `GUIA_DETALLE`.

Objetivo

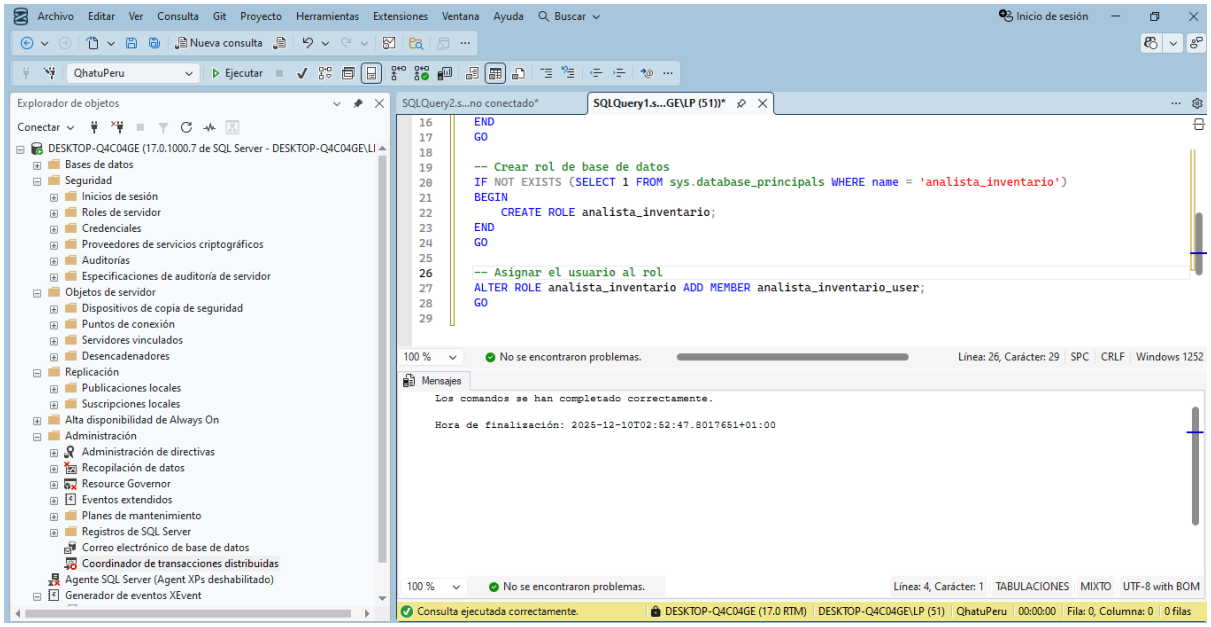
Restringir la visibilidad de columnas de precio a usuarios de negocio que sólo deben ver inventario; permitir lectura de todo lo demás salvo precios.

Paso a paso

1. Crear roles y usuarios (rol `analista_inventario` y login de ejemplo):

```
USE master;
GO
-- Crear login SQL (si no existe)
IF NOT EXISTS (SELECT 1 FROM sys.server_principals
WHERE name = 'QhatuAnalistaUser')
BEGIN
    CREATE LOGIN QhatuAnalistaUser WITH PASSWORD =
'ClaveTemp!23', CHECK_POLICY = OFF;
END
GO

USE QhatuPeru;
GO
-- Crear usuario de base de datos para el login
IF NOT EXISTS (SELECT 1 FROM
sys.database_principals WHERE name =
'analista_inventario_user')
BEGIN
    CREATE USER analista_inventario_user FOR LOGIN
QhatuAnalistaUser;
END
GO

-- Crear rol de base de datos
IF NOT EXISTS (SELECT 1 FROM
sys.database_principals WHERE name =
'analista_inventario')
BEGIN
    CREATE ROLE analista_inventario;
END
GO
```

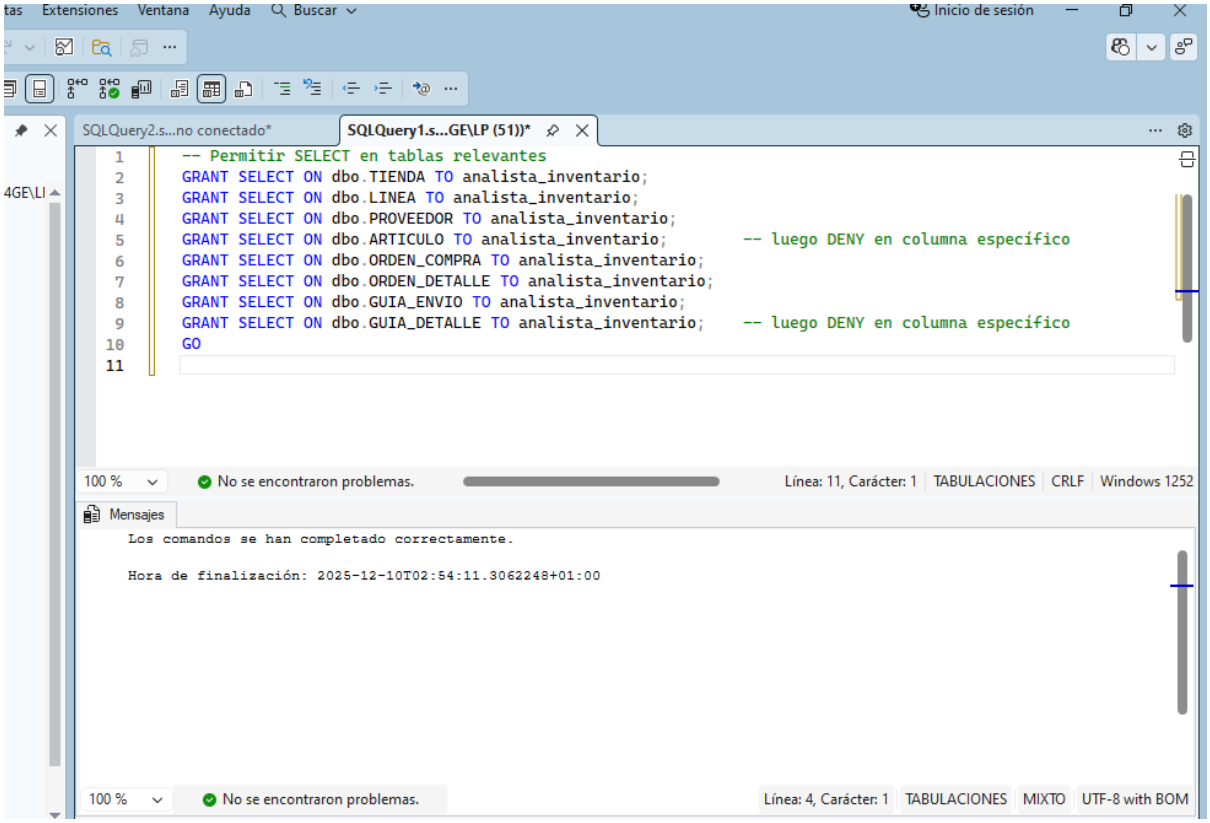
```
-- Asignar el usuario al rol
ALTER ROLE analista_inventario ADD MEMBER
analista_inventario_user;
GO
```



2. Dar permisos generales de SELECT sobre tablas necesarias (sin precios):

```
-- Permitir SELECT en tablas relevantes
GRANT SELECT ON dbo.TIENDA TO analista_inventario;
GRANT SELECT ON dbo.LINEA TO analista_inventario;
GRANT SELECT ON dbo.PROVEEDOR TO
analista_inventario;
GRANT SELECT ON dbo.ARTICULO TO
analista_inventario;          -- luego DENY en
columna específico
GRANT SELECT ON dbo.ORDEN_COMPRA TO
analista_inventario;
GRANT SELECT ON dbo.ORDEN_DETALLE TO
analista_inventario;
GRANT SELECT ON dbo.GUIA_ENVIO TO
analista_inventario;
```

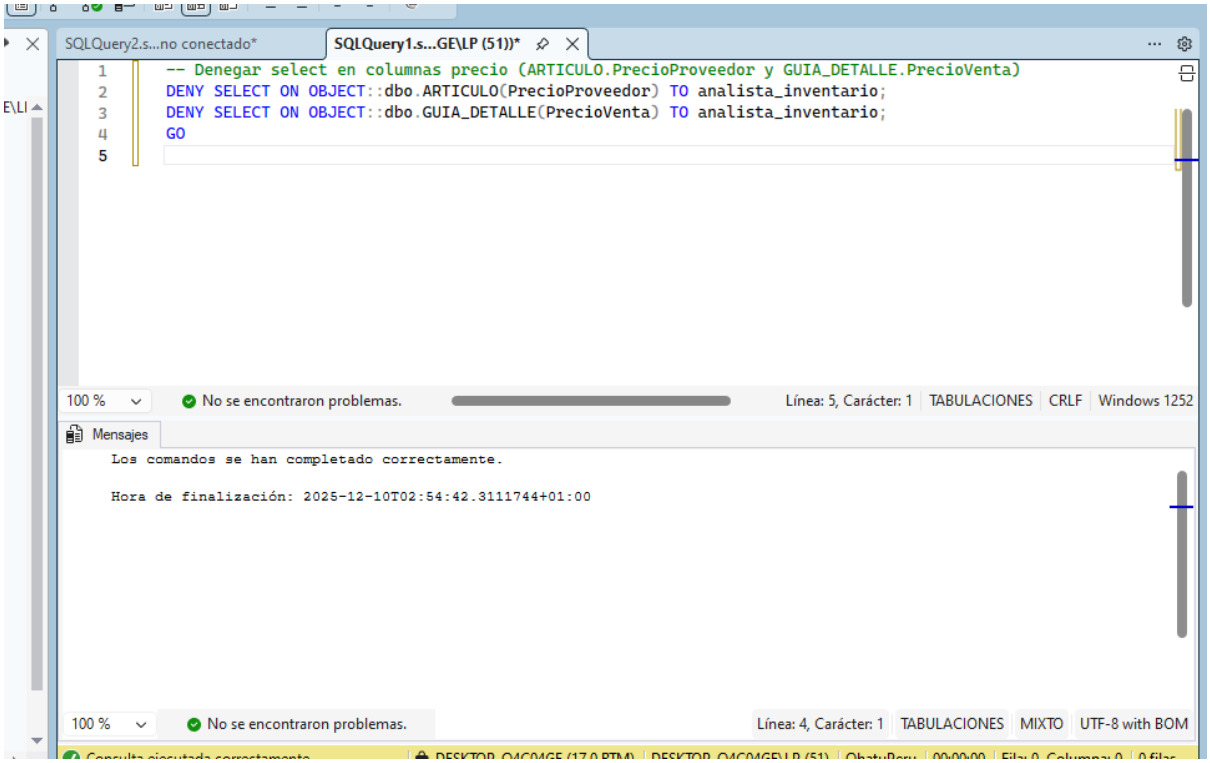
```
GRANT SELECT ON dbo.GUIA_DETALLE TO
analista_inventario;     -- luego DENY en columna
específico
GO
```



3. Denegar SELECT sobre columnas de precio:

```
-- Denegar select en columnas precio
(ARTICULO.PrecioProveedor y
GUIA_DETALLE.PrecioVenta)
DENY SELECT ON
OBJECT::dbo.ARTICULO(PrecioProveedor) TO
analista_inventario;
DENY SELECT ON
OBJECT::dbo.GUIA_DETALLE(PrecioVenta) TO
analista_inventario;
```

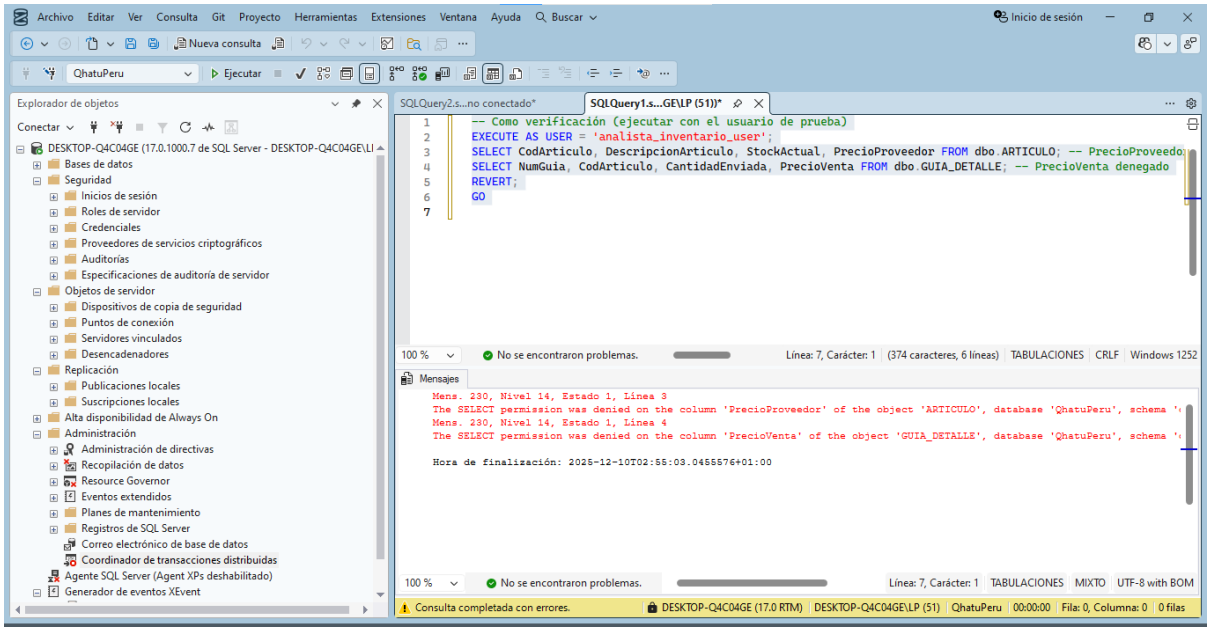
GO



4. Prueba rápida (ejecutar como `analista_inventario_user` o simular):

```
-- Como verificación (ejecutar con el usuario de prueba)
EXECUTE AS USER = 'analista_inventario_user';
SELECT CodArticulo, DescripcionArticulo, StockActual, PrecioProveedor FROM dbo.ARTICULO; -- PrecioProveedor debe fallar o aparecer NULL según driver
SELECT NumGuia, CodArticulo, CantidadEnviada, PrecioVenta FROM dbo.GUIA_DETALLE; -- PrecioVenta denegado
REVERT;
```

GO



Explicación

- `GRANT SELECT ON dbo.ARTICULO` permite leer la tabla; sin embargo, `DENY SELECT ON OBJECT::dbo.ARTICULO(PrecioProveedor)` **anula** ese permiso para la columna específica, impidiendo que el rol vea los precios. Lo mismo para `GUIA_DETALLE.PrecioVenta`.
- DENY tiene prioridad sobre GRANT, por eso garantiza el bloqueo aunque existan permisos heredados.

Buenas prácticas

- Evitar usar DENY masivamente; prefierir principio de *mínimos privilegios* (dar sólo lo necesario).
- Documentar qué columnas están denegadas y por qué.
- No usar usuarios con permisos elevados para tareas diarias.
- Testear con cuentas de prueba que representen cada rol.

- Revisar permisos periódicamente y auditar accesos (ver Proyecto 7).

5 Protección de datos: Implementación básica de TDE (Transparent Data Encryption)

Enunciado

Habilitar TDE en la base `QhatuPeru` para proteger archivos MDF/LDF en reposo. Crear la master key, el certificado de servidor en la base `master`, crear el Database Encryption Key y activar el cifrado.

Objetivo

Proteger los archivos físicos de la base (MDF/LDF) contra acceso no autorizado al nivel de archivos (drives/backups sin autorización).

Paso a paso

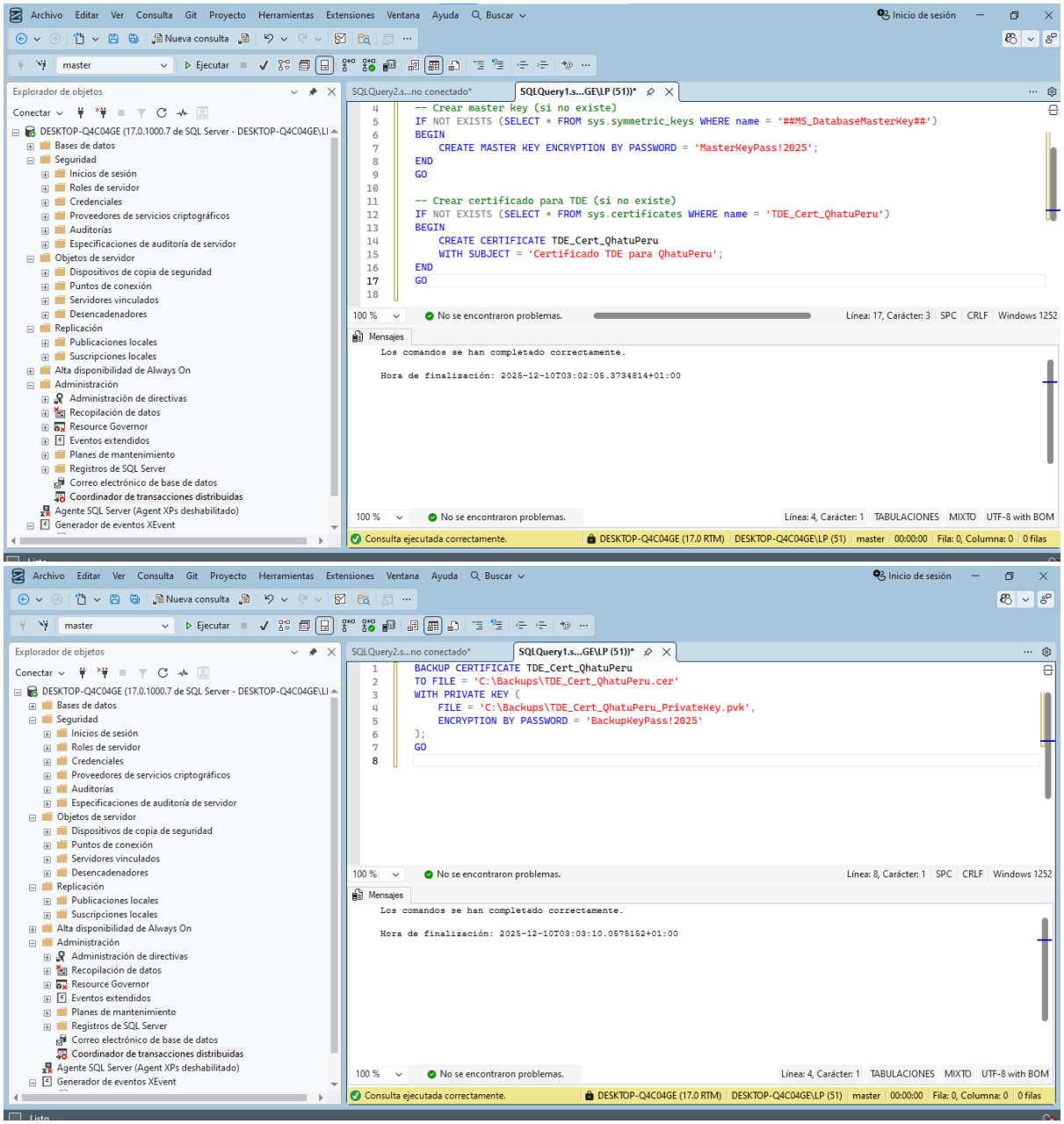
1. Crear master key en `master` (si no existe) y certificado para TDE:

```
USE master;
GO

-- Crear master key (si no existe)
IF NOT EXISTS (SELECT * FROM sys.symmetric_keys
WHERE name = '##MS_DatabaseMasterKey##')
BEGIN
    CREATE MASTER KEY ENCRYPTION BY PASSWORD =
```

```
'MasterKeyPass!2025';
END
GO

-- Crear certificado para TDE (si no existe)
IF NOT EXISTS (SELECT * FROM sys.certificates WHERE
name = 'TDE_Cert_QhatuPeru')
BEGIN
    CREATE CERTIFICATE TDE_Cert_QhatuPeru
    WITH SUBJECT = 'Certificado TDE para
QhatuPeru';
END
GO
```

IMPORTANTE: Haz backup del certificado y la clave privada inmediatamente y guárdalos en lugar seguro (media externa / HSM) para poder restaurar backups cifrados:

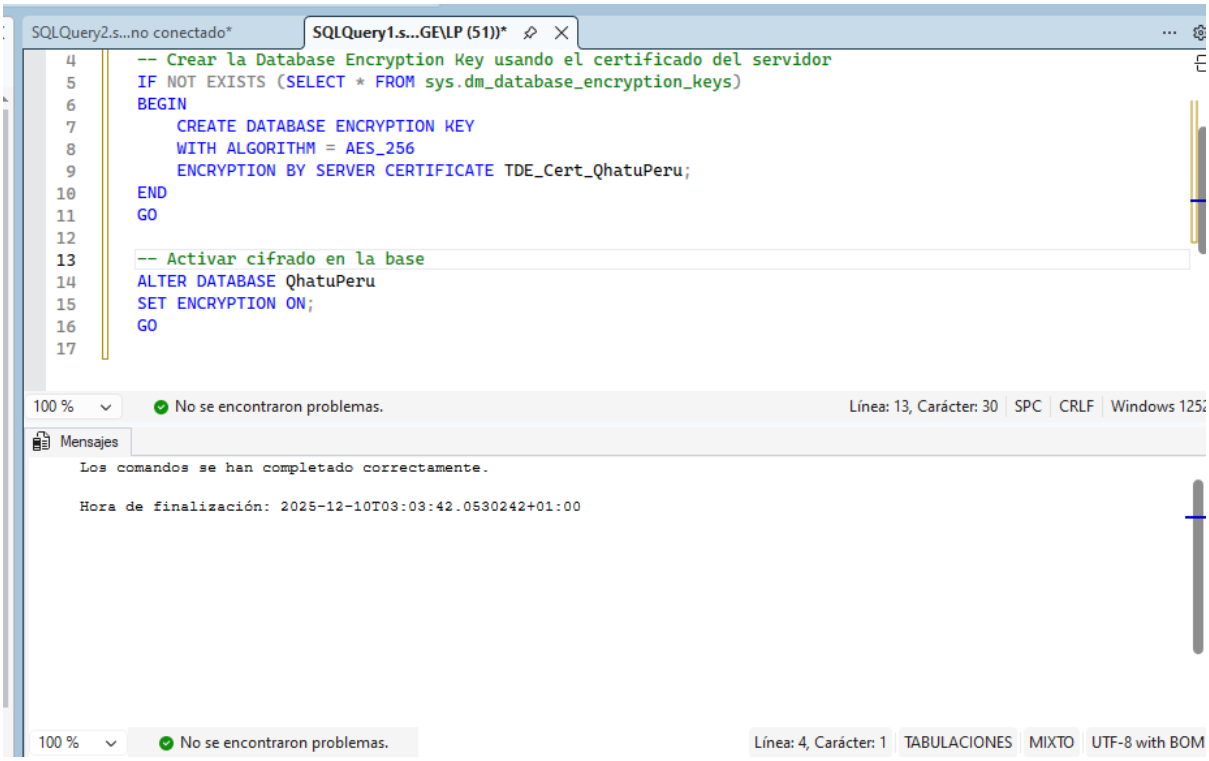
```
BACKUP CERTIFICATE TDE_Cert_QhatuPeru
TO FILE = 'C:\Backups\TDE_Cert_QhatuPeru.cer'
WITH PRIVATE KEY (
    FILE =
'C:\Backups\TDE_Cert_QhatuPeru_PrivateKey.pvk',
    ENCRYPTION BY PASSWORD = 'BackupKeyPass!2025'
);
GO
```



2. Crear Database Encryption Key en QhatuPeru y activar TDE:

```
USE QhatuPeru;
GO

-- Crear la Database Encryption Key usando el
certificado del servidor
IF NOT EXISTS (SELECT * FROM
sys.dm_database_encryption_keys)
BEGIN
```

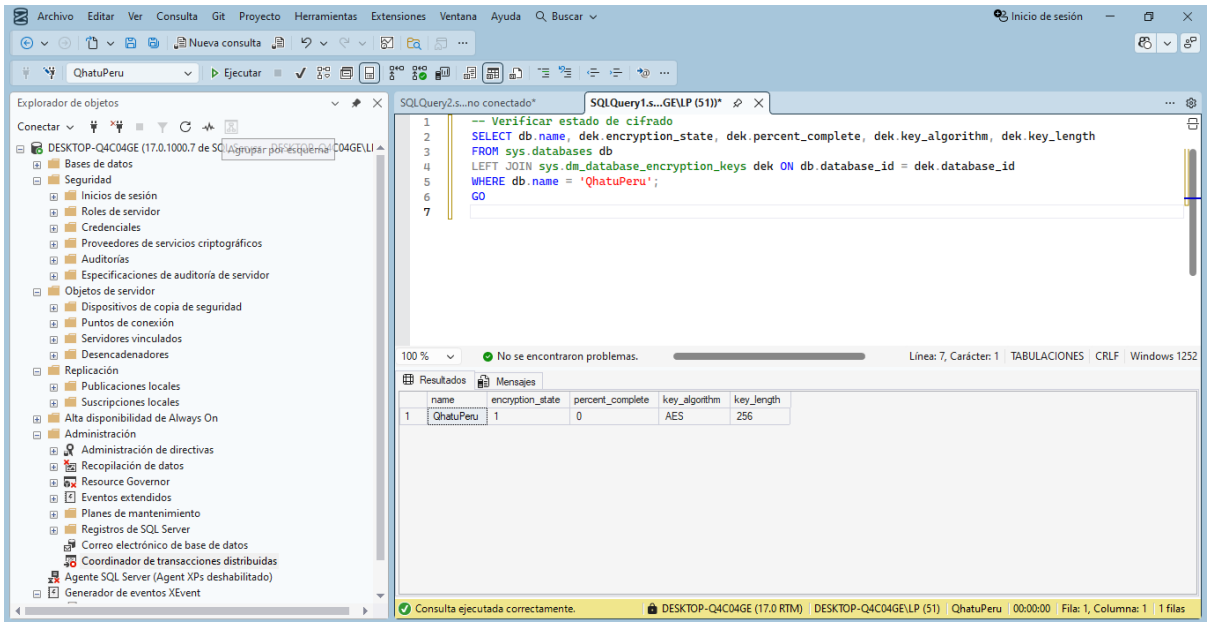
```
    CREATE DATABASE ENCRYPTION KEY
    WITH ALGORITHM = AES_256
    ENCRYPTION BY SERVER CERTIFICATE
TDE_Cert_QhatuPeru;
END
GO

-- Activar cifrado en la base
ALTER DATABASE QhatuPeru
SET ENCRYPTION ON;
GO
```



3. Verificar estado:

```
-- Verificar estado de cifrado
SELECT db.name, dek.encryption_state,
dek.percent_complete, dek.key_algorithm,
dek.key_length
FROM sys.databases db
LEFT JOIN sys.dm_database_encryption_keys dek ON
```

```
db.database_id = dek.database_id
WHERE db.name = 'QhatuPeru';
GO
```



Explicación

- TDE cifra los archivos de datos y logs en reposo; el certificado usado por TDE debe existir en `master`.
- Hacer backup del certificado y su clave privada es crítico para poder restaurar la BD cifrada en otro servidor.

Buenas prácticas

- Guardar backups del certificado y su clave en un almacén seguro y fuera del servidor (HSM, Vault).
- Probar restauraciones en una instancia de prueba.
- Usar AES_256.
- Documentar la contraseña usada para el `CREATE MASTER KEY` guardada de forma segura.

6 Implementación de Always Encrypted (columna de datos sensibles)

Enunciado

Configurar un ejemplo de Always Encrypted para la columna `PrecioProveedor` (o crear `PrecioProveedor_ENC`) usando una Column Master Key (CMK) alojada en el almacén de certificados y una Column Encryption Key (CEK). Mostrar el DDL que crea la columna cifrada.

Objetivo

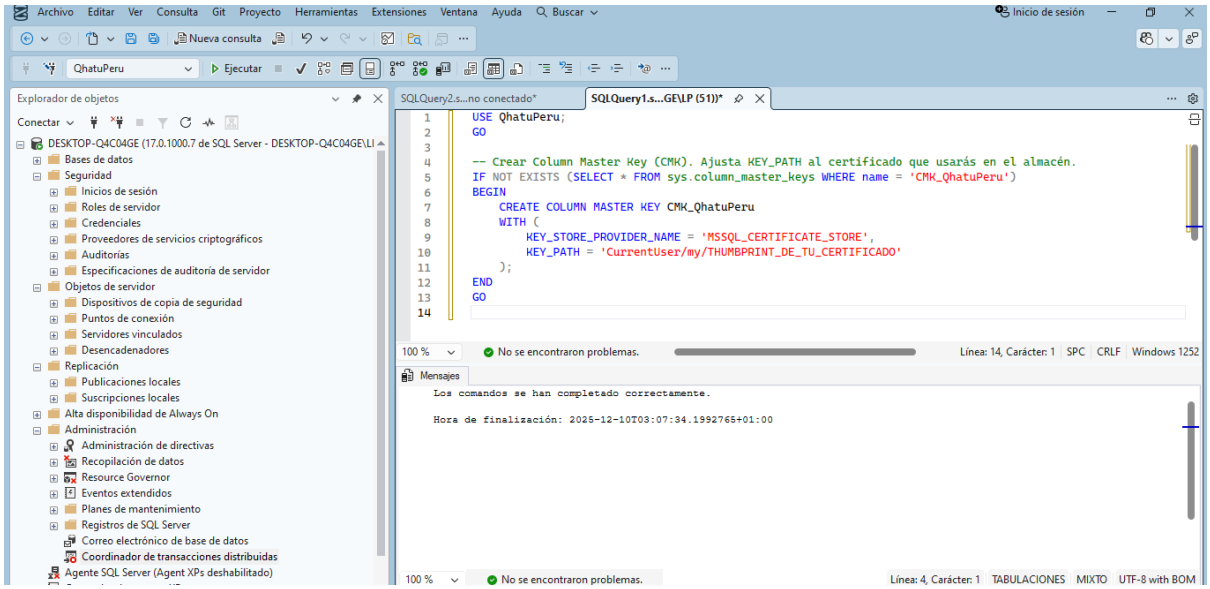
Proteger en el cliente y en tránsito la columna de precios de proveedor de modo que el servidor SQL no pueda ver los valores en texto claro (las aplicaciones autorizadas con CMK/CEK pueden descifrarlos).

Paso a paso (con código)

Nota: **Always Encrypted** requiere soporte del cliente (driver/SSMS) para crear la CEK y generar el `encrypted_value`. En T-SQL **puedes** crear la Column Master Key (si el almacén existe) pero normalmente la creación de la CEK y valores cifrados se hace desde el cliente (SSMS/PowerShell/C#). Aquí te doy los pasos + DDL de la columna.

1. Crear Column Master Key (ejemplo usando almacén de certificados Windows):

```
USE QhatuPeru;
GO

-- Crear Column Master Key (CMK). Ajusta KEY_PATH
al certificado que usarás en el almacén.
IF NOT EXISTS (SELECT * FROM sys.column_master_keys
WHERE name = 'CMK_QhatuPeru')
BEGIN
    CREATE COLUMN MASTER KEY CMK_QhatuPeru
    WITH (
        KEY_STORE_PROVIDER_NAME =
'MSSQL_CERTIFICATE_STORE',
        KEY_PATH =
'CurrentUser/my/THUMBPRINT_DE_TU_CERTIFICADO'
    );
END
GO
```



2. (Desde cliente/SSMS) Crear Column Encryption Key (CEK). **Ejemplo conceptual**: la creación de CEK normalmente la realiza SSMS o herramientas que generan el

encrypted_value. Aquí indico el comando con un placeholder para encrypted_value:

```
-- Este paso suele hacerse desde SSMS o driver; aquí se muestra la estructura T-SQL con placeholder
USE QhatuPeru;
GO

IF NOT EXISTS (SELECT * FROM sys.column_encryption_keys WHERE name = 'CEK_QhatuPeru')
BEGIN
    CREATE COLUMN ENCRYPTION KEY CEK_QhatuPeru
    WITH VALUES (
        (
            COLUMN_MASTER_KEY = CMK_QhatuPeru,
            ALGORITHM = 'RSA_OAEP',
            ENCRYPTED_VALUE = 0x... -- <-- Valor binario generado por el cliente/SSMS
        )
    );
END
GO
```

3. Crear nueva columna cifrada o alterar tabla para agregar columna PrecioProveedor_ENC:

```
USE QhatuPeru;
GO

-- Agregar columna nueva que usará Always Encrypted (DETERMINISTIC o RANDOMIZED según necesidades)
```

```
ALTER TABLE dbo.ARTICULO
ADD PrecioProveedor_ENC decimal(18,2)
    COLLATE Latin1_General_BIN2
    ENCRYPTED WITH (
        COLUMN_ENCRYPTION_KEY = CEK_QhatuPeru,
        ENCRYPTION_TYPE = DETERMINISTIC,
        ALGORITHM = 'AEAD_AES_256_CBC_HMAC_SHA_256'
    ) NULL;
GO
```

4. Migración de datos (cliente): leer `PrecioProveedor` y escribir en `PrecioProveedor_ENC` usando un cliente que tenga acceso a la CMK/CEK (por ejemplo, SSMS con Always Encrypted enabled, o una app .NET configurada). Ejemplo conceptual (se hace desde cliente):

```
-- Ejecutar desde SSMS con soporte Always Encrypted
ON:
UPDATE dbo.ARTICULO
SET PrecioProveedor_ENC = PrecioProveedor
WHERE PrecioProveedor IS NOT NULL;
GO
```

Explicación

- CMK (Column Master Key) apunta al proveedor de claves (certificado, HSM).
- CEK (Column Encryption Key) cifra datos de columna y su `encrypted_value` se genera por el cliente.

- El servidor almacena únicamente datos cifrados; sólo los clientes con acceso a las claves pueden descifrar.

Buenas prácticas

- Usar determinístico solo cuando necesites búsquedas/joins encriptadas; para mayor seguridad usar randomized.
- Guardar CMK en HSM o almacén seguro (no solo en el servidor).
- Probar la migración en ambiente de pruebas.
- Actualizar las aplicaciones para que usen drivers compatibles con Always Encrypted.

7 Auditoría de seguridad: crear SQL Server Audit para inicios de sesión y cambios de esquema

Enunciado

Configurar un **Server Audit** que registre intentos de login fallidos y exitosos, y un **Database Audit Specification** que registre cambios DDL (CREATE/ALTER/DROP) en la base `QhatuPeru` sobre objetos críticos.

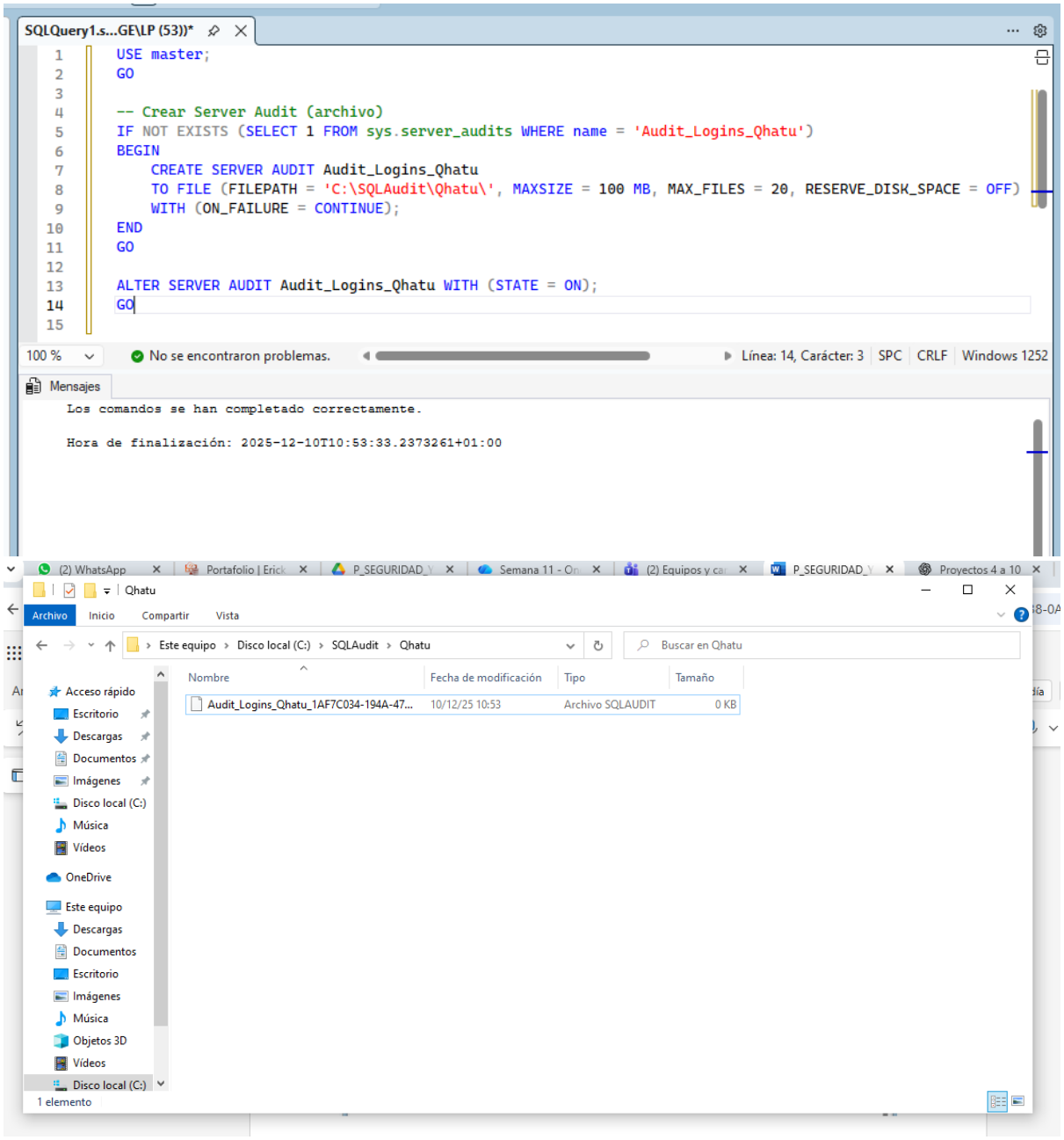
Objetivo

Mantener registro de accesos al servidor y cambios de esquema para auditoría forense y cumplimiento.

Paso a paso (con código)

1. Crear carpeta para archivos de auditoría (ejecutar en SO) y crear Server Audit:

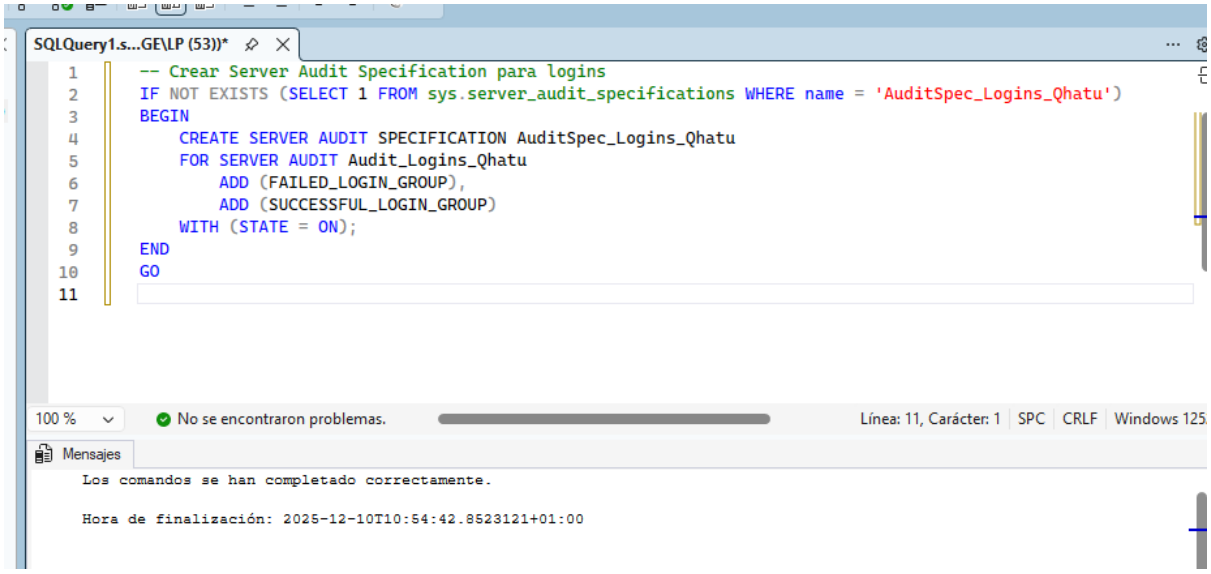
```
USE master;
GO

-- Crear Server Audit (archivo)
IF NOT EXISTS (SELECT 1 FROM sys.server_audits
WHERE name = 'Audit_Logins_Qhatu')
BEGIN
    CREATE SERVER AUDIT Audit_Logins_Qhatu
    TO FILE (FILEPATH = 'C:\SQLAudit\Qhatu\',
MAXSIZE = 100 MB, MAX_FILES = 20,
RESERVE_DISK_SPACE = OFF)
    WITH (ON_FAILURE = CONTINUE);
END
GO

ALTER SERVER AUDIT Audit_Logins_Qhatu WITH (STATE =
ON);
GO
```



2. Agregar acciones de inicio de sesión (exitoso y fallido) al Server Audit:

```
-- Crear Server Audit Specification para logins
IF NOT EXISTS (SELECT 1 FROM
sys.server_audit_specifications WHERE name =
'AuditSpec_Logins_Qhatu')
BEGIN
    CREATE SERVER AUDIT SPECIFICATION
```

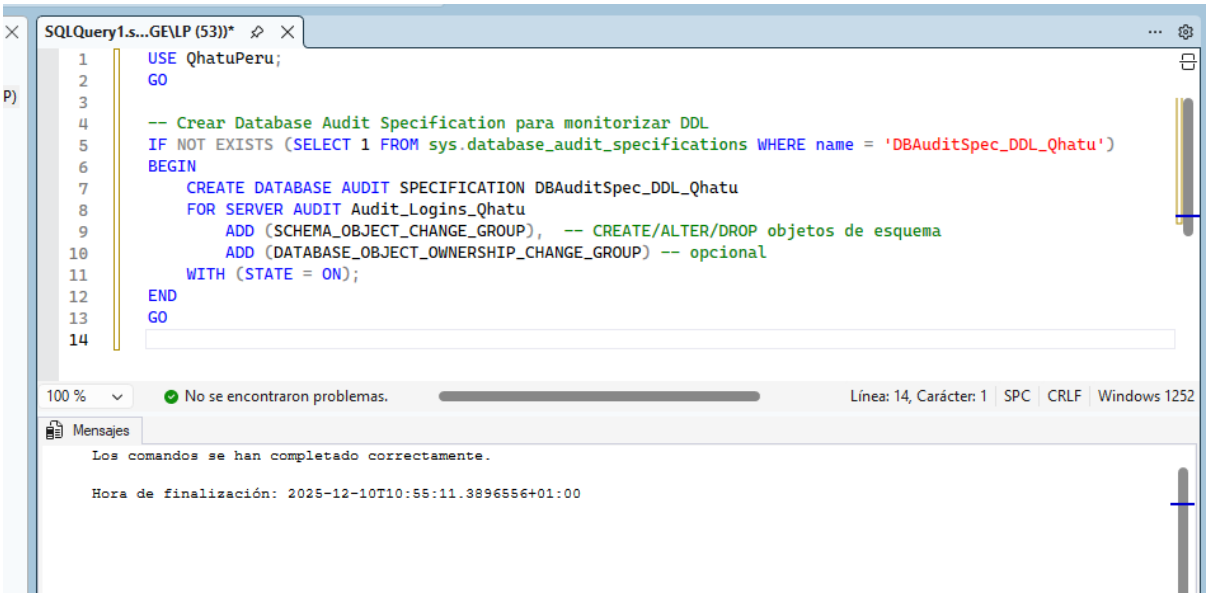
```
AuditSpec_Logins_Qhatu
    FOR SERVER AUDIT Audit_Logins_Qhatu
        ADD (FAILED_LOGIN_GROUP),
        ADD (SUCCESSFUL_LOGIN_GROUP)
    WITH (STATE = ON);
END
GO
```



3. Crear Database Audit Specification para DDL en QhatuPeru:

```
USE QhatuPeru;
GO

-- Crear Database Audit Specification para
monitorizar DDL
IF NOT EXISTS (SELECT 1 FROM
sys.database_audit_specifications WHERE name =
'DBAuditSpec_DDL_Qhatu')
BEGIN
    CREATE DATABASE AUDIT SPECIFICATION
DBAuditSpec_DDL_Qhatu
    FOR SERVER AUDIT Audit_Logins_Qhatu
```

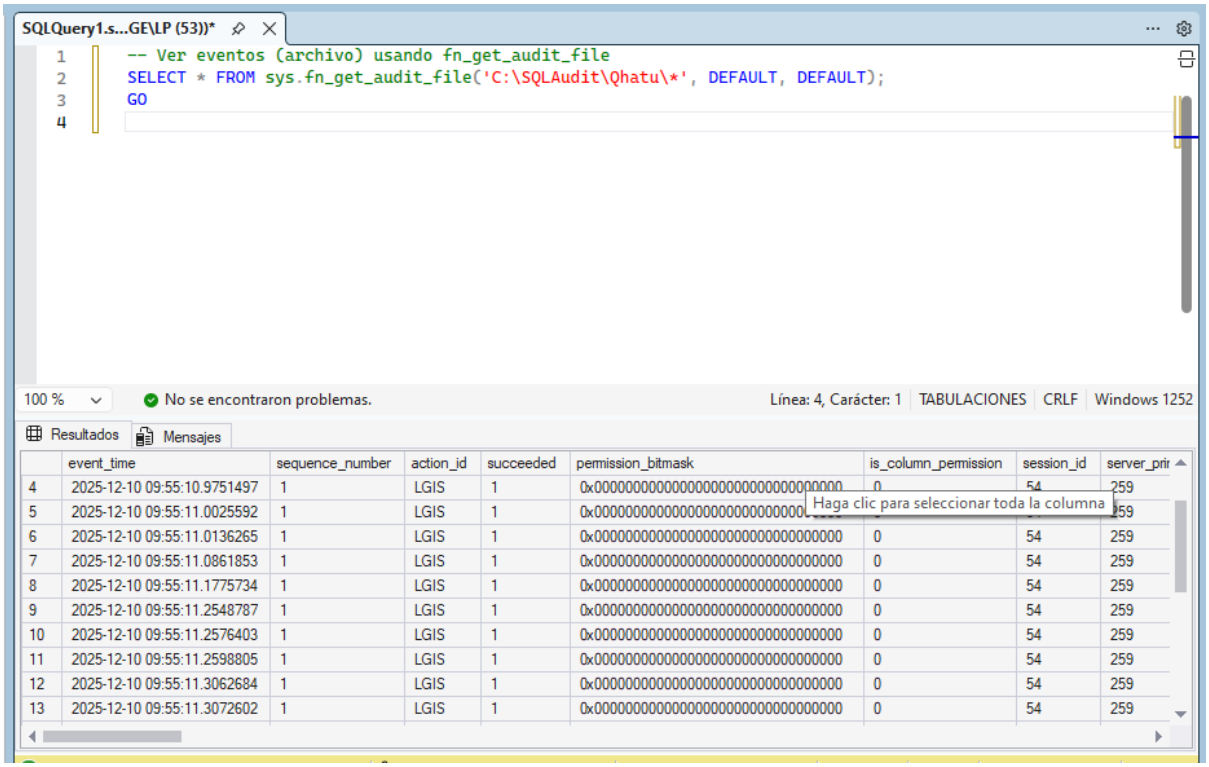
```
        ADD (SCHEMA_OBJECT_CHANGE_GROUP),   --
CREATE/ALTER/DROP objetos de esquema
        ADD
(DATABASE_OBJECT_OWNERSHIP_CHANGE_GROUP) --
opcional
    WITH (STATE = ON);
END
GO
```



4. Verificar eventos:

```
-- Ver eventos (archivo) usando fn_get_audit_file
SELECT * FROM
sys.fn_get_audit_file('C:\SQLAudit\Qhatu\*',
DEFAULT, DEFAULT);
```

GO



	event_time	sequence_number	action_id	succeeded	permission_bitmask	is_column_permission	session_id	server_prin
4	2025-12-10 09:55:10.9751497	1	LGIS	1	0x0000000000000000000000000000000000	0	54	259
5	2025-12-10 09:55:11.0025592	1	LGIS	1	0x0000000000000000000000000000000000	0	54	259
6	2025-12-10 09:55:11.0136265	1	LGIS	1	0x0000000000000000000000000000000000	0	54	259
7	2025-12-10 09:55:11.0861853	1	LGIS	1	0x0000000000000000000000000000000000	0	54	259
8	2025-12-10 09:55:11.1775734	1	LGIS	1	0x0000000000000000000000000000000000	0	54	259
9	2025-12-10 09:55:11.2548787	1	LGIS	1	0x0000000000000000000000000000000000	0	54	259
10	2025-12-10 09:55:11.2576403	1	LGIS	1	0x0000000000000000000000000000000000	0	54	259
11	2025-12-10 09:55:11.2598805	1	LGIS	1	0x0000000000000000000000000000000000	0	54	259
12	2025-12-10 09:55:11.3062684	1	LGIS	1	0x0000000000000000000000000000000000	0	54	259
13	2025-12-10 09:55:11.3072602	1	LGIS	1	0x0000000000000000000000000000000000	0	54	259

Explicación

- `Server Audit` define el destino (archivo) y `Server Audit Specification` indica qué acciones de servidor registrar.
- `Database Audit Specification` enlaza al `Server Audit` y define acciones dentro de la base (por ejemplo cambios DDL).
- Los grupos predefinidos (`FAILED_LOGIN_GROUP`, `SCHEMA_OBJECT_CHANGE_GROUP`) facilitan la captura de eventos comunes.

Buenas prácticas

- Rotación y retención de archivos de auditoría.
- Asegurar permisos y cifrado en el directorio de auditoría.
- Revisiones periódicas de los logs de auditoría.
- Integrar con SIEM si existe (para alertas).

8 Monitoreo de eventos y alertas con Extended Events + Auditoría

Enunciado

Configurar una sesión de Extended Events que capture `deadlocks` y eventos de `login_failed`, guardar en archivo y crear una vista/función que permita consultar los XEvent desde la base `QhatuPeru`.

Objetivo

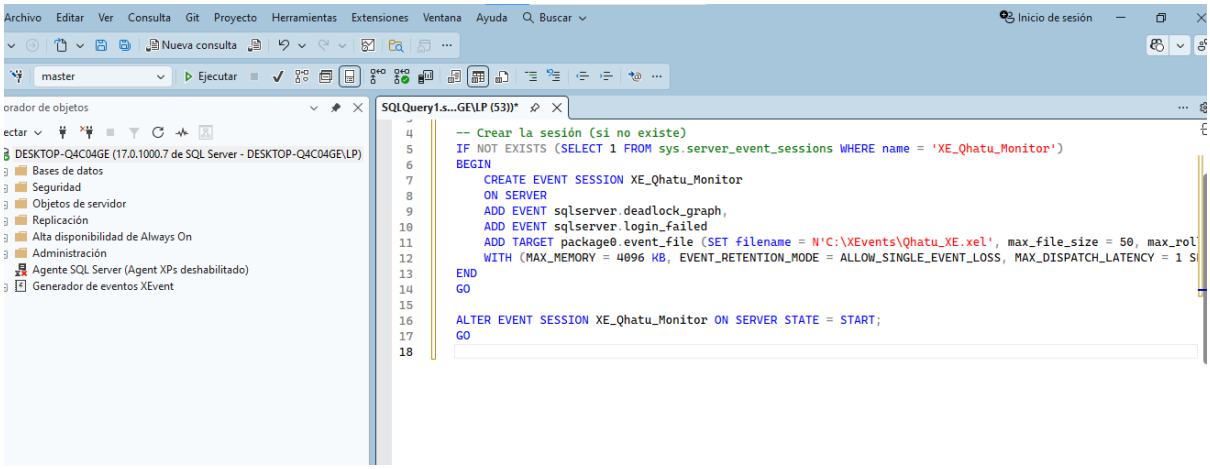
Capturar eventos operacionales críticos (deadlocks, fallos de login) para diagnóstico y alertas.

Paso a paso (con código)

1. Crear sesión Extended Events y escribir a archivo:

```
USE master;
GO

-- Crear la sesión (si no existe)
IF NOT EXISTS (SELECT 1 FROM
sys.server_event_sessions WHERE name =
'XE_Qhatu_Monitor')
BEGIN
    CREATE EVENT SESSION XE_Qhatu_Monitor
    ON SERVER
    ADD EVENT sqlserver.deadlock_graph,
    ADD EVENT sqlserver.login_failed
    ADD TARGET package0.event_file (SET filename =
```

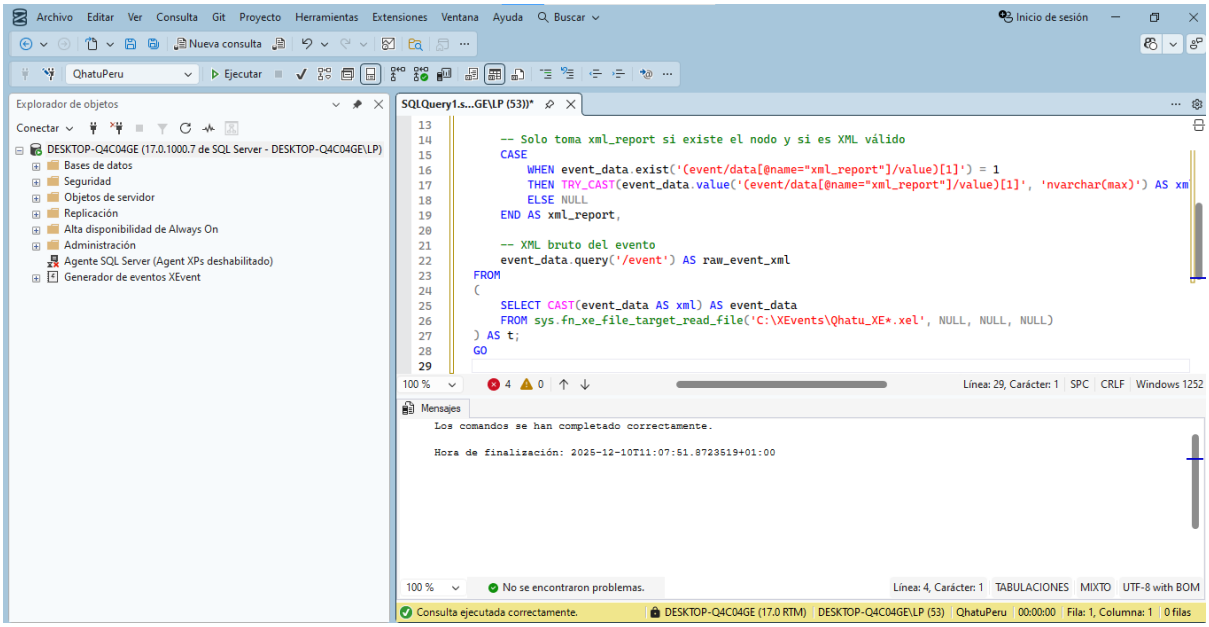
```
N'C:\XEvents\Qhatu_XE.xel', max_file_size = 50,
max_rollover_files = 5)
    WITH (MAX_MEMORY = 4096 KB,
EVENT_RETENTION_MODE = ALLOW_SINGLE_EVENT_LOSS,
MAX_DISPATCH_LATENCY = 1 SECONDS);
END
GO

ALTER EVENT SESSION XE_Qhatu_Monitor ON SERVER
STATE = START;
GO
```



2. Crear vista en la base QhatuPeru para leer eventos (usamos sys.fn_xe_file_target_read_file):

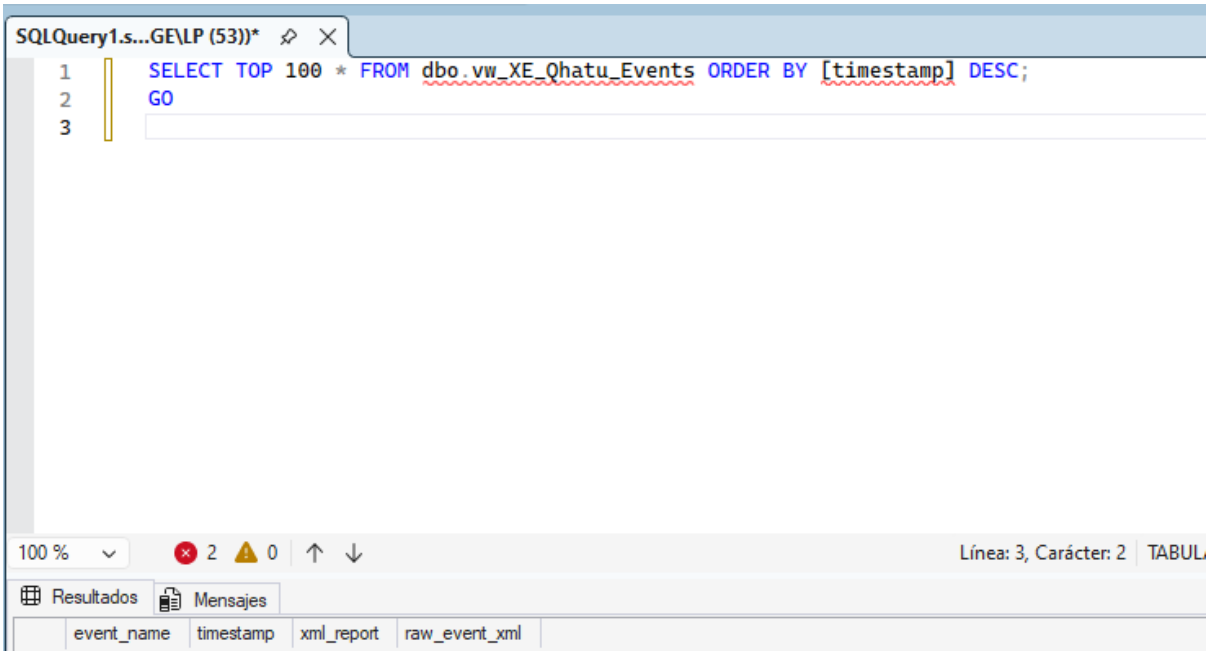
```
USE QhatuPeru;
GO

IF OBJECT_ID('dbo.vw_XE_Qhatu_Events', 'V') IS NOT NULL
   DROP VIEW dbo.vw_XE_Qhatu_Events;
GO

CREATE VIEW dbo.vw_XE_Qhatu_Events
AS
SELECT
```

```
   event_data.value('(event/@name)[1]', 'varchar(100)') AS
event_name,
   event_data.value('(event/@timestamp)[1]', 'datetime2') AS
[timestamp],

   -- Solo toma xml_report si existe el nodo y si es XML válido
   CASE
      WHEN
event_data.exist('(event/data[@name="xml_report"]/value)[
1]') = 1
      THEN
TRY_CAST(event_data.value('(event/data[@name="xml_rep
ort"]/value)[1]', 'nvarchar(max)') AS xml)
      ELSE NULL
   END AS xml_report,

   -- XML bruto del evento
   event_data.query('/event') AS raw_event_xml
FROM
(
   SELECT CAST(event_data AS xml) AS event_data
   FROM
sys.fn_xe_file_target_read_file('C:\XEvents\Qhatu_XE*.xel',
NULL, NULL, NULL)
) AS t;
GO
```



3. Consultar la vista:

```
SELECT TOP 100 * FROM dbo.vw_XE_Qhatu_Events ORDER
BY [timestamp] DESC;
GO
```



Explicación

- deadlock_graph captura información completa del deadlock para análisis.

- `login_failed` detecta intentos de acceso fallidos (útil junto con la auditoría).
- Los archivos `.xel` son leídos por `fn_xe_file_target_read_file` y la vista facilita consultas desde la BD.

Buenas prácticas

- Limitar tamaño y rotación de archivos XE.
- Integrar con alertas (SQL Agent) si se detecta un deadlock o número anómalo de `login_failed`.
- No dejar sesiones XE demasiado verbosas en producción sin control.

9 Implementación de enmascaramiento dinámico + acceso controlado

Enunciado

Aplicar **Dynamic Data Masking (DDM)** a columnas sensibles (por ejemplo `Telefono` en PROVEEDOR) y crear una vista segura para usuarios que necesiten ver datos completos mediante una función que valide rol.

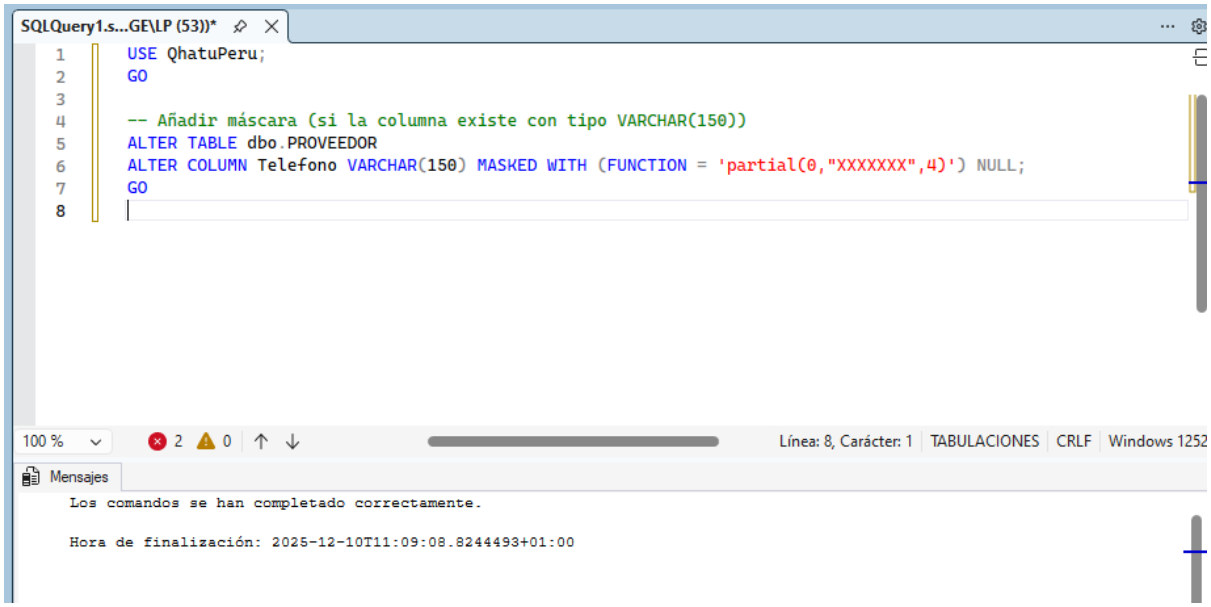
Objetivo

Reducir exposición accidental de datos sensibles en aplicaciones y herramientas de desarrollo; permitir acceso completo solo a roles autorizados.

Paso a paso (con código)

1. Aplicar máscara a PROVEEDOR.Telefono:

```
USE QhatuPeru;
GO

-- Añadir máscara (si la columna existe con tipo
VARCHAR(150))
ALTER TABLE dbo.PROVEEDOR
ALTER COLUMN Telefono VARCHAR(150) MASKED WITH
(FUNCTION = 'partial(0,"XXXXXXX",4)') NULL;
GO
```

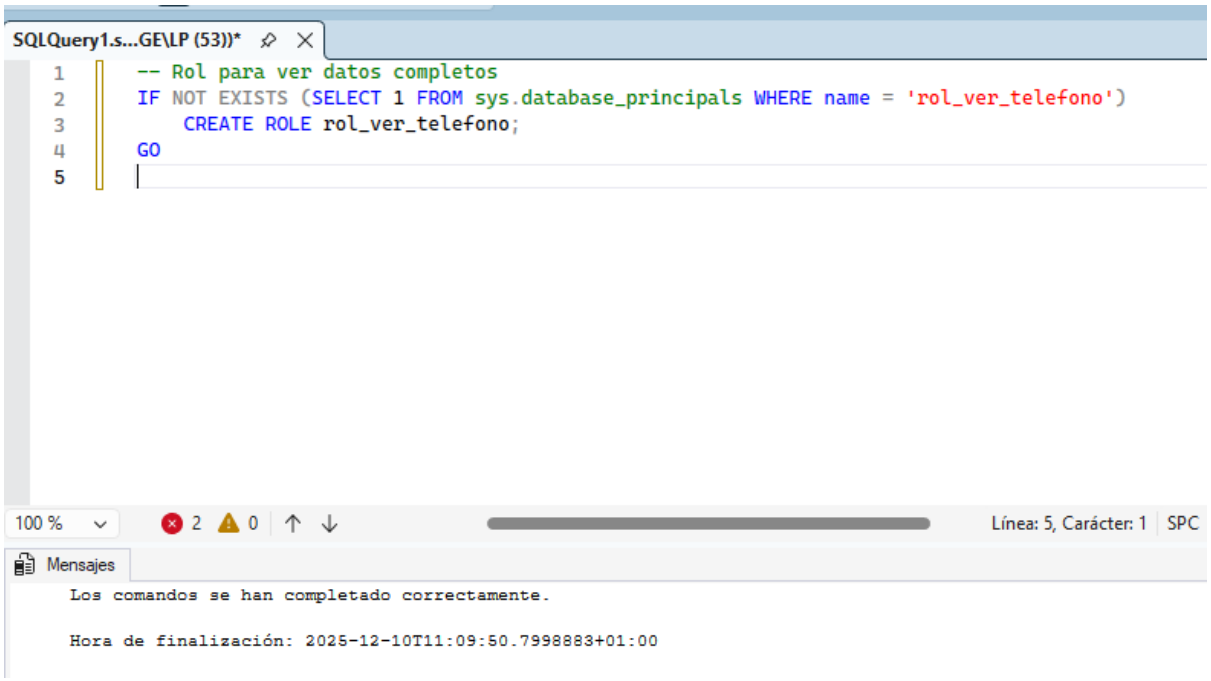


(La máscara partial(0,"XXXXXXX",4) muestra los últimos 4 dígitos; ajusta según preferencia.)

2. Crear rol que puede ver datos sin máscara (rol_ver_telefono) y asignarlo a un usuario de ejemplo:

```
-- Rol para ver datos completos
IF NOT EXISTS (SELECT 1 FROM
sys.database_principals WHERE name =
'rol_ver_telefono')
    CREATE ROLE rol_ver_telefono;
```

```
GO

-- Supongamos que existe el usuario
analista_inventario_user; agregamos al rol
ALTER ROLE rol_ver_telefono ADD MEMBER
analista_inventario_user;
GO
```

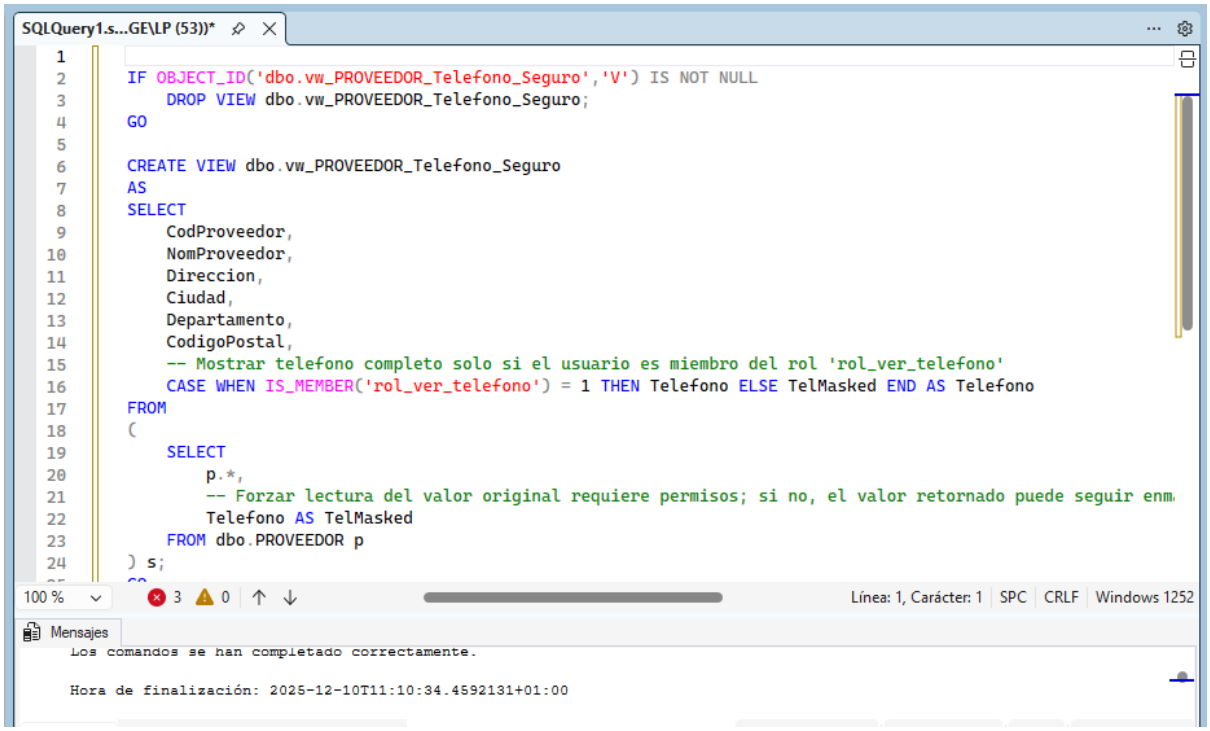


3. Crear vista segura que entrega Teléfono completo sólo si el usuario es miembro de `rol_ver_telefono`:

```
-- Vista que usa IS_MEMBER para validar rol
IF
OBJECT_ID('dbo.vw_PROVEEDOR_Telefono_Seguro','V')
IS NOT NULL
    DROP VIEW dbo.vw_PROVEEDOR_Telefono_Seguro;
GO

CREATE VIEW dbo.vw_PROVEEDOR_Telefono_Seguro
AS
SELECT
```

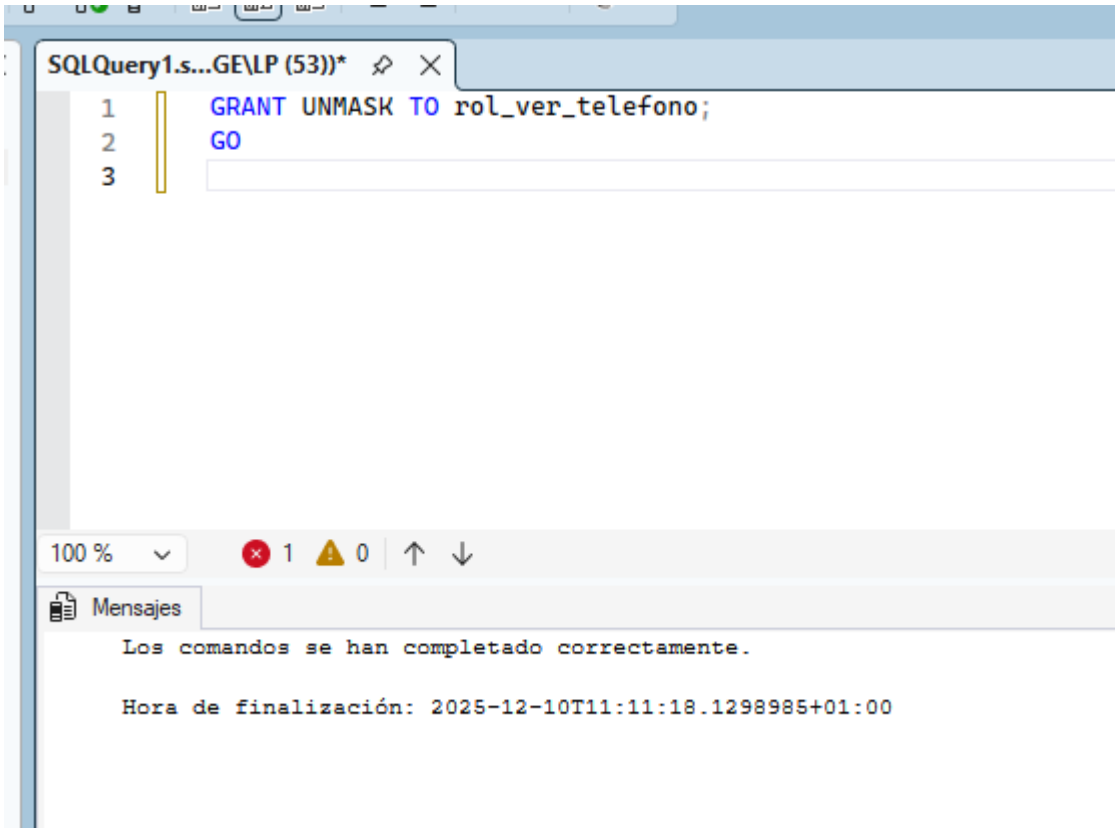
```
    CodProveedor,
    NomProveedor,
    Direccion,
    Ciudad,
    Departamento,
    CodigoPostal,
    -- Mostrar telefono completo solo si el usuario
es miembro del rol 'rol_ver_telefono'
    CASE WHEN IS_MEMBER('rol_ver_telefono') = 1
THEN Telefono ELSE TelMasked END AS Telefono
FROM
(
    SELECT
        p.*,
        -- Forzar lectura del valor original
requiere permisos; si no, el valor retornado puede
seguir enmascarado
        Telefono AS TelMasked
    FROM dbo.PROVEEDOR p
) s;
```

GO



Nota: Dynamic Data Masking es aplicado en servidor; si el usuario tiene UNMASK permission or es miembro de db_owner, verá los datos sin máscara. Para dar permiso UNMASK:

```
GRANT UNMASK TO rol_ver_telefono;
GO
```



Explicación

- `MASKED WITH` aplica una máscara a la columna para usuarios sin permisos `UNMASK`.
- `IS_MEMBER('rol_ver_telefono') = 1` permite diferenciar en tiempo de ejecución si el usuario actual pertenece al rol que debería ver datos completos; alternativamente se puede otorgar `UNMASK` al rol.

Buenas prácticas

- No confiar sólo en DDM para seguridad fuerte — es una capa adicional para reducir exposición accidental.
- Auditar quién recibe `UNMASK`.
- Usar vistas controladas y procedimientos almacenados para exponer datos críticos.

- Evitar dar db_owner a usuarios que no deben ver datos completos.

10 Capstone: Integración (roles, TDE, Always Encrypted, auditoría)

Enunciado

Proyecto integrador: crear un rol auditor_seguridad, habilitar TDE (si no está), preparar Always Encrypted para una columna sensible (ej. PrecioProveedor_ENC), configurar auditoría de accesos a esa tabla y dejar un procedimiento almacenado que registre cambios críticos; la traza será soportada por el audit.

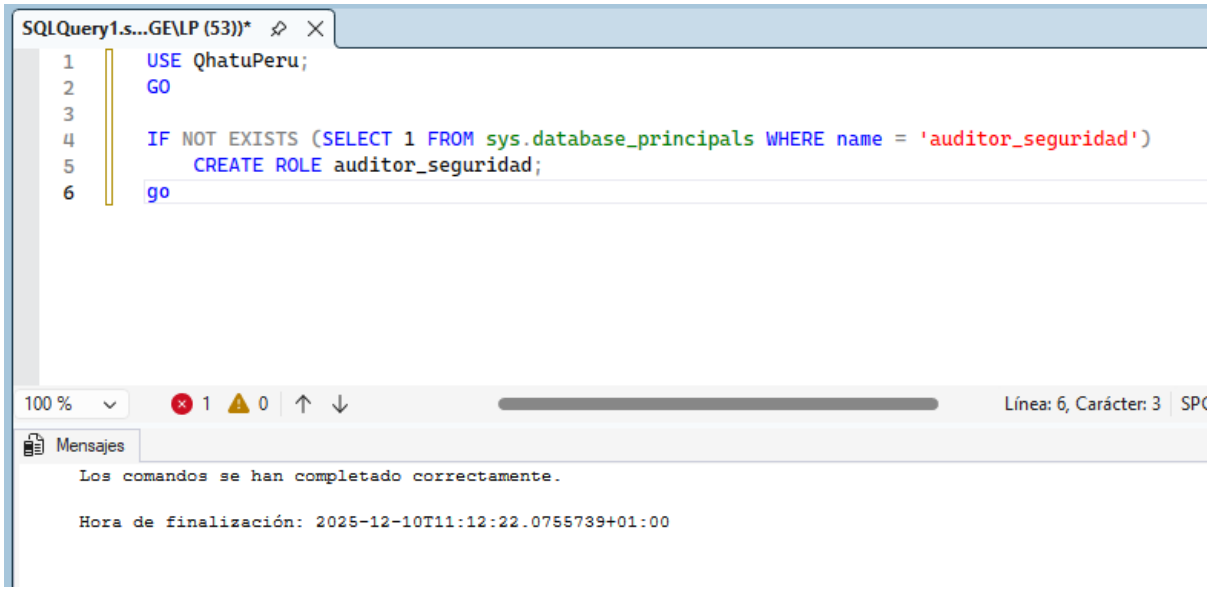
Objetivo

Demostrar la integración de controles: cifrado en reposo (TDE), cifrado a nivel de columna (Always Encrypted), control de acceso (roles/deny/mask) y auditoría centralizada.

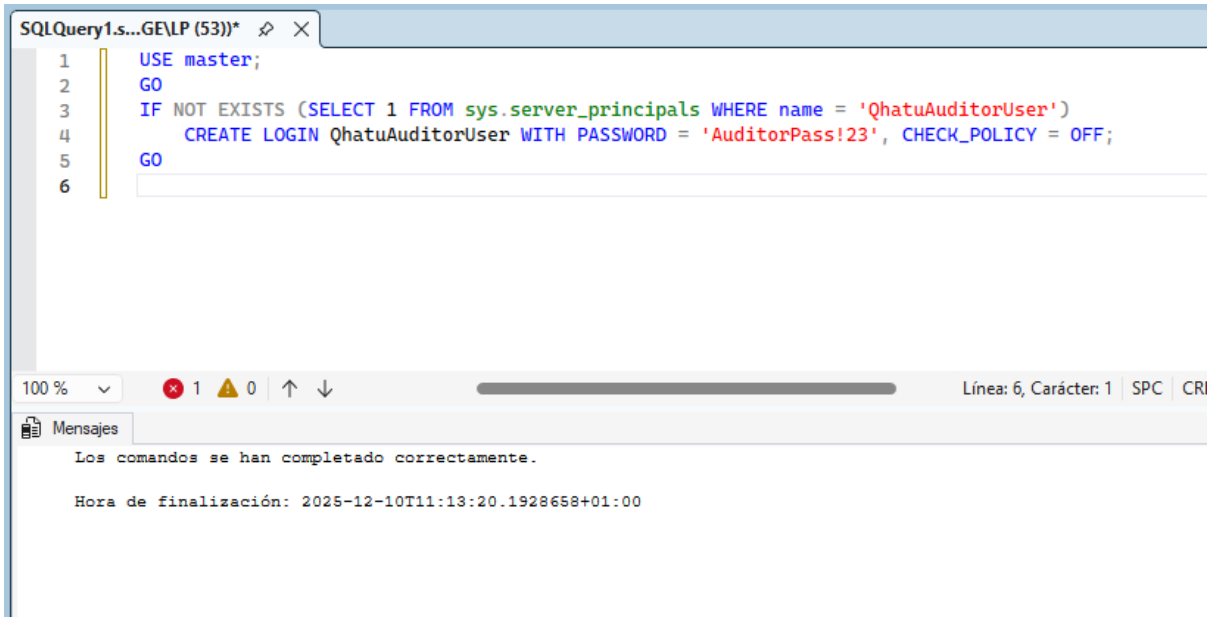
Paso a paso (con código)

1. Crear rol auditor_seguridad y usuario de auditor:

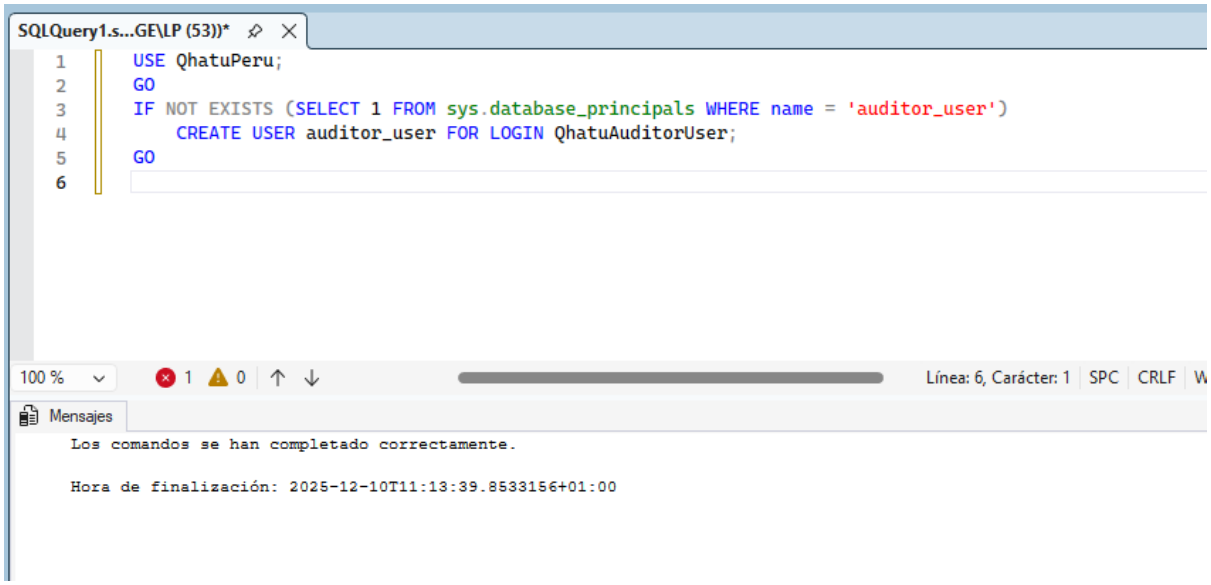
```
USE QhatuPeru;
GO

IF NOT EXISTS (SELECT 1 FROM
sys.database_principals WHERE name =
'auditor_seguridad')
    CREATE ROLE auditor_seguridad;
GO
```



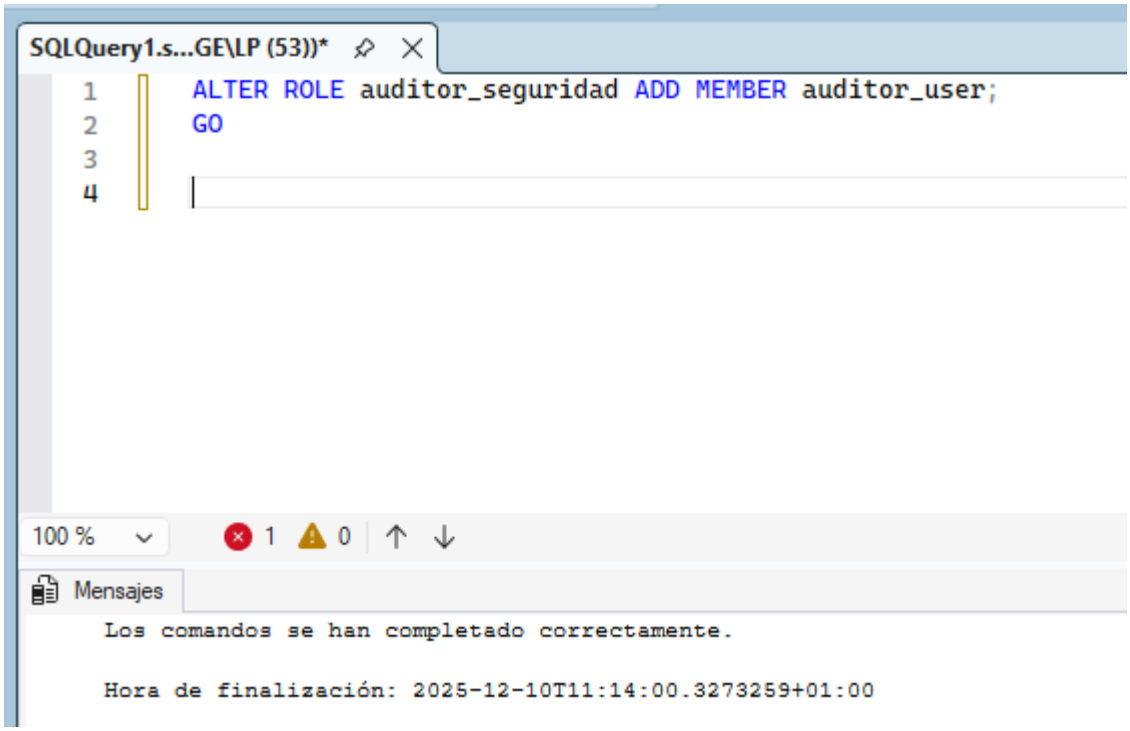

```
-- Usuario local para auditoría (vinculado a login existente o crear otro)
-- Asumo login 'QhatuAuditorUser' ya creado en master; si no, crearlo (ejemplo)
USE master;
GO
IF NOT EXISTS (SELECT 1 FROM sys.server_principals WHERE name = 'QhatuAuditorUser')
    CREATE LOGIN QhatuAuditorUser WITH PASSWORD = 'AuditorPass!23', CHECK_POLICY = OFF;
GO
```




```
USE QhatuPeru;
GO
IF NOT EXISTS (SELECT 1 FROM
sys.database_principals WHERE name =
'auditor_user')
    CREATE USER auditor_user FOR LOGIN
QhatuAuditorUser;
GO
```



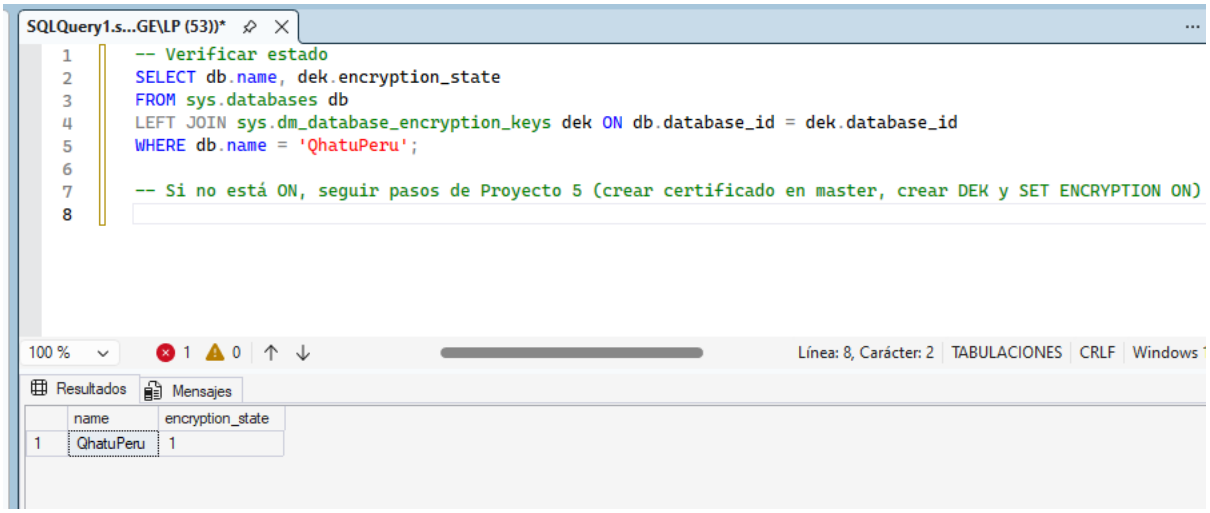

```
ALTER ROLE auditor_seguridad ADD MEMBER
auditor_user;
GO
```



2. Asegurar TDE (ejecutar pasos de Proyecto 5 si no está habilitado). (Resumen:)

```
-- Verificar estado
SELECT db.name, dek.encryption_state
FROM sys.databases db
LEFT JOIN sys.dm_database_encryption_keys dek ON
db.database_id = dek.database_id
WHERE db.name = 'QhatuPeru';

-- Si no está ON, seguir pasos de Proyecto 5 (crear
certificado en master, crear DEK y SET ENCRYPTION
```

ON)



3. Preparar Always Encrypted (crear PrecioProveedor_ENC) — resumen de pasos:

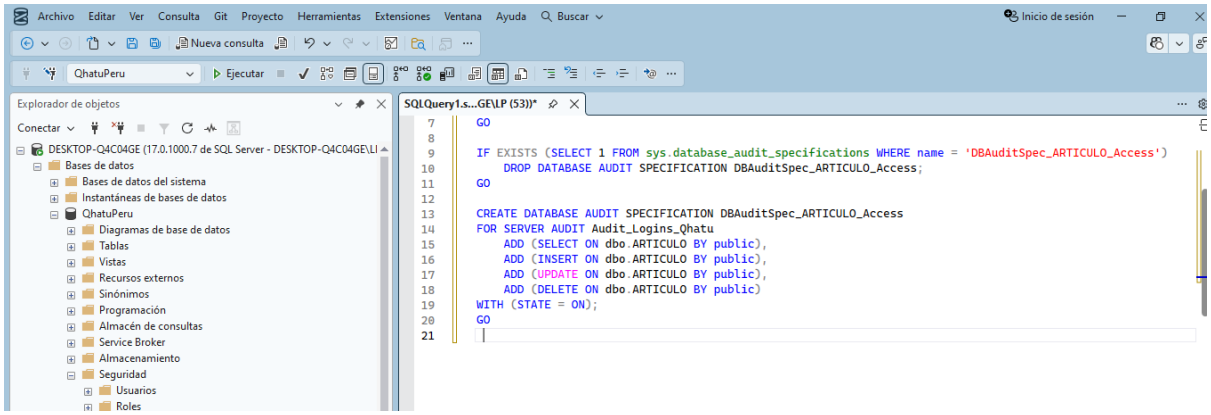
```
-- Crear CMK y CEK según Proyecto 6, agregar
columna PrecioProveedor_ENC a ARTICULO
ALTER TABLE dbo.ARTICULO
ADD PrecioProveedor_ENC decimal(18,2) COLLATE
Latin1_General_BIN2
    ENCRYPTED WITH (
        COLUMN_ENCRYPTION_KEY = CEK_QhatuPeru,
        ENCRYPTION_TYPE = DETERMINISTIC,
        ALGORITHM = 'AEAD_AES_256_CBC_HMAC_SHA_256'
    ) NULL;
GO
```

(Migrar datos desde PrecioProveedor a PrecioProveedor_ENC desde cliente).

4. Configurar auditoría para accesos a la tabla ARTICULO (SELECT/INSERT/UPDATE/DELETE):

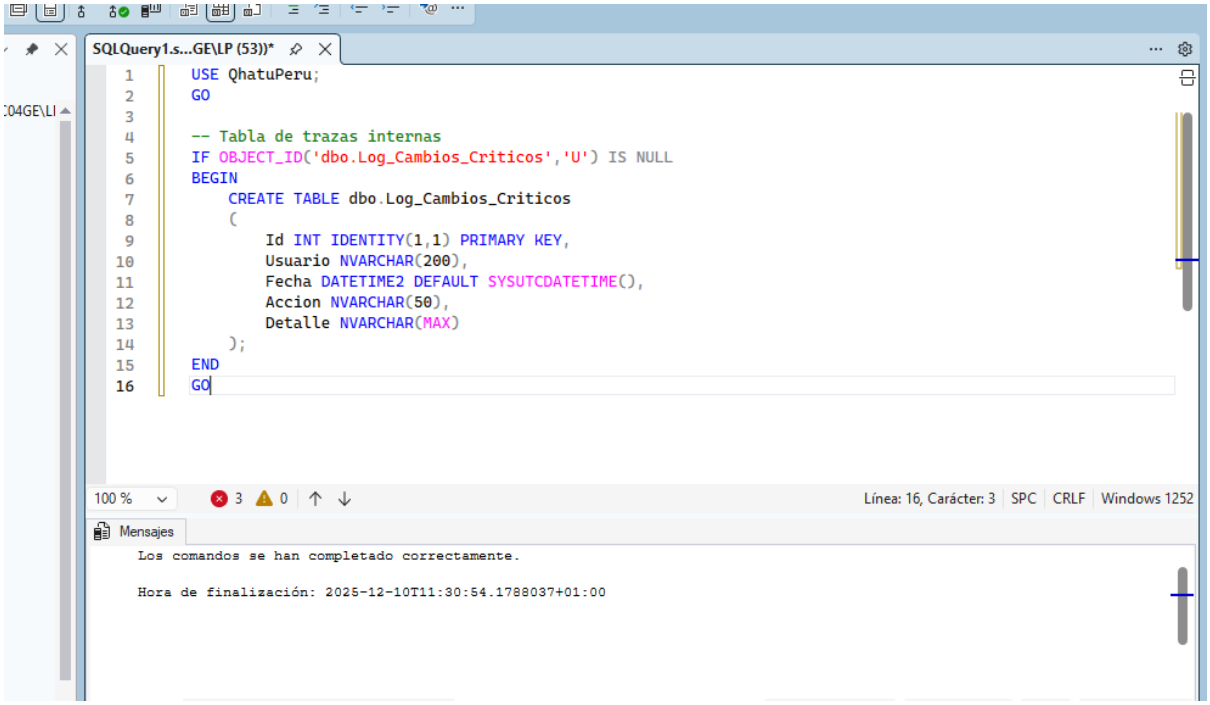
```
USE master;
GO
-- Usar el Server Audit 'Audit_Logins_Qhatu' del
Proyecto 7 o crear uno nuevo

-- Crear Database Audit Specification para capturar
operaciones sobre ARTICULO
USE QhatuPeru;
GO

IF EXISTS (SELECT 1 FROM
sys.database_audit_specifications WHERE name =
'DBAuditSpec_ARTICULO_Access')
    DROP DATABASE AUDIT SPECIFICATION
DBAuditSpec_ARTICULO_Access;
GO

CREATE DATABASE AUDIT SPECIFICATION
DBAuditSpec_ARTICULO_Access
FOR SERVER AUDIT Audit_Logins_Qhatu
    ADD (SELECT ON dbo.ARTICULO BY public),
    ADD (INSERT ON dbo.ARTICULO BY public),
    ADD (UPDATE ON dbo.ARTICULO BY public),
    ADD (DELETE ON dbo.ARTICULO BY public)
WITH (STATE = ON);
```

GO



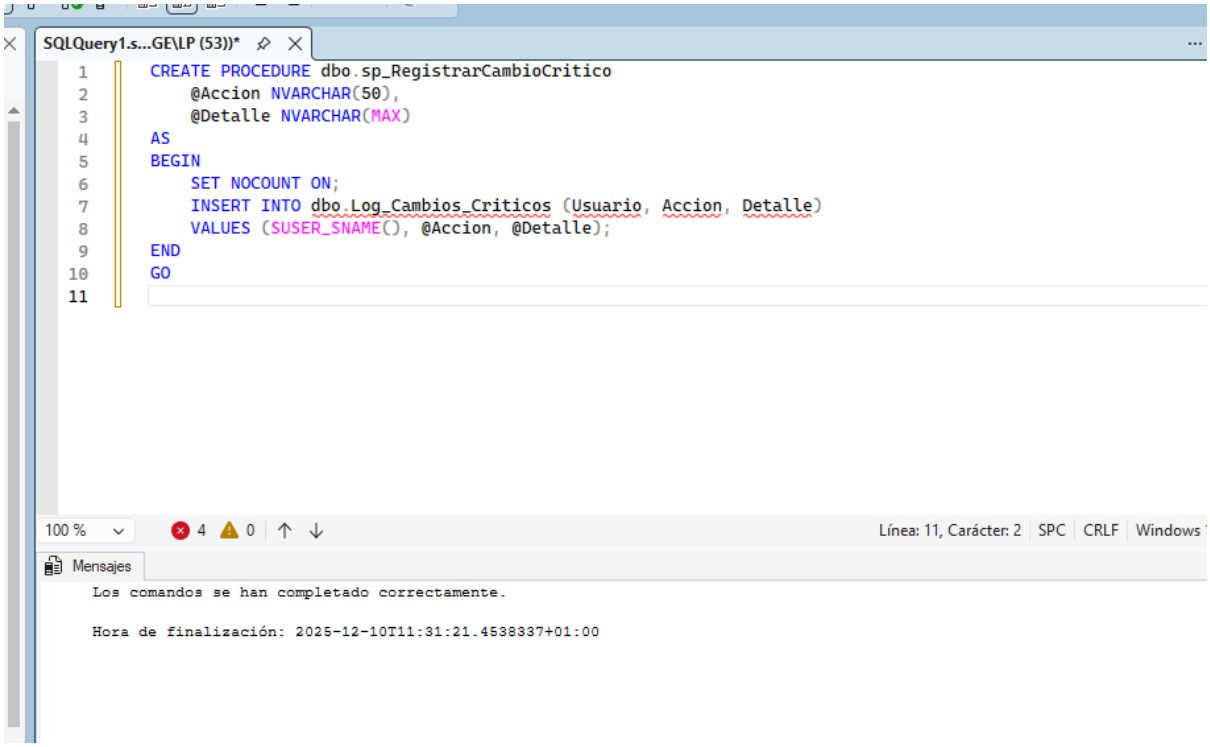
5. Procedimiento almacenado para registrar cambios críticos (usa tabla de log interna y además el Audit grabará el evento):

```
USE QhatuPeru;
GO

-- Tabla de trazas internas
IF OBJECT_ID('dbo.Log_Cambios_Criticos','U') IS
NULL
BEGIN
    CREATE TABLE dbo.Log_Cambios_Criticos
    (
        Id INT IDENTITY(1,1) PRIMARY KEY,
        Usuario NVARCHAR(200),
        Fecha DATETIME2 DEFAULT SYSUTCDATETIME(),
        Accion NVARCHAR(50),
        Detalle NVARCHAR(MAX)
    );
END
GO
```




```
-- Procedimiento que registra cambios críticos y puede ser llamado por triggers o procesos administrativos
CREATE PROCEDURE dbo.sp_RegistrarCambioCritico
    @Accion NVARCHAR(50),
    @Detalle NVARCHAR(MAX)
AS
BEGIN
    SET NOCOUNT ON;
    INSERT INTO dbo.Log_Cambios_Criticos (Usuario, Accion, Detalle)
    VALUES (SUSER_SNAME(), @Accion, @Detalle);
END
```

GO



Explicación

- Combina TDE (protección de archivos), Always Encrypted (protección de datos sensibles en columnas) y Auditoría (registro de accesos y cambios).
- El procedimiento `sp_RegistrarCambioCritico` ofrece un registro interno adicional; la auditoría de SQL Server crea registros independientes en el archivo de auditoría.

Buenas prácticas

- Mantener separados roles de administración y auditoría.
- Proteger y respaldar claves/certificados.
- Probar restauración de backups en servidores diferentes (con certificados cargados).

- Configurar retención/rotación de logs de auditoría y revisar alertas periódicamente.
- No incluir claves/contraseñas en código fuente.