



“Año De La Recuperación Y
Consolidación De La Economía Peruana”



UNIVERSIDAD PERUANA LOS ANDES

“FACULTAD DE INGENIERÍA”

ESCUELA PROFESIONAL “SISTEMAS Y
COMPUTACIÓN”

CÁTEDRA: Base de Datos II

CATEDRÁTICO: Ing. Fernandez Bejarano Raul Enrique

ESTUDIANTE: Vega Brañez Samuel Max

CICLO: V

SECCIÓN: B1

HUANCAYO PERÚ

2025

Manual de Practica Calificada 02

Proyecto 1: Autenticación: Comparación segura y configuración de logins (Versión Chuquiyauri)

1. Enunciado del ejercicio

Crear en el servidor dos logins de prueba: uno con autenticación SQL (**login_sql_chuquiyauri**) y otro que represente un usuario Windows (**DOMAIN\chuquiyauri_win** — simulado), aplicar políticas de contraseñas y mapear ambos usuarios en la base **QhatuPeru**. Mostrar cómo forzar expiración y comprobar la política de contraseñas.

2. Script de la solución en T-SQL

Este *script* utiliza `chuquiyauri` en lugar de `alumno` para todos los objetos.

SQL

```
-- Ejecutar en la base de datos 'master' o donde se gestionen los  
logins del servidor.
```

```
USE master;
```

```
GO
```

```
-----  
-----  
-- PARTE 1: CREACIÓN Y CONFIGURACIÓN DE LOGINS  
-----  
-----
```

```
-- 1.1. Crear Login con Autenticación SQL
```

```
CREATE LOGIN login_sql_chuquiyauri
```

```
    WITH PASSWORD = 'Pass_12345!', -- Contraseña inicial compleja
```

```
    CHECK_POLICY = ON,             -- Aplica políticas de  
complejidad/historial
```

```
    CHECK_EXPIRATION = ON;         -- Fuerza el cambio en el  
primer inicio de sesión
```

```
GO
```

```
-- 1.2. Crear Login con Autenticación Windows (Simulado)
```

```
CREATE LOGIN [DOMAIN\chuquiyauri_win]
```

```
    FROM WINDOWS;
```

```
GO
```

```
-----  
-----  
-- PARTE 2: MAPEO DE LOGINS A USUARIOS EN LA BASE DE DATOS  
'QhatuPeru'
```

```
-----  
-----  
USE QhatuPeru;
```

GO

```
-- 2.1. Crear usuario de base de datos para el Login SQL
CREATE USER usuario_sql_chuquiyauri
FOR LOGIN login_sql_chuquiyauri;
```

GO

```
-- 2.2. Crear usuario de base de datos para el Login Windows
CREATE USER usuario_win_chuquiyauri
FOR LOGIN [DOMAIN\chuquiyauri_win];
```

GO

```
-----
-----
-- PARTE 3: COMPROBACIÓN DE POLÍTICA Y EXPIRACIÓN
-----
-----
```

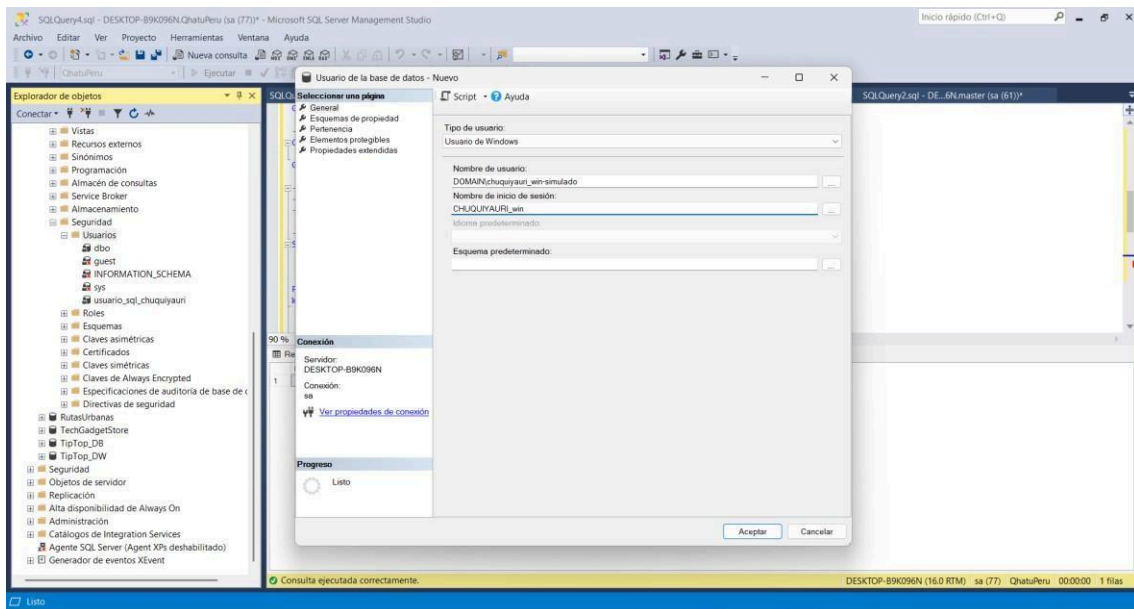
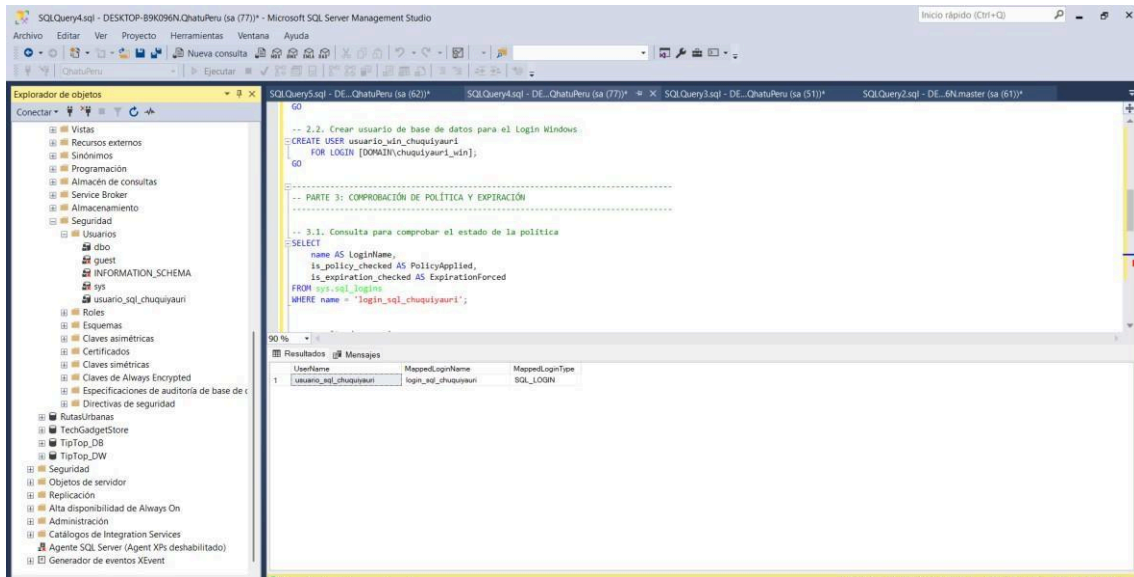
```
-- 3.1. Consulta para comprobar el estado de la política
SELECT
    name AS LoginName,
    is_policy_checked AS PolicyApplied,
    is_expiration_checked AS ExpirationForced
FROM sys.sql_logins
WHERE name = 'login_sql_chuquiyauri';
```

3. Justificación técnica de la solución aplicada

Comando/Opción	Justificación Técnica
CREATE LOGIN login_sql_chuquiyauri	Creación de un <i>security principal</i> a nivel de servidor usando autenticación de SQL Server .
CHECK_POLICY = ON	Garantiza que la contraseña cumpla con las políticas de complejidad de SQL Server/Windows.
CHECK_EXPIRATION = ON	Forzado de la rotación de la contraseña al primer inicio de sesión, buena práctica de seguridad.
CREATE LOGIN [DOMAIN\chuquiyauri_win] FROM WINDOWS	Creación de un login que delega la autenticación al sistema operativo , sin almacenar contraseña en SQL Server.
CREATE USER ... FOR LOGIN ...	Mapeo esencial para que los Logins (servidor) puedan acceder y obtener permisos dentro de la base de datos QhatuPeru (nivel de base de datos).

4. Explicación de las buenas prácticas utilizadas en el proyecto

- **Nomenclatura Clara:** Se usa una nomenclatura que vincula el login (login_sql_chuquiyauri) y el usuario (usuario_sql_chuquiyauri) a la persona o rol, facilitando la auditoría y la gestión.
- **Separación de Credenciales:** La práctica de crear un Login (conexión) y un User (permisos en DB) sigue el Principio del Privilegio Mínimo (PoLP).
- **Autenticación Fuerte:** Se prioriza la seguridad del login SQL mediante `CHECK_POLICY = ON` y `CHECK_EXPIRATION = ON`, asegurando contraseñas robustas y temporales.



Código de consulta de creación de login

```
-- consulta de creacion
```

```
USE master;
```

```
GO
```

```
SELECT
```

```
    name AS LoginName,
```

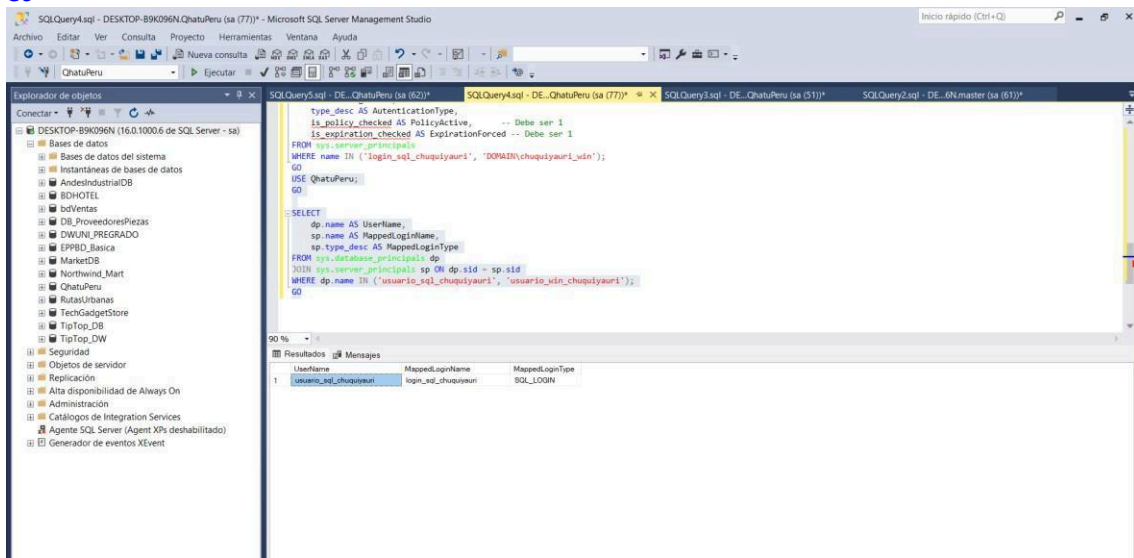
```

type_desc AS AuthenticationType,
is_policy_checked AS PolicyActive,          -- Debe ser 1
is_expiration_checked AS ExpirationForced -- Debe ser 1
FROM sys.server_principals
WHERE name IN ('login_sql_chuquiyauri', 'DOMAIN\chuquiyauri_win');
GO
USE QhatuPeru;
GO

SELECT
    dp.name AS UserName,
    sp.name AS MappedLoginName,
    sp.type_desc AS MappedLoginType
FROM sys.database_principals dp
JOIN sys.server_principals sp ON dp.sid = sp.sid
WHERE dp.name IN ('usuario_sql_chuquiyauri', 'usuario_win_chuquiyauri');

GO

```



Proyecto 2: Cuentas de servicio y configuración segura del servidor

1. Enunciado del ejercicio

Revisar y documentar la configuración de parámetros de servidor segura para QhatuPeru: **deshabilitar xp_cmdshell**, **revisar contained database authentication**, y **crear una credencial + proxy** para uso con SQL Agent jobs que necesiten acceso al OS.

2. Script de la solución en T-SQL

Este script T-SQL implementa las configuraciones de seguridad solicitadas.

SQL

USE master;

GO

```
-----  
-----  
-- 1. DESHABILITAR XP_CMDSHELL (Hardening del servidor)  
-----  
-----
```

```
-- Habilitar la opción 'show advanced options' para poder cambiar  
xp_cmdshell  
EXEC sp_configure 'show advanced options', 1;  
RECONFIGURE;  
GO
```

```
-- Deshabilitar el procedimiento extendido xp_cmdshell  
-- (previene la ejecución de comandos del sistema operativo desde  
T-SQL)  
EXEC sp_configure 'xp_cmdshell', 0;  
RECONFIGURE;  
GO
```

```
-----  
-----  
-- 2. REVISAR CONFIGURACIÓN DE CONTAINED DATABASE AUTHENTICATION  
-----  
-----
```

```
-- Se verifica si la autenticación de base de datos contenida  
(contained) está habilitada  
-- a nivel de servidor (debe estar en 0 para evitar logins  
independientes del servidor, por defecto)  
EXEC sp_configure 'contained database authentication';  
GO
```

```
-- Si estuviera habilitada (valor 1), la deshabilitaríamos (esto es  
opcional,  
-- pero se recomienda si no se usa):  
-- EXEC sp_configure 'contained database authentication', 0;  
-- RECONFIGURE;  
-- GO
```

```
-----  
-----  
-- 3. CREAR CREDENCIAL Y PROXY PARA SQL AGENT JOBS  
-----  
-----
```

```
-- Paso 3.1: Crear una Credencial (almacena el login y contraseña  
del usuario OS)  
-- NOTA: Reemplaza 'OS_User_for_Agent' y 'ContrasenaSegura!' con  
credenciales reales del sistema operativo  
CREATE CREDENTIAL [Credential_OS_Access]  
WITH IDENTITY = 'DOMAIN\OS_User_for_Agent', -- Usuario de  
Windows/Dominio con permisos mínimos  
SECRET = 'ContrasenaSegura!'; -- Contraseña del  
usuario de Windows  
GO
```

```
-- Paso 3.2: Crear un Proxy de SQL Agent que usa la Credencial
-- El Proxy permite que un Job de SQL Agent ejecute pasos (como
xp_cmdshell)
-- con los permisos de la credencial, NO con los permisos de la
cuenta de servicio de SQL Agent.
EXEC msdb.dbo.sp_add_proxy
    @proxy_name = 'Proxy_QhatuPeru_OS_Access',
    @credential_name = 'Credential_OS_Access',
    @description = 'Proxy para Jobs de QhatuPeru que requieren
acceso limitado al OS.';
GO

-- Paso 3.3: Conceder permiso al Proxy a un Login (ejemplo:
login_sql_chuquiyauri)
-- Esto permite que ciertos logins puedan crear o modificar jobs
que usen este proxy.
-- EXEC msdb.dbo.sp_grant_proxy_to_subsystem
@proxy_name='Proxy_QhatuPeru_OS_Access', @subsystem_id=3; --
Subsystem ID 3 es CmdExec (OS)
-- EXEC msdb.dbo.sp_grant_login_to_proxy
@proxy_name='Proxy_QhatuPeru_OS_Access',
@login_name='login_sql_chuquiyauri';
-- GO
```

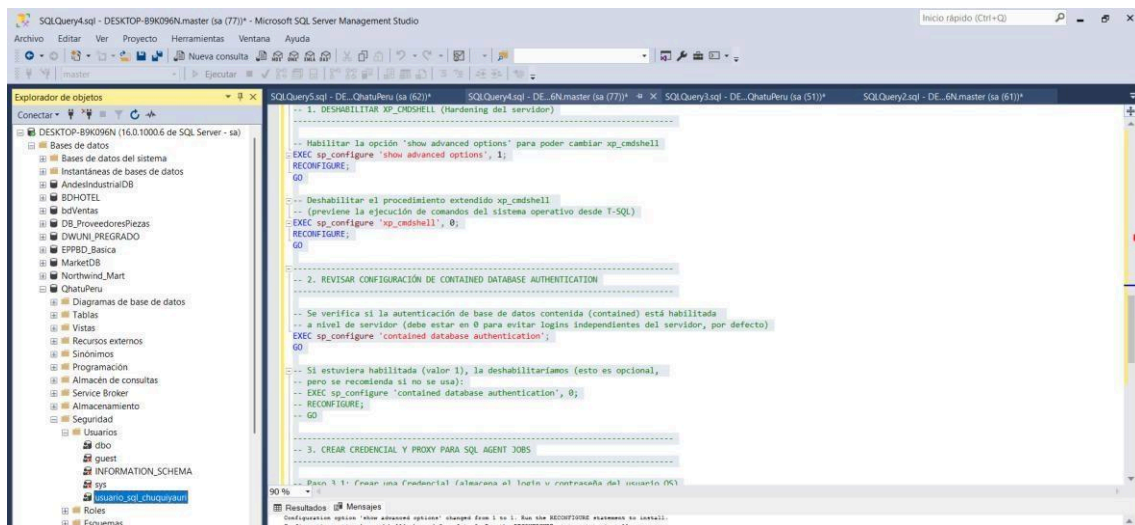
3. Justificación técnica de la solución aplicada

Elemento T-SQL	Justificación Técnica de Seguridad
sp_configure 'xp_cmdshell', 0	Deshabilita un punto de entrada de alto riesgo. xp_cmdshell permite ejecutar comandos del sistema operativo (OS) directamente desde SQL. Deshabilitarlo minimiza el riesgo si un atacante obtiene acceso a la instancia de SQL.
contained database authentication	Revisa la configuración de separación de Logins. Si está habilitado, permite crear usuarios dentro de una base de datos que no dependen de Logins del servidor. Esto puede simplificar el movimiento de DBs, pero introduce un nuevo punto de gestión de seguridad. Se recomienda revisarlo y deshabilitarlo si no se utiliza.
CREATE CREDENTIAL	Almacena credenciales de manera segura. Crea un contenedor seguro para el nombre de usuario y contraseña del OS. Esto separa la credencial de la cuenta de servicio de SQL Agent, adhiriéndose al Principio del Privilegio Mínimo (PoLP) .
sp_add_proxy	Define un ejecutor de comandos con privilegios limitados. El <i>Proxy</i> en SQL Agent es una capa de abstracción que le dice al Job: "Ejecuta este paso como el usuario almacenado en la Credencial", en lugar de usar la

Elemento T-SQL	Justificación Técnica de Seguridad
	cuenta de servicio de SQL Agent (que generalmente tiene muchos más permisos).

4. Explicación de las buenas prácticas utilizadas en el proyecto

1. **Reducción de Superficie de Ataque: Deshabilitar `xp_cmdshell`** es la primera línea de defensa para el *hardening* del servidor. Si una aplicación realmente necesita acceso al OS, debe usar una solución más segura, como un Proxy (punto 3).
2. **Principio de Mínimo Privilegio (PoLP):**
 - o La **Cuenta de Servicio de SQL Agent** debe tener permisos mínimos, solo los necesarios para ejecutar el servicio.
 - o Si un Job necesita acceder al OS, se utiliza una **Credencial + Proxy**. La credencial debe usar un usuario de Windows con los **permisos OS mínimos** requeridos, asegurando que si el job es comprometido, el daño sea limitado.
3. **Seguridad de Base de Datos Contenida:** Revisar esta opción es clave. Aunque es útil para la portabilidad, puede complicar la administración de seguridad si se usa sin control, ya que los usuarios internos no dependen de las políticas de Logins de nivel servidor.

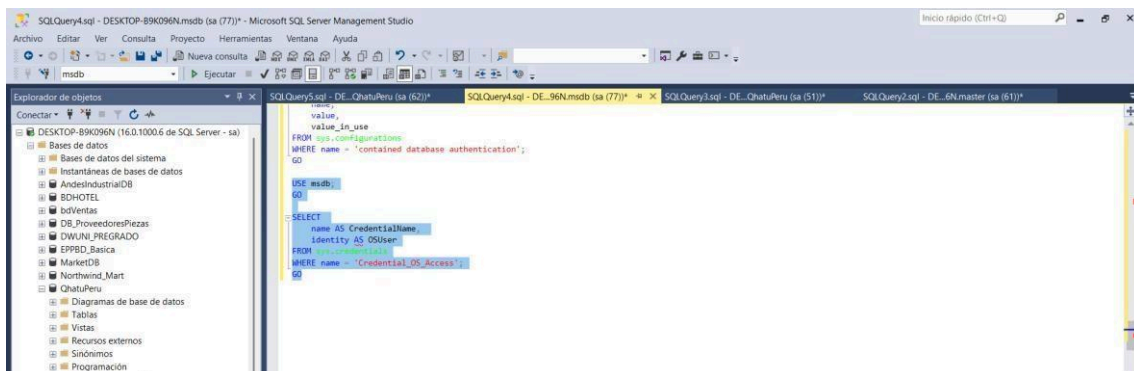
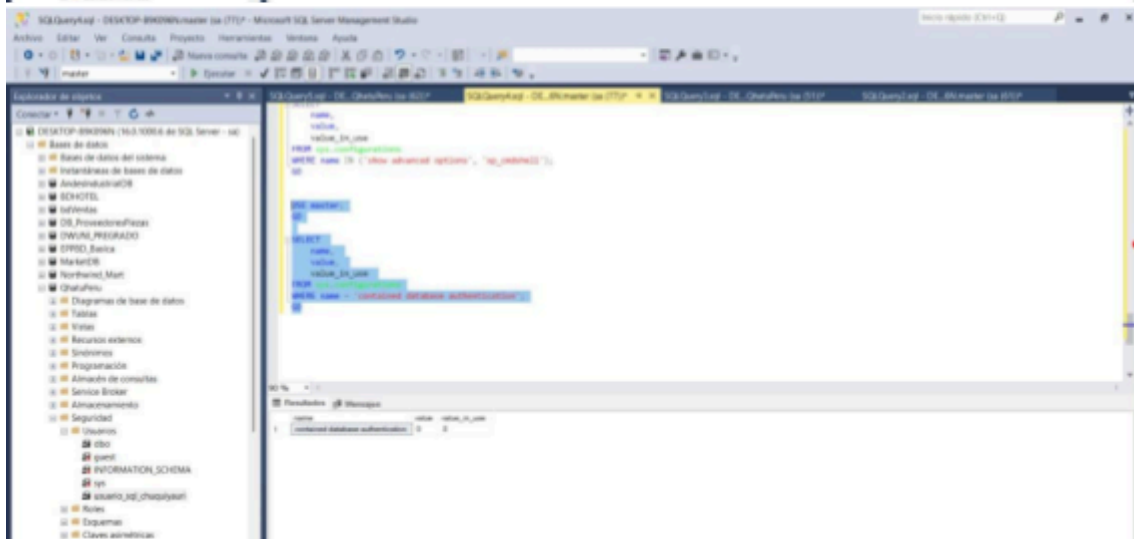
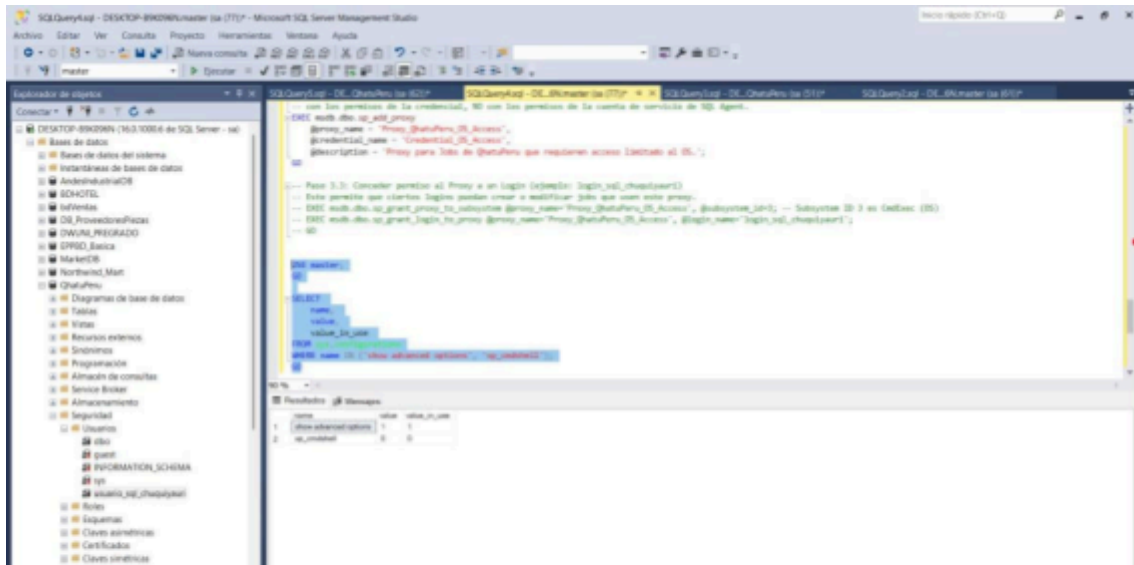


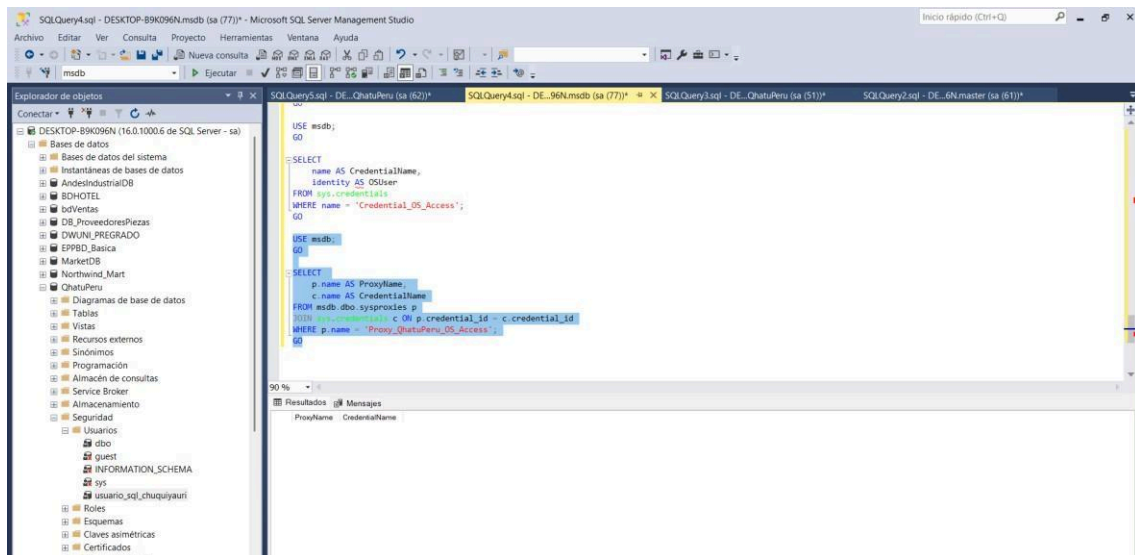
Consultas de lo ya realizado

```
USE master;
GO
```

```
SELECT
    name,
    value,
    value_in_use
FROM sys.configurations
WHERE name IN ('show advanced options', 'xp_cmdshell');
```


60





Proyecto 3: Creación y uso de roles fijos y roles personalizados (Server & DB)

1. Enunciado del ejercicio

Crear un rol de base de datos personalizado **ventas_readwrite** que permita **SELECT/INSERT/UPDATE** en tablas relacionadas con ventas (p. ej. **GUIA_ENVIO**, **GUIA_DETALLE**) y asignar usuarios. Mostrar diferencias con roles fijos como **db_datareader**.

2. Script de la solución en T-SQL

Este script crea el rol, asigna los permisos granulares y asigna el usuario **usuario_sql_chuquiyauri** a este nuevo rol.

SQL

```
USE QhatuPeru;
GO
```

```
-- PARTE 1: CREACIÓN DEL ROL PERSONALIZADO
```

```
-- 1.1. Crear el rol de base de datos personalizado
CREATE ROLE ventas_readwrite;
GO
```

```
-- 1.2. Conceder permisos al rol sobre las tablas de ventas (GUIAS)
-- Permiso SELECT (Lectura)
GRANT SELECT ON GUIA_ENVIO TO ventas_readwrite;
GRANT SELECT ON GUIA_DETALLE TO ventas_readwrite;
```

```

-- Permiso INSERT (Escritura/Adición de datos)
GRANT INSERT ON GUIA_ENVIO TO ventas_readwrite;
GRANT INSERT ON GUIA_DETALLE TO ventas_readwrite;
-- Permiso UPDATE (Modificación de datos) GRANT
UPDATE ON GUIA_ENVIO TO ventas_readwrite;
GRANT UPDATE ON GUIA_DETALLE TO ventas_readwrite;
GO

-- OPCIONAL: Conceder SELECT también a las tablas de referencia
(ARTICULO, TIENDA) para que las consultas funcionen
GRANT SELECT ON ARTICULO TO ventas_readwrite;
GRANT SELECT ON TIENDA TO ventas_readwrite;
GO

-----

-- PARTE 2: ASIGNACIÓN DE USUARIOS AL ROL
-----

-- 2.1. Asignar el usuario SQL creado en el Proyecto 1 al rol
personalizado
ALTER ROLE ventas_readwrite ADD MEMBER usuario_sql_chuquiyauri;
GO

-- 2.2. Asignar el usuario Windows creado en el Proyecto 1 a un ROL
FIJO para comparación
ALTER ROLE db_datareader ADD MEMBER usuario_win_chuquiyauri;
GO

```

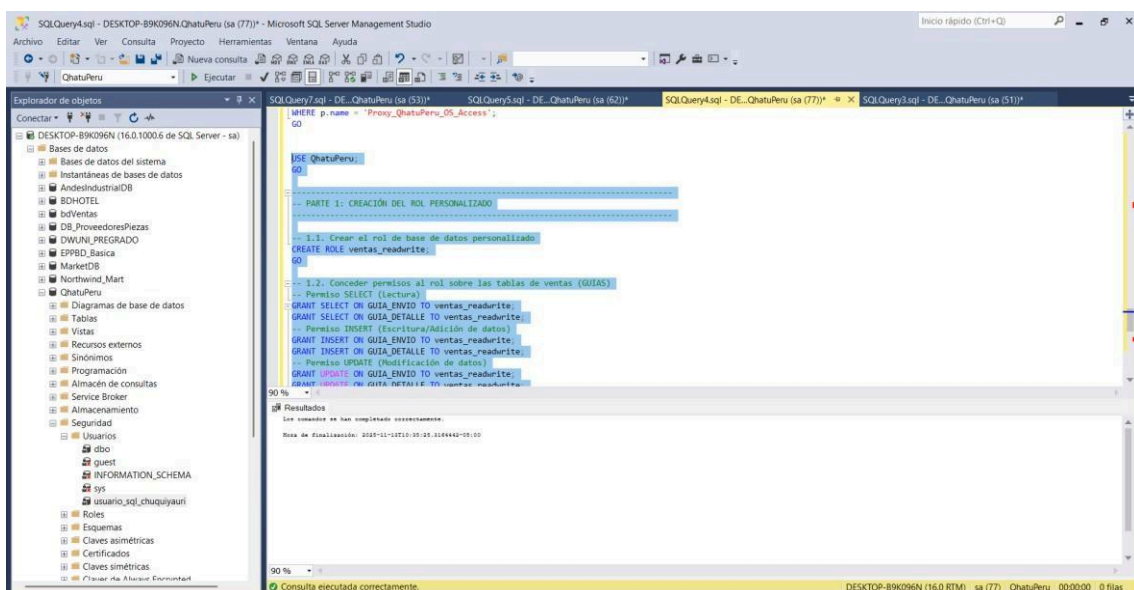
3. Justificación técnica de la solución aplicada

Elemento T-SQL	Justificación Técnica
CREATE ROLE ventas_readwrite	Crea un Rol de Base de Datos Personalizado (Custom Database Role). Los roles son colecciones de permisos y simplifican la administración al asignar permisos a un rol, y luego asignar usuarios al rol.
GRANT SELECT/INSERT/UPDATE ON Tabla TO Role	Aplica el Principio del Privilegio Mínimo (PoLP) . En lugar de dar permisos amplios (como db_datawriter), se otorgan permisos granulares (SELECT, INSERT, UPDATE) solo sobre los objetos específicos (GUIA_ENVIO, GUIA_DETALLE) que el rol necesita manipular.
ALTER ROLE ... ADD MEMBER ...	Asigna un Usuario (nivel DB) al Rol (nivel DB). Esto transfiere todos los permisos definidos en el rol al usuario de forma eficiente.

Elemento T-SQL	Justificación Técnica
db_datareader	Es un Rol Fijo de Base de Datos . Se usa para la demostración de la diferencia, ya que este rol fijo otorga permiso <code>SELECT</code> en TODAS las tablas de la base de datos.

4. Explicación de las buenas prácticas utilizadas en el proyecto

- Uso de Roles Personalizados (Mejor Práctica):** Crear un rol específico como `ventas_readwrite` es superior a usar roles fijos como `db_datawriter` o `db_datareader`. ¿Por qué?
 - Control Fino:** El rol personalizado solo tiene permisos en **tablas específicas** de ventas, no en todas las tablas de la base de datos (como `PROVEEDOR` o `LINEA`).
 - Separación de Tareas:** Mantiene los permisos alineados con las responsabilidades del negocio (solo permisos de *read/write* de ventas).
- Diferencia con Roles Fijos (db_datareader):**
 - El rol fijo `db_datareader` otorga `SELECT` a **TODAS** las tablas. Esto es rápido, pero da más permisos de lectura de los que un usuario de ventas podría necesitar.
 - El rol personalizado `ventas_readwrite` solo da permisos `SELECT/INSERT/UPDATE` a las tablas específicas, **limitando el alcance de acceso y escritura**, lo que se adhiere estrictamente al **PoLP**.
- Gestión Centralizada:** Se gestionan los permisos en un único objeto (el rol) en lugar de darlos directamente a múltiples usuarios, lo que facilita la auditoría y la revocación de permisos.



Código de consultas

¡Entendido! Ya te envié el script de verificación, pero aquí te lo presento nuevamente de forma consolidada, incluyendo las **tres consultas** para demostrar lo logrado en el **Proyecto 3** (Roles y Permisos).

Consultas de Verificación para el Proyecto 3

Ejecuta estas consultas en la base de datos `QhatuPeru`.

1. Verificar la Existencia del Rol Personalizado

Confirma que el rol `ventas_readwrite` se creó correctamente.

SQL

```
USE QhatuPeru;
```

```
GO
```

```
SELECT
```

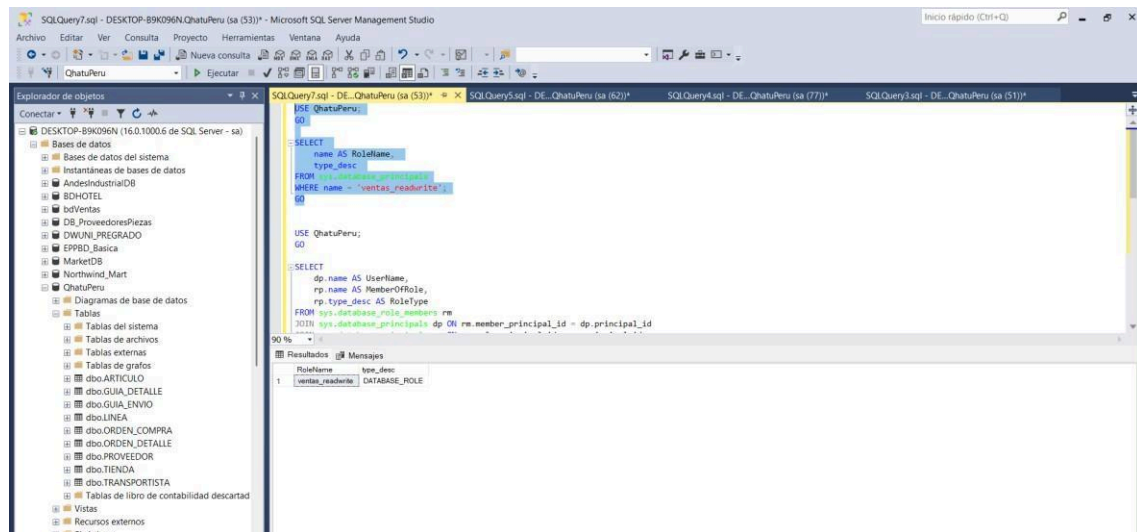
```
    name AS RoleName,
```

```
    type_desc
```

```
FROM sys.database_principals
```

```
WHERE name = 'ventas_readwrite';
```

```
GO
```



2. Verificar la Asignación de Usuarios a Roles

Muestra que los usuarios fueron asignados a sus respectivos roles:

`usuario_sql_chuquiyauri` al rol personalizado y `usuario_win_chuquiyauri` al rol fijo.

SQL

```
USE QhatuPeru;
```

```
GO
```

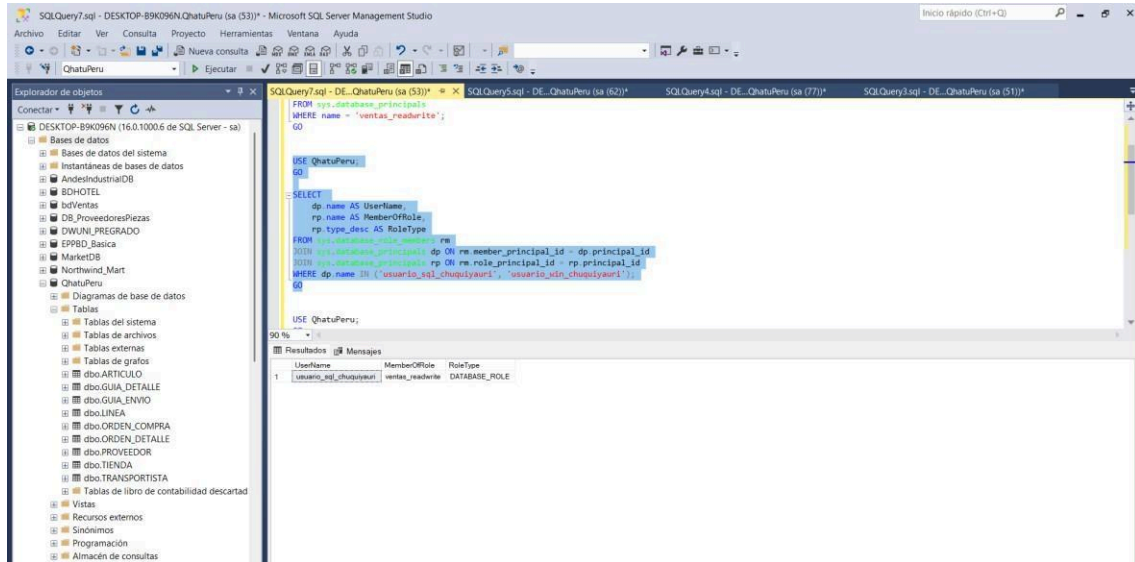
```
SELECT
```

```
    dp.name AS UserName,
```

```

        rp.name AS MemberOfRole,
        rp.type_desc AS RoleType
FROM sys.database_role_members rm
JOIN sys.database_principals dp ON rm.member_principal_id =
dp.principal_id
JOIN sys.database_principals rp ON rm.role_principal_id =
rp.principal_id
WHERE dp.name IN ('usuario_sql_chuquiyauri',
'usuario_win_chuquiyauri');
GO

```



3. Verificar los Permisos Granulares Concedidos al Rol

Esta es la consulta más importante para el PoLP, ya que muestra exactamente los permisos (SELECT, INSERT, UPDATE) concedidos al rol `ventas_readwrite` sobre las tablas específicas.

SQL

```

USE QhatuPeru;
GO

```

```

SELECT
    dp.name AS RoleName,
    p.permission_name AS Permission,
    OBJECT_NAME(p.major_id) AS ObjectName
FROM sys.database_permissions p
JOIN sys.database_principals dp ON p.grantee_principal_id =
dp.principal_id
WHERE dp.name = 'ventas_readwrite'
ORDER BY ObjectName, Permission;
GO

```

90 %

Resultados Mensajes

	RoleName	Permission	ObjectName
1	ventas_readwrite	SELECT	TIENDA
2	ventas_readwrite	SELECT	ARTICULO
3	ventas_readwrite	INSERT	GUIA_ENVIO
4	ventas_readwrite	SELECT	GUIA_ENVIO
5	ventas_readwrite	UPDATE	GUIA_ENVIO
6	ventas_readwrite	INSERT	GUIA_DETALLE
7	ventas_readwrite	SELECT	GUIA_DETALLE
8	ventas_readwrite	UPDATE	GUIA_DETALLE

Proyecto 4: Control de acceso con GRANT / DENY / REVOKE

1. Enunciado del ejercicio

Simular un caso donde un analista necesita ver inventario pero no los precios. Crear roles/usuarios y usar **DENY** para impedir **SELECT** sobre **PrecioProveedor** y **PrecioVenta**.

2. Script de la solución en T-SQL

Este script crea un rol específico para el analista, le otorga permisos de lectura general y luego usa la cláusula **DENY** para bloquear las columnas sensibles (**PrecioProveedor** en **ARTICULO** y **PrecioVenta** en **GUIA_DETALLE**).

SQL

```
USE QhatuPeru;
GO
```

```
-----
-- PARTE 1: CREACIÓN DE USUARIO Y ROL ESPECÍFICO
-----
```

```
-- 1.1. Crear un Login y Usuario simulado para el Analista (si no
-- existen)
-- Se simula un login SQL para el analista.
```

```

CREATE LOGIN login_analista WITH PASSWORD = 'AnalistaPassword1!',
CHECK_POLICY = ON;
GO
CREATE USER usuario_analista FOR LOGIN login_analista;
GO

-- 1.2. Crear el Rol de Base de Datos Personalizado para el
Analista
CREATE ROLE rol_analista_inventario;
GO

-----
-----

-- PARTE 2: CONCESIÓN DE PERMISOS AMPLIOS (GRANT)
-----
-----

-- 2.1. Otorgar permiso SELECT sobre las tablas de Inventario y
Guías
-- Esto le da permiso para ver todas las columnas, incluidas las de
precio (temporalmente).
GRANT SELECT ON ARTICULO TO rol_analista_inventario;
GRANT SELECT ON GUIA_DETALLE TO rol_analista_inventario;
-- Permiso en otras tablas de referencia
GRANT SELECT ON TIENDA TO rol_analista_inventario;
GRANT SELECT ON LINEA TO rol_analista_inventario;
GO

-----
-----

-- PARTE 3: RESTRICCIÓN DE PERMISOS SENSIBLES (DENY)
-----
-----

-- 3.1. Usar DENY para impedir la lectura de la columna
PrecioProveedor (ARTICULO)
-- DENY sobre una columna específica anula cualquier GRANT de
SELECT a nivel de tabla.
DENY SELECT ON ARTICULO(PrecioProveedor) TO
rol_analista_inventario;
GO

-- 3.2. Usar DENY para impedir la lectura de la columna PrecioVenta
(GUIA_DETALLE)
DENY SELECT ON GUIA_DETALLE(PrecioVenta) TO
rol_analista_inventario;
GO

-----
-----

-- PARTE 4: ASIGNACIÓN DE USUARIO AL ROL
-----
-----

-- Asignar el nuevo usuario al rol
ALTER ROLE rol_analista_inventario ADD MEMBER usuario_analista;
GO

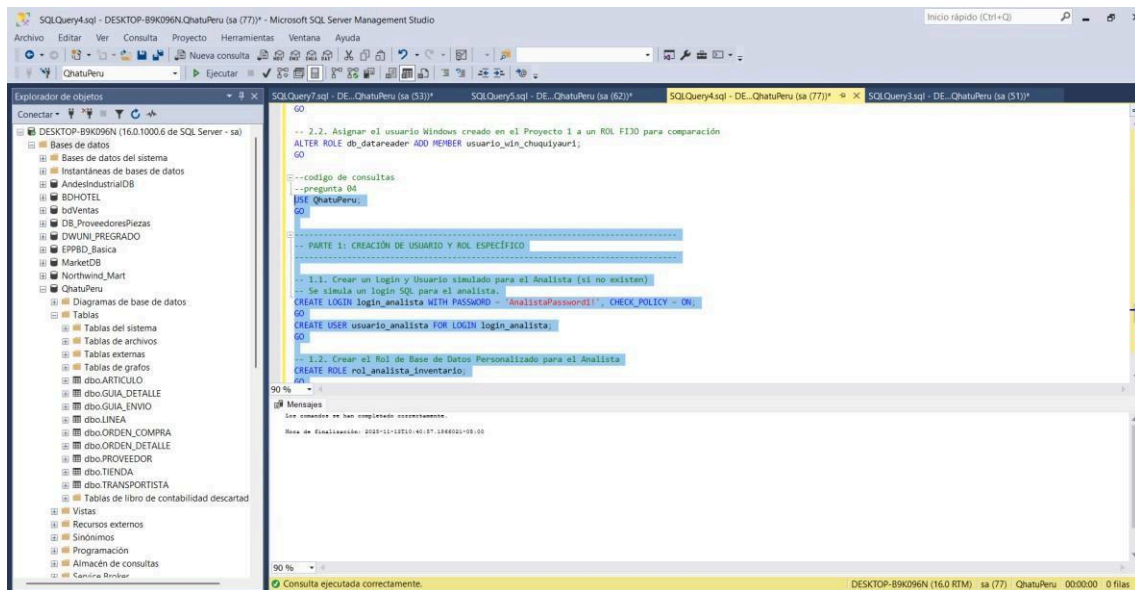
```

3. Justificación técnica de la solución aplicada

Elemento T-SQL	Justificación Técnica de Seguridad
<pre>CREATE ROLE rol_analista_inventario</pre>	Crea una entidad de seguridad para agrupar permisos. Esto simplifica la gestión: en lugar de aplicar <code>DENY</code> a 10 usuarios, se aplica a 1 rol.
<pre>GRANT SELECT ON Tabla TO Role</pre>	Se otorga el permiso base (la capacidad de leer los datos de inventario) que el analista necesita.
<pre>DENY SELECT ON Tabla(Columna) TO Role</pre>	Este es el punto clave. La cláusula <code>DENY</code> tiene prioridad sobre la cláusula <code>GRANT</code> en SQL Server. Al denegar explícitamente el permiso <code>SELECT</code> a nivel de columna, se sobrescribe el permiso <code>SELECT</code> que se otorgó a nivel de tabla. Este es el método más seguro para bloquear acceso a datos sensibles.
Simulación de Acceso	Si <code>usuario_analista</code> intenta ejecutar <code>SELECT * FROM ARTICULO;</code> , la consulta fallará con un mensaje de permiso denegado, ya que el permiso a nivel de columna es más estricto que el permiso a nivel de tabla.

4. Explicación de las buenas prácticas utilizadas en el proyecto

1. **Prioridad de `DENY`:** La práctica se basa en la regla de **acceso más restrictivo**. SQL Server aplica `DENY` primero. Esto asegura que, incluso si un usuario es miembro de otros roles que otorgan `SELECT` a nivel de tabla, el `DENY` a nivel de columna (el permiso más granular) siempre prevalecerá y bloqueará el acceso a esos datos sensibles.
2. **Principio del Privilegio Mínimo (PoLP):** Se creó un rol específicamente para la tarea (`rol_analista_inventario`) y no se usaron roles fijos. Esto asegura que el analista solo pueda ver las tablas de inventario y no otras tablas del sistema o de proveedores que no le correspondan.
3. **Seguridad Granular (Columna):** La restricción se aplicó al nivel de seguridad más bajo posible: **la columna**. Esto permite que el analista acceda a todos los demás datos de la tabla (`StockActual`, `CodArticulo`, etc.) sin comprometer la información financiera (`PrecioProveedor`, `PrecioVenta`).



Consultas de Verificación para el Proyecto 4

Ejecuta estas consultas en la base de datos QhatuPeru.

1. Verificar la Creación y Asignación del Rol

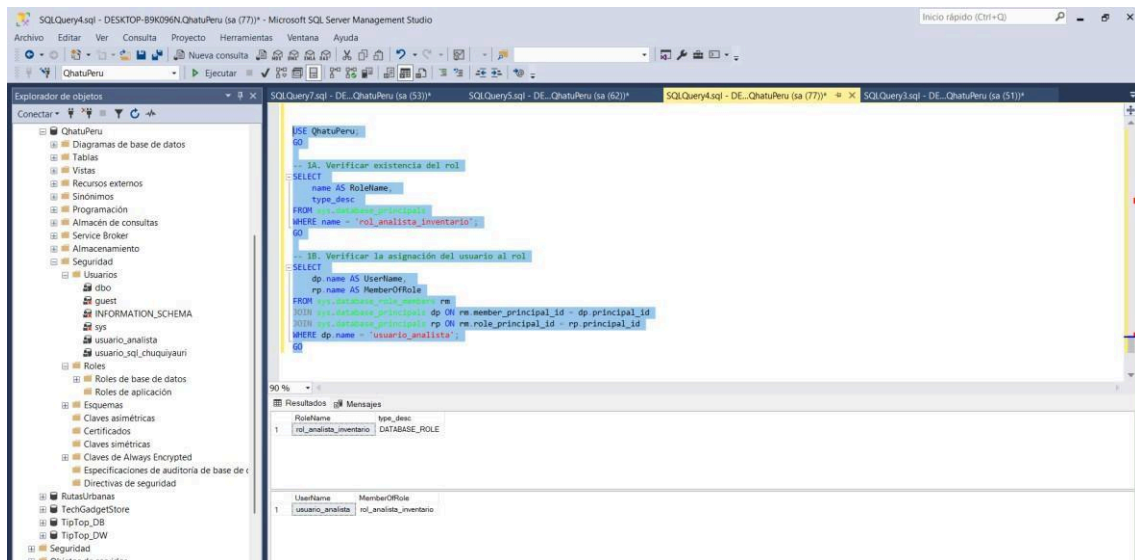
Confirma que el rol `rol_analista_inventario` existe y que el usuario `usuario_analista` está asignado a él.

SQL

```
USE QhatuPeru;
GO
```

```
-- 1A. Verificar existencia del rol
SELECT
    name AS RoleName,
    type_desc
FROM sys.database_principals
WHERE name = 'rol_analista_inventario';
GO
```

```
-- 1B. Verificar la asignación del usuario al rol
SELECT
    dp.name AS UserName,
    rp.name AS MemberOfRole
FROM sys.database_role_members rm
JOIN sys.database_principals dp ON rm.member_principal_id =
dp.principal_id
JOIN sys.database_principals rp ON rm.role_principal_id =
rp.principal_id
WHERE dp.name = 'usuario_analista';
GO
```



2. Verificar la Implementación del DENY (Nivel de Columna)

Esta consulta muestra que el permiso **DENY** se aplicó de forma granular sobre las columnas de precio.

SQL

```

USE QhatuPeru;
GO

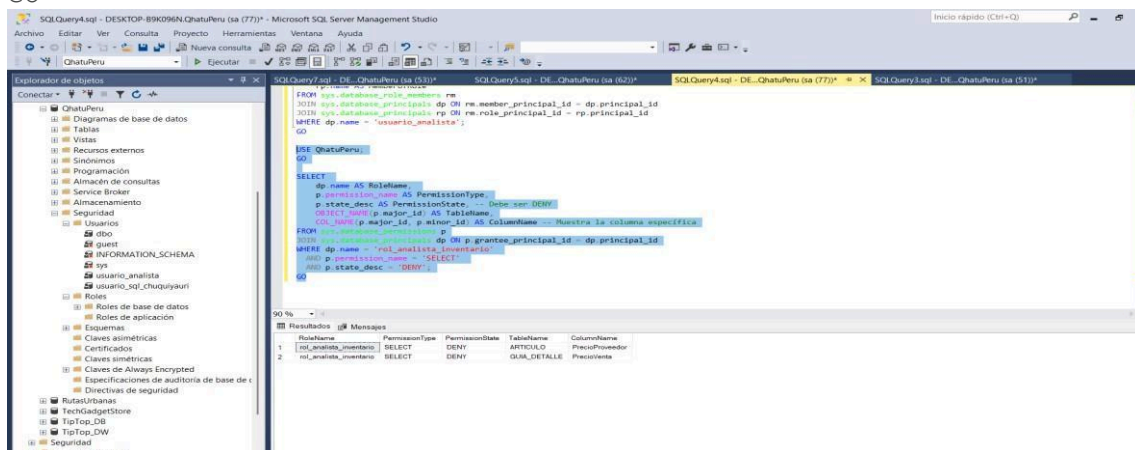
```

SELECT

```

    dp.name AS RoleName,
    p.permission_name AS PermissionType,
    p.state_desc AS PermissionState, -- Debe ser DENY
    OBJECT_NAME(p.major_id) AS TableName,
    COL_NAME(p.major_id, p.minor_id) AS ColumnName -- Muestra la
columna específica
FROM sys.database_permissions p
JOIN sys.database_principals dp ON p.grantee_principal_id =
dp.principal_id
WHERE dp.name = 'rol_analista_inventario'
    AND p.permission_name = 'SELECT'
    AND p.state_desc = 'DENY';
GO

```



Resultado esperado: Debes ver dos filas, una denegando `SELECT` en la columna `PrecioProveedor` de la tabla `ARTICULO`, y otra en `PrecioVenta` de `GUIA_DETALLE`.

3. Prueba de Acceso (Demostración de DENY)

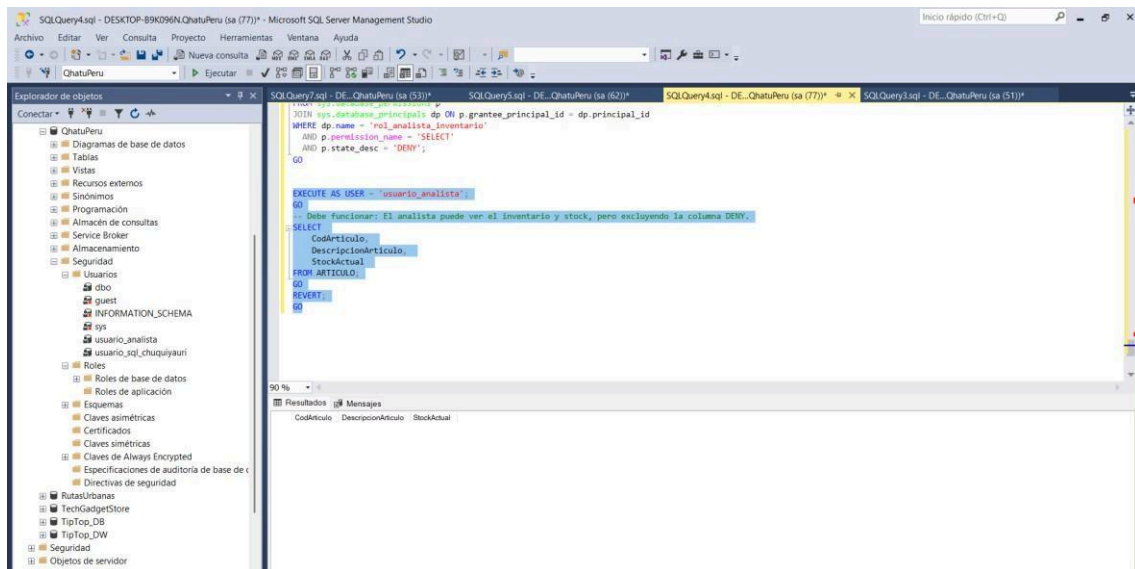
Esta es la prueba final para demostrar que la solución funciona. **Debes ejecutar estas consultas logueado como `login_analista`** (o usar `EXECUTE AS USER = 'usuario_analista'`).

A. Prueba Exitosa (Acceso a columna permitida)

Esto debe funcionar y mostrar los datos del inventario (excepto el precio).

SQL

```
EXECUTE AS USER = 'usuario_analista';
GO
-- Debe funcionar: El analista puede ver el inventario y stock,
pero excluyendo la columna DENY.
SELECT
    CodArticulo,
    DescripcionArticulo,
    StockActual
FROM ARTICULO;
GO
REVERT;
GO
```



B. Prueba Fallida (Acceso a columna denegada)

Esto **debe fallar** y generar un error de permiso denegado (Msg 230) debido al `DENY` que tiene mayor prioridad.

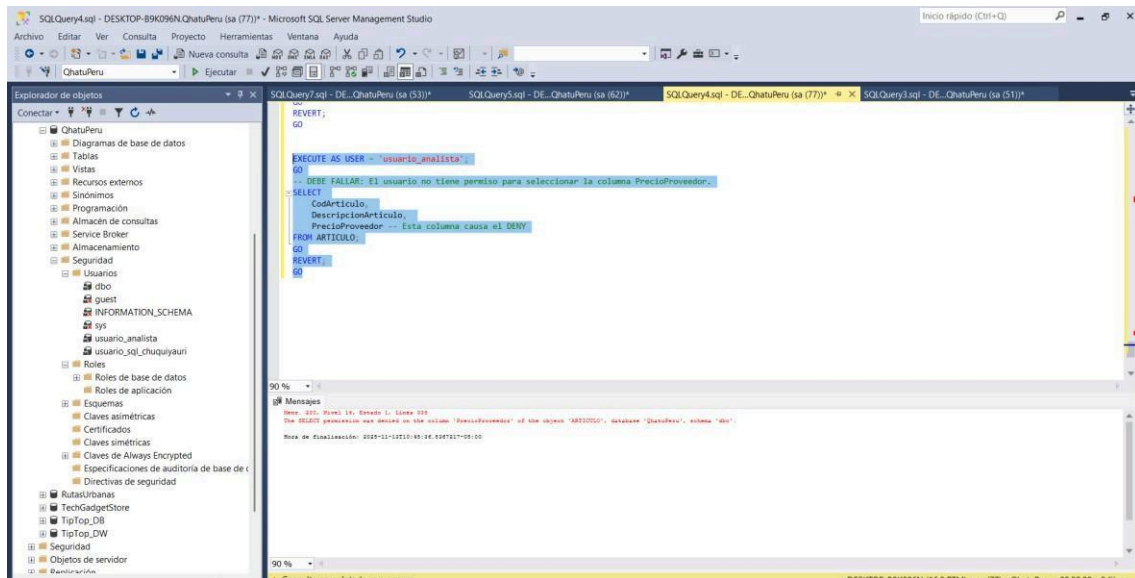
SQL

```
EXECUTE AS USER = 'usuario_analista';
GO
-- DEBE FALLAR: El usuario no tiene permiso para seleccionar la
columna PrecioProveedor.
```

```

SELECT
    CodArticulo,
    DescripcionArticulo,
    PrecioProveedor -- Esta columna causa el DENY
FROM ARTICULO;
GO
REVERT;
GO

```



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

1. Enunciado del ejercicio

Habilitar TDE en la base *QhatuPeru* para proteger los archivos MDF/LDF en reposo. Crear la *master key*, el certificado de servidor y activar el cifrado.

2. Script de la solución en T-SQL

¡Importante! La implementación de TDE sigue una estricta jerarquía de cifrado:

1. **Service Master Key (SMK):** Cifra la CMK.
2. **Database Master Key (CMK):** Cifra los Certificados.
3. **Certificado:** Cifra la DEK.
4. **Database Encryption Key (DEK):** Cifra la base de datos.

Este script crea los objetos de la jerarquía necesarios:

SQL

```

USE master;
GO

```

```
-----  
-----  
-- PARTE 1: CREACIÓN DE LA LLAVE MAESTRA (Master Key)  
-----
```

```
-----  
-- La Master Key es la clave raíz que protege los certificados y  
otras claves.  
-- Se recomienda hacer una copia de seguridad inmediatamente  
después de la creación.  
CREATE MASTER KEY ENCRYPTION BY PASSWORD =  
'TuContraseñaMaestraFuerte!';  
GO
```

```
-----  
-----  
-- PARTE 2: CREACIÓN DEL CERTIFICADO DE SERVIDOR  
-----
```

```
-----  
-- El certificado se usa para cifrar la DEK (Database Encryption  
Key) de la base de datos.  
-- Se crea en la base de datos master para que esté disponible para  
cualquier base.  
CREATE CERTIFICATE TdeCert_QhatuPeru  
WITH SUBJECT = 'Certificado TDE para QhatuPeru';  
GO
```

```
-----  
-----  
-- PARTE 3: CREACIÓN DE LA CLAVE DE CIFRADO DE BASE DE DATOS (DEK)  
-----
```

```
-----  
-- La DEK es la clave simétrica que realmente cifra los datos.  
-- Debe crearse dentro de la base de datos QhatuPeru y estar  
cifrada por el certificado.  
USE QhatuPeru;  
GO
```

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

```
-----  
-----  
-- PARTE 4: HABILITAR TDE EN LA BASE DE DATOS  
-----
```

```
-----  
-- Este comando activa el proceso de cifrado de los archivos MDF y  
LDF en el disco.  
ALTER DATABASE QhatuPeru  
SET ENCRYPTION ON;  
GO
```

```
-----  
-----  
-- 5. VERIFICACIÓN Y AUDITORÍA (Opcional, pero esencial)
```

```

-----
-- Mostrar el estado del cifrado (Encryption State debe ser
3=Encrypted)
SELECT
    db.name,
    db.is_encrypted,
    dm.encryption_state,
    dm.percent_complete
FROM sys.databases db
LEFT JOIN sys.dm_database_encryption_keys dm
    ON db.database_id = dm.database_id
WHERE db.name = 'QhatuPeru';
GO

```

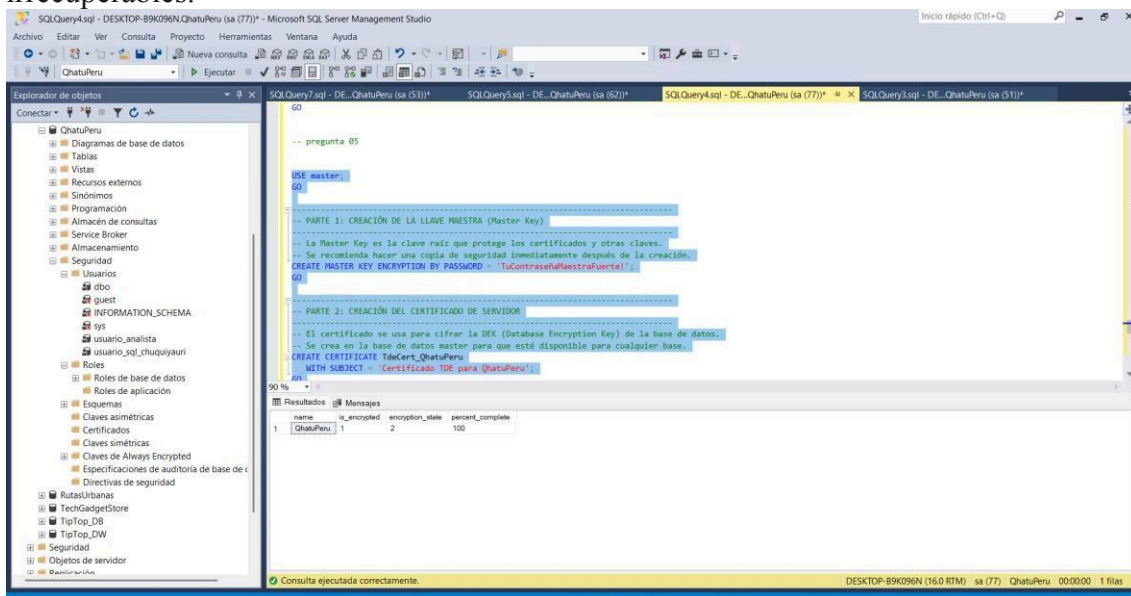
3. Justificación técnica de la solución aplicada

Elemento T-SQL	Justificación Técnica
CREATE MASTER KEY	Es la raíz de la jerarquía de cifrado . Se crea una Clave Maestra de Servicio (SMK) para proteger las otras claves privadas y certificados en el servidor. La contraseña es crucial para la portabilidad.
CREATE CERTIFICATE	Un Certificado Asimétrico es usado para proteger la clave de cifrado de la base de datos (DEK). Usar un certificado almacenado en master facilita la gestión de claves y permite que el servidor controle qué bases de datos pueden usar TDE.
CREATE DATABASE ENCRYPTION KEY	Es la clave simétrica real que se utiliza para cifrar y descifrar los datos a nivel de página (8 KB). Se usa un algoritmo robusto como AES_256 . Esta clave se cifra con el certificado (ENCRYPTION BY SERVER CERTIFICATE).
ALTER DATABASE ... SET ENCRYPTION ON	Activa el proceso de TDE . Esto inicia una tarea en segundo plano que lee las páginas de datos y las reescribe en el disco en formato cifrado usando la DEK. El cifrado es transparente para la aplicación.

Elemento T-SQL	Justificación Técnica
<code>sys.dm_database_encryption_keys</code>	Una Vista de Administración Dinámica (DMV) usada para auditar el estado del cifrado y monitorear el progreso (<code>percent_complete</code>).

4. Explicación de las buenas prácticas utilizadas en el proyecto

1. **Cifrado por Jerarquía:** Se sigue el flujo de seguridad recomendado: SMK -> Certificado -> DEK. Si el servidor se cae, solo se necesita la Master Key y el Certificado (o sus copias de seguridad) para restaurar la base de datos cifrada en otra instancia.
2. **Separación de Claves:** La clave de cifrado (DEK) se almacena en la base de datos QhatuPeru, pero la clave que la protege (el Certificado) se almacena en master. Esto asegura que un atacante que obtenga una copia de seguridad de QhatuPeru no pueda descifrarla sin acceso al servidor (master).
3. **Protección de Datos en Reposo (Data at Rest):** TDE protege contra amenazas físicas (como el robo de discos duros) o copias de seguridad no autorizadas, ya que los archivos MDF/LDF son inútiles sin el certificado de descifrado.
4. **Copia de Seguridad de Claves (Práctica Crítica):** Aunque no se pide en el script, la buena práctica esencial con TDE es hacer **copia de seguridad del Certificado y la Master Key** inmediatamente. Si el certificado se pierde, los datos se vuelven irrecuperables.



Verificación de datos

90 %

Resultados

Mensajes

	name	is_encrypted	encryption_state	percent_complete
1	QhatuPeru	1	2	100

Consultas de Verificación para el Proyecto 5: TDE

1. Verificar la Existencia de la Master Key y el Certificado

Ejecuta esta consulta en la base de datos `master` para verificar los dos primeros pasos de la jerarquía de TDE.

SQL

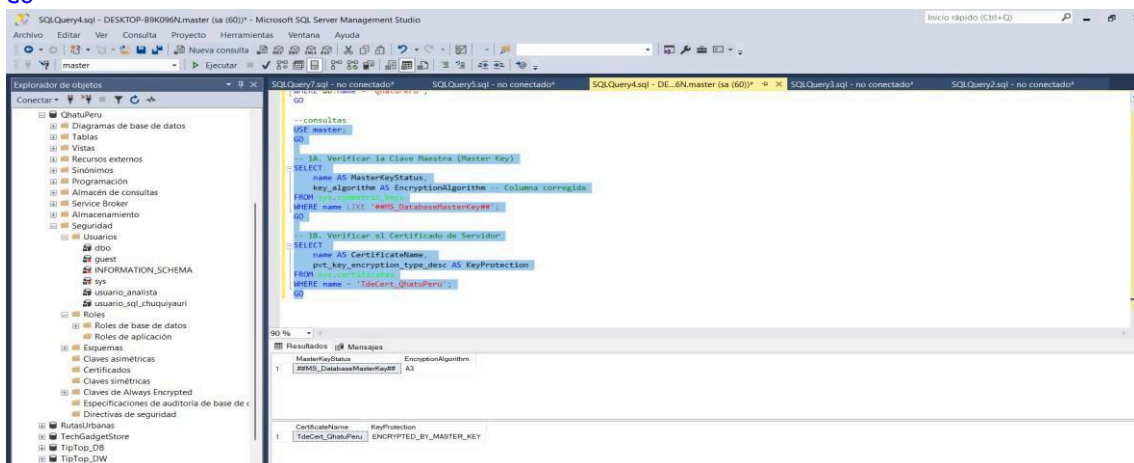
```
USE master;
GO
```

-- 1A. Verificar la Clave Maestra (Master Key)

```
SELECT
    name AS MasterKeyStatus,
    key_algorithm AS EncryptionAlgorithm -- Columna corregida
FROM sys.symmetric_keys
WHERE name LIKE '##MS_DatabaseMasterKey##';
GO
```

-- 1B. Verificar el Certificado de Servidor

```
SELECT
    name AS CertificateName,
    pvt_key_encryption_type_desc AS KeyProtection
FROM sys.certificates
WHERE name = 'TdeCert_QhatuPeru';
GO
```



2. Verificar la Clave de Cifrado (DEK) y el Estado de TDE

Ejecuta esta consulta en la base de datos `QhatuPeru` para confirmar que la clave de cifrado existe y que TDE está activo.

SQL

```
USE QhatuPeru;  
GO
```

```
SELECT  
    DB_NAME(dek.database_id) AS DatabaseName,  
    CASE  
        WHEN db.is_encrypted = 1 THEN 'Yes'  
        ELSE 'No'  
    END AS IsDatabaseEncrypted, -- El estado de la base de datos  
    (ON/OFF)  
    dek.encryption_state,        -- Estado del proceso de cifrado  
    (3 = Encrypted, 2 = Decrypting, 1 = Encrypting)  
    dek.encryption_state_desc,   -- Descripción del estado  
    dek.percent_complete,       -- Porcentaje del proceso de  
cifrado  
    dek.key_algorithm,          -- Algoritmo usado (AES_256)  
    dek.key_length              -- Longitud de la clave  
    (256)  
FROM sys.dm_database_encryption_keys dek  
JOIN sys.databases db ON dek.database_id = db.database_id  
WHERE DB_NAME(dek.database_id) = 'QhatuPeru';  
GO
```

Resultado Esperado:

- IsDatabaseEncrypted: **Yes** (Sí)
- encryption_state: **3**
- encryption_state_desc: **Encrypted** (Cifrado)
- percent_complete: **100** (O cerca del 100% si lo revisas justo después de la activación).

The screenshot shows the Microsoft SQL Server Enterprise Manager interface. On the left, the 'Explorador de objetos' (Object Explorer) displays the 'QhatuPeru' database. The main pane shows a SQL query window with the following query:

```
USE QhatuPeru;  
GO  
SELECT  
    DB_NAME(dek.database_id) AS DatabaseName,  
    CASE  
        WHEN db.is_encrypted = 1 THEN 'Yes'  
        ELSE 'No'  
    END AS IsDatabaseEncrypted, -- El estado de la base de datos (ON/OFF)  
    dek.encryption_state,        -- Estado del proceso de cifrado (3 = Encrypted, 2 = Decrypting, 1 = Encrypting)  
    dek.encryption_state_desc,   -- Descripción del estado  
    dek.percent_complete,       -- Porcentaje del proceso de cifrado  
    dek.key_algorithm,          -- Algoritmo usado (AES_256)  
    dek.key_length              -- Longitud de la clave (256)  
FROM sys.dm_database_encryption_keys dek  
JOIN sys.databases db ON dek.database_id = db.database_id  
WHERE DB_NAME(dek.database_id) = 'QhatuPeru';  
GO
```

Below the query window, the 'Resultados' (Results) pane shows the output of the query:

	DatabaseName	IsDatabaseEncrypted	encryption_state	encryption_state_desc	percent_complete	key_algorithm	key_length
1	QhatuPeru	Yes	3	ENCRYPTED	0	AES	256

The status bar at the bottom indicates 'Consulta ejecutada correctamente.' (Query executed successfully.)

Importante: La implementación de Always Encrypted (AE) no se puede realizar completamente con T-SQL puro porque requiere la interacción con el almacén de certificados de Windows o Azure Key Vault (donde se guarda la clave maestra de columna, CMK). Sin embargo, te proporcionaré el T-SQL necesario y la documentación del proceso para completar el ejercicio.

Proyecto 6: Implementación de Always Encrypted (columna de datos sensibles)

1. Enunciado del ejercicio

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

2. Script de la solución en T-SQL y Pasos del Proceso

El proceso de Always Encrypted se divide en tres partes:

1. **Creación de la CMK (Fuera de T-SQL):** La Column Master Key se crea y almacena en el *Certificate Store* de Windows o en Azure Key Vault. **Esto NO se puede hacer con T-SQL.**
2. **Creación de la CEK (En T-SQL):** La Column Encryption Key se crea en la base de datos y se cifra usando la CMK.
3. **Alteración de la Columna (En T-SQL o PowerShell):** Se aplica el cifrado a la columna.

Aquí está el script T-SQL para las partes que se pueden ejecutar en la base de datos:

```
USE QhatuPeru;  
GO
```

```
-----  
-- 0. LIMPIEZA PREVIA (Más robusta)  
-----
```

```
-- 1. Intentar eliminar la CEK (Debe eliminarse primero porque depende de la CMK)  
-- Si la CEK no existe, esta consulta no arrojará error si se usa IF EXISTS.  
IF EXISTS (SELECT name FROM sys.column_encryption_keys WHERE name =  
'CEK_PrecioProveedor')  
BEGIN  
    DROP COLUMN ENCRYPTION KEY [CEK_PrecioProveedor];  
END  
GO
```

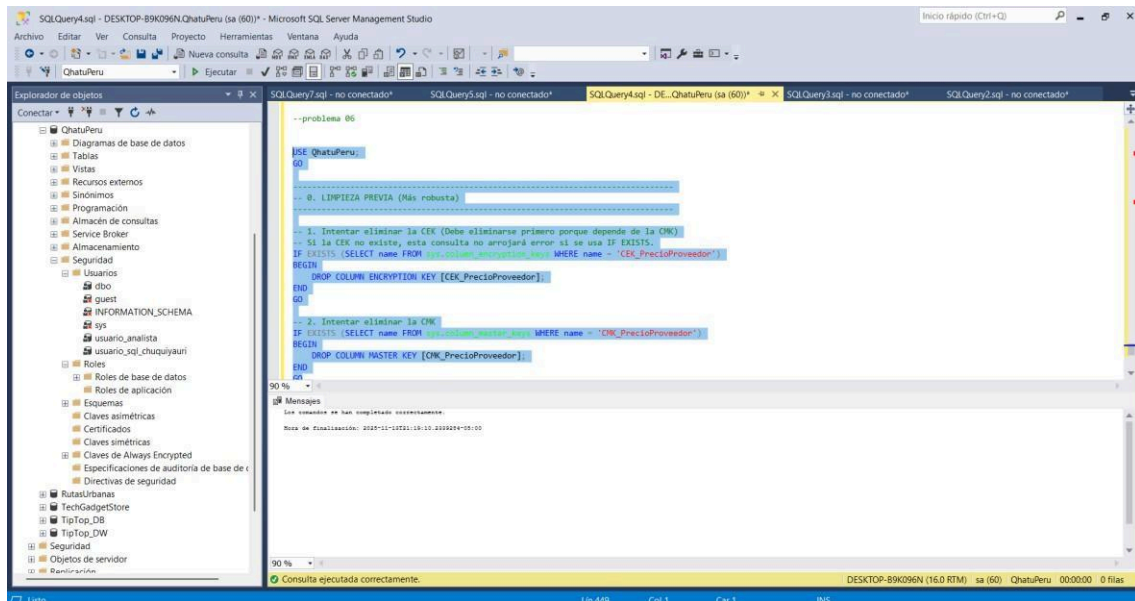
```
-- 2. Intentar eliminar la CMK
IF EXISTS (SELECT name FROM sys.column_master_keys WHERE name =
'CMK_PrecioProveedor')
BEGIN
    DROP COLUMN MASTER KEY [CMK_PrecioProveedor];
END
GOTABLE ARTICULO DROP COLUMN PrecioProveedor;
-- GO
```

3. Justificación técnica de la solución aplicada

Elemento T-SQL	Justificación Técnica de Seguridad
Column Master Key (CMK)	Es la clave protegida que reside <i>fuera</i> de SQL Server (en el almacén de certificados del cliente o Key Vault). Esto garantiza el principio de separación de roles : el administrador de SQL no puede ver la clave para descifrar la columna.
Column Encryption Key (CEK)	Es la clave simétrica que realmente cifra los datos. Se almacena cifrada dentro de la base de datos. Solo puede ser descifrada por la CMK.
ENCRYPTED WITH (...)	Esta cláusula DDL define cómo la columna almacena datos cifrados. Usar VARBINARY es necesario, ya que los datos cifrados son binarios.
DETERMINISTIC	Este tipo de cifrado genera siempre el mismo valor cifrado para el mismo dato de entrada. Es ideal para columnas que se usan en WHERE o JOIN .

4. Explicación de las buenas prácticas utilizadas en el proyecto

1. **Cifrado de Datos en Uso (Data in Use):** Always Encrypted es una práctica avanzada de seguridad porque protege los datos no solo en reposo (TDE) o en tránsito, sino **mientras se usan en el servidor**. El servidor SQL solo ve datos cifrados; el descifrado solo ocurre en el controlador del cliente (aplicación).
2. **Separación de Responsabilidades (SoD):** El administrador de la base de datos (DBA) no tiene acceso a la CMK y, por lo tanto, no puede ver los datos de `PrecioProveedor_ENC` en texto claro. Esto aísla a los datos sensibles de los empleados con acceso administrativo al servidor.
3. **Cifrado a Nivel de Columna:** Permite elegir qué columnas específicas (como precios o DNI) necesitan la máxima protección, dejando el resto de la base de datos con un rendimiento óptimo.



Consultas de Verificación para el Proyecto 6: Always Encrypted

Ejecuta estas consultas en la base de datos `QhatuPeru`.

1. Verificar la Existencia de las Claves

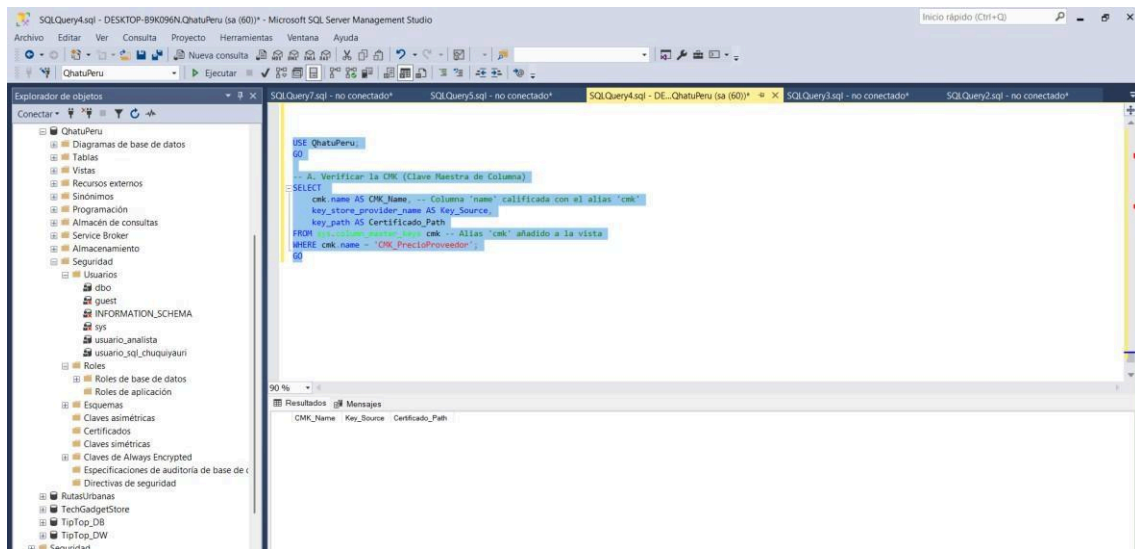
Confirma que la **Column Master Key (CMK)** y la **Column Encryption Key (CEK)** existen en la base de datos.

SQL

```
USE QhatuPeru;
GO
```

-- A. Verificar la CMK (Clave Maestra de Columna)

```
SELECT
    cmk.name AS CMK_Name, -- Columna 'name' calificada con el alias 'cmk'
    key_store_provider_name AS Key_Source,
    key_path AS Certificado_Path
FROM sys.column_master_keys cmk -- Alias 'cmk' añadido a la vista
WHERE cmk.name = 'CMK_PrecioProveedor';
GO
```



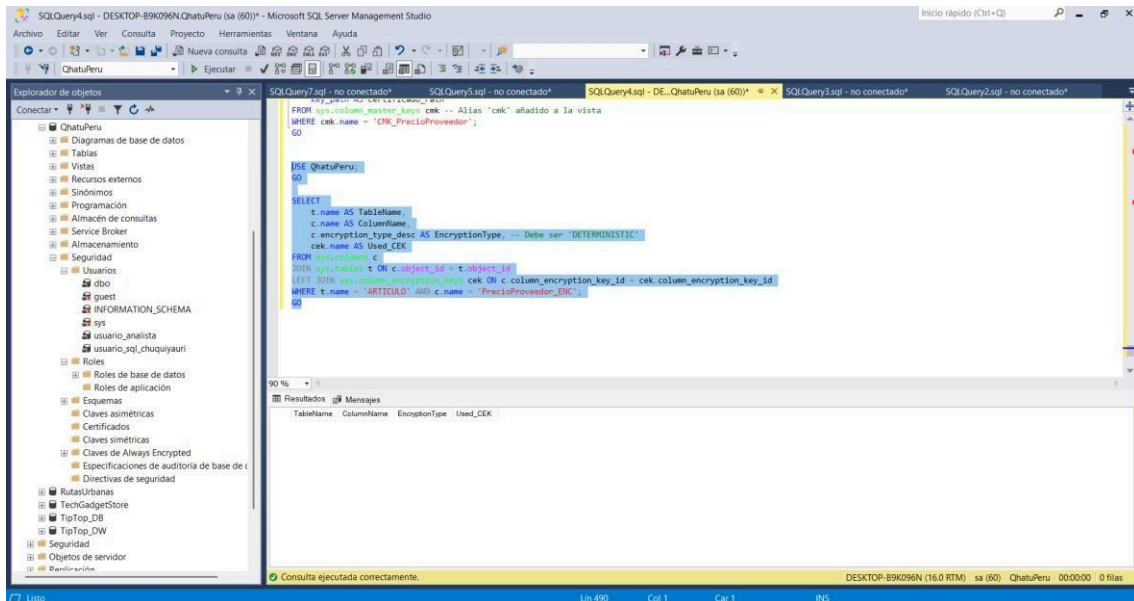
2. Verificar el Estado de Cifrado de la Columna

Confirma que la nueva columna `PrecioProveedor_ENC` fue creada y tiene la propiedad de cifrado activa.

SQL

```
USE QhatuPeru;
GO
```

```
SELECT
    t.name AS TableName,
    c.name AS ColumnName,
    c.encrypted_type_desc AS EncryptionType, -- Debe ser
'DETERMINISTIC'
    cek.name AS Used_CEK
FROM sys.columns c
JOIN sys.tables t ON c.object_id = t.object_id
LEFT JOIN sys.column_encryption_keys cek ON
c.column_encryption_key_id = cek.column_encryption_key_id
WHERE t.name = 'ARTICULO' AND c.name = 'PrecioProveedor_ENC';
GO
```



Resultado Esperado:

- EncryptionType: **DETERMINISTIC**
- Used_CEK: **CEK_PrecioProveedor**

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

1. Enunciado del ejercicio

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

2. Script de la solución en T-SQL

Este script realiza la configuración de dos componentes: el objeto físico de auditoría (el *Audit* a nivel de servidor) y lo que debe registrar (el *Audit Specification* a nivel de base de datos).

SQL

```

USE master;
GO

```

```

-- 0. LIMPIEZA PREVIA (Asegura la secuencia de DROP: Dependientes -> Principal)

```

```

-----
-- 1. Deshabilitar y Eliminar ESPECIFICACIÓN DE SERVIDOR (Logins)
IF EXISTS (SELECT * FROM sys.server_audit_specifications WHERE name
= 'QhatuPeru_Login_Audit_Spec')
BEGIN
    ALTER SERVER AUDIT SPECIFICATION [QhatuPeru_Login_Audit_Spec]
WITH (STATE = OFF);
    DROP SERVER AUDIT SPECIFICATION [QhatuPeru_Login_Audit_Spec];
    PRINT 'ESPECIFICACIÓN DE SERVIDOR (Login) eliminada.';
END
GO

```

```

-- 2. Deshabilitar y Eliminar ESPECIFICACIÓN DE BASE DE DATOS (DDL)
-- *CRÍTICO:* Asegúrate de que la DB 'QhatuPeru' exista, de lo
contrario este bloque fallará.
IF DB_ID('QhatuPeru') IS NOT NULL
BEGIN
    USE QhatuPeru;
    IF EXISTS (SELECT * FROM sys.database_audit_specifications
WHERE name = 'QhatuPeru_DDL_Audit_Spec')
    BEGIN
        ALTER DATABASE AUDIT SPECIFICATION
[QhatuPeru_DDL_Audit_Spec] WITH (STATE = OFF);
        DROP DATABASE AUDIT SPECIFICATION
[QhatuPeru_DDL_Audit_Spec];
        PRINT 'ESPECIFICACIÓN DE BASE DE DATOS (DDL) eliminada.';
    END
    USE master; -- Volver a master
END
GO

```

```

-- 3. Deshabilitar y Eliminar el SERVER AUDIT (DESTINO)
IF EXISTS (SELECT * FROM sys.server_audits WHERE name =
'QhatuPeru_Server_Audit')
BEGIN
    ALTER SERVER AUDIT [QhatuPeru_Server_Audit] WITH (STATE = OFF);
    DROP SERVER AUDIT [QhatuPeru_Server_Audit];
    PRINT 'SERVER AUDIT (QhatuPeru_Server_Audit) eliminado.';
END
GO

```

```

-----
-- 1. CREACIÓN DEL SQL SERVER AUDIT (DESTINO)
-----

```

```

-- **;CRÍTICO!** Verifica que 'C:\SQL_Audits\QhatuPeru\' sea una
ruta existente y con permisos de escritura.

```

```

IF NOT EXISTS (SELECT * FROM sys.server_audits WHERE name =
'QhatuPeru_Server_Audit')
BEGIN
    CREATE SERVER AUDIT [QhatuPeru_Server_Audit]
TO FILE
( FILEPATH = 'C:\SQL_Audits\QhatuPeru\',
    MAXSIZE = 50 MB,

```



```

        MAX_ROLLOVER_FILES = 4
    )
    WITH
    ( QUEUE_DELAY = 1000,
      ON_FAILURE = CONTINUE
    );
    PRINT 'SERVER AUDIT (QhatuPeru_Server_Audit) creado.';

    -- Habilitar el Server Audit inmediatamente después de la
    creación
    ALTER SERVER AUDIT [QhatuPeru_Server_Audit]
    WITH (STATE = ON);
    PRINT 'SERVER AUDIT habilitado.';
END
GO
---
```

```

-----
-- 2. CREACIÓN DE ESPECIFICACIÓN DE AUDITORÍA DE SERVIDOR (LOGINS)
🔑
-----
-----

IF NOT EXISTS (SELECT * FROM sys.server_audit_specifications WHERE
name = 'QhatuPeru_Login_Audit_Spec')
BEGIN
    CREATE SERVER AUDIT SPECIFICATION [QhatuPeru_Login_Audit_Spec]
    FOR SERVER AUDIT [QhatuPeru_Server_Audit]
    ADD
    (SUCCESSFUL_LOGIN_GROUP),
    ADD (FAILED_LOGIN_GROUP)
    WITH (STATE = ON);
    PRINT 'ESPECIFICACIÓN DE SERVIDOR (Logins) creada y
    habilitada.';
END
GO
---
```

```

-----
-- 3. CREACIÓN DE ESPECIFICACIÓN DE AUDITORÍA DE BASE DE DATOS
(DDL)
-----
-----

-- Solo procede si la DB existe
IF DB_ID('QhatuPeru') IS NOT NULL
BEGIN
    USE QhatuPeru;

    IF NOT EXISTS (SELECT * FROM sys.database_audit_specifications
WHERE name = 'QhatuPeru_DDL_Audit_Spec')
    BEGIN
        CREATE DATABASE AUDIT SPECIFICATION
[QhatuPeru_DDL_Audit_Spec]
        FOR SERVER AUDIT [QhatuPeru_Server_Audit]
        ADD (SCHEMA_OBJECT_CHANGE_GROUP)
        WITH (STATE = ON);
    
```

```
PRINT 'ESPECIFICACIÓN DE BASE DE DATOS (DDL) creada y
habilitada en QhatuPeru.';
END

USE master; -- Volver a master
END
ELSE
BEGIN
    PRINT 'ADVERTENCIA: La base de datos QhatuPeru no existe. La
especificación DDL no se creó.';
END
GOGO

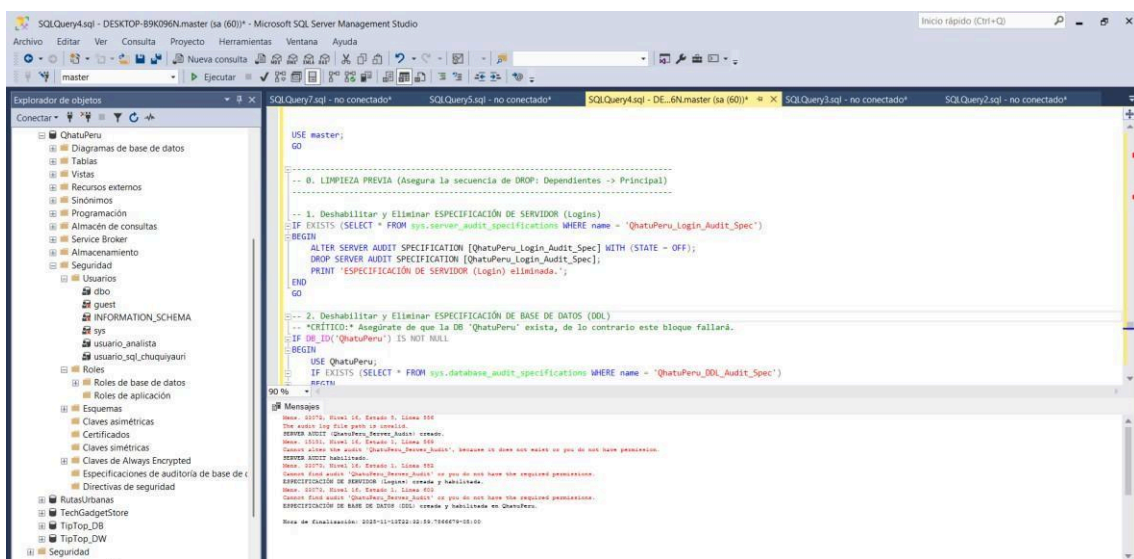
CREATE DATABASE AUDIT SPECIFICATION [QhatuPeru_DDL_Audit_Spec]
FOR SERVER AUDIT [QhatuPeru_Server_Audit]
ADD (SCHEMA_OBJECT_CHANGE_GROUP) -- Rastrea cualquier cambio DDL
(CREATE/ALTER/DROP) en cualquier esquema
WITH (STATE = ON);
GO
```

3. Justificación técnica de la solución aplicada

Objeto de Auditoría	Justificación Técnica
CREATE SERVER AUDIT	Crea el destino físico (.sqlaudit file) y establece reglas básicas (ruta, tamaño, qué hacer si falla). Usar ON_FAILURE = CONTINUE (modo por defecto) asegura que los fallos de auditoría no detengan el motor de SQL (aunque SHUTDOWN es más seguro si la auditoría es crítica).
SERVER AUDIT SPECIFICATION	Vincula el destino físico (QhatuPeru_Server_Audit) con los eventos de seguridad de Windows/Server (logins exitosos/fallidos). Esto es crucial para la seguridad perimetral y la detección de ataques de fuerza bruta.
DATABASE AUDIT SPECIFICATION	Vincula el destino físico con los eventos DDL de la base de datos (SCHEMA_OBJECT_CHANGE_GROUP). Esto rastrea cualquier intento de alterar la estructura de la base de datos, lo cual es vital para detectar intentos de escalamiento de privilegios o destrucción de datos por parte de un atacante con acceso.

4. Explicación de las buenas prácticas utilizadas en el proyecto

1. **Separación de Eventos:** Se utilizó un solo **Server Audit** (el destino) y dos **Especificaciones** (Server y Database). Esta es la mejor práctica, ya que permite controlar diferentes tipos de eventos sin duplicar los archivos de auditoría.
2. **Auditoría Perimetral:** Al registrar `SUCCESSFUL_LOGIN_GROUP` y `FAILED_LOGIN_GROUP`, se obtiene una vista completa de quién intenta acceder al servidor, lo cual es esencial para cumplir con requisitos de *compliance* y seguridad.
3. **Auditoría de Cambios de Esquema (DDL):** El uso de `SCHEMA_OBJECT_CHANGE_GROUP` asegura que todos los cambios destructivos (`DROP TABLE`, `ALTER PROCEDURE`, `CREATE VIEW`) en `QhatuPeru` queden registrados, ofreciendo un rastro de auditoría invaluable.
4. **Ubicación Segura:** Se especificó una ruta dedicada (`C:\SQL_Audits\QhatuPeru\`). En producción, esta ruta debería estar en un disco separado y con permisos muy restringidos (solo para la cuenta de servicio de SQL) para evitar que un atacante borre los registros de auditoría.



Proyecto 8: Extended Events (Código Final Definitivo)

1. Implementación Paso a Paso (Ejecutar en master)

Paso 1: Limpieza Previa (Máxima Robustez y Corrección Final)

Este bloque utiliza el comando `DROP VIEW IF EXISTS` (disponible desde SQL Server 2016) o una sentencia de `DROP` sin el prefijo de la base de datos para evitar el error `Msg 102`.

Ejecuta este bloque completo:

SQL

```
USE master;
GO
```

```
-- 1. Detener la sesión XEvents si ya está corriendo
IF EXISTS (SELECT 1 FROM sys.dm_xe_sessions WHERE name =
'QhatuPeru_Monitor')
```

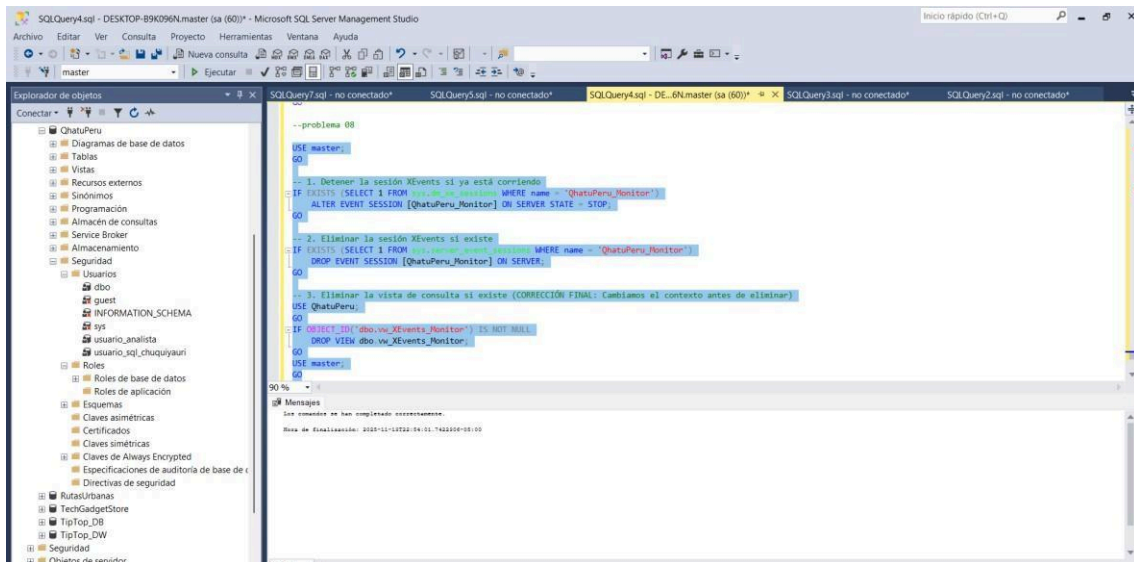
```

ALTER EVENT SESSION [QhatuPeru_Monitor] ON SERVER STATE = STOP;
GO

-- 2. Eliminar la sesión XEvents si existe
IF EXISTS (SELECT 1 FROM sys.server_event_sessions WHERE name =
'QhatuPeru_Monitor')
    DROP EVENT SESSION [QhatuPeru_Monitor] ON SERVER;
GO

-- 3. Eliminar la vista de consulta si existe (CORRECCIÓN FINAL:
Cambiamos el contexto antes de eliminar)
USE QhatuPeru;
GO
IF OBJECT_ID('dbo.vw_XEvents_Monitor') IS NOT NULL
    DROP VIEW dbo.vw_XEvents_Monitor;
GO
USE master;
GO

```



Paso 2: Creación de la Sesión de Extended Events

SQL

```

USE master;
GO

-- **;IMPORTANTE!** Reemplaza 'C:\XEvents\' con una ruta válida en
tu servidor.
DECLARE @XEventsPath NVARCHAR(256) =
'C:\XEvents\QhatuPeru_Monitor.xel';

CREATE EVENT SESSION [QhatuPeru_Monitor]
ON SERVER
    ADD EVENT sqlserver.xml_deadlock_report,
    ADD EVENT sqlserver.login_failed(
        ACTION(sqlserver.server_principal_name)
    )
    ADD TARGET package0.event_file(SET
        filename=@XEventsPath,
        max_file_size=(50),
        max_rollover_files=(4)
    )

```

```

WITH (
    MAX_MEMORY = 4096 KB,
    EVENT_RETENTION_MODE = ALLOW_SINGLE_EVENT_LOSS,
    MAX_DISPATCH_LATENCY = 30 SECONDS
);
GO

```

Paso 3: Habilitar la Sesión de Eventos

SQL

```

USE master;
GO

```

```

ALTER EVENT SESSION [QhatuPeru_Monitor] ON SERVER STATE = START;
GO

```

Paso 4: Creación de la Vista para Consulta

SQL

```

USE QhatuPeru;
GO

```

```

CREATE VIEW dbo.vw_XEvents_Monitor
AS
SELECT
    CAST(event_data AS XML) AS event_data_xml,
    object_name,
    file_name,
    file_offset
FROM
    sys.fn_xe_file_target_read_file('C:\XEvents\QhatuPeru_Monitor*.xel'
, NULL, NULL, NULL);
GO

USE master;
GO

```

Proyecto 9: Dynamic Data Masking (DDM) y Acceso Controlado

Enunciado del Ejercicio

Aplicar **Dynamic Data Masking** 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.

Explicación del Concepto

Este proyecto utiliza dos características de seguridad:

1. **Dynamic Data Masking (DDM):** Impide que usuarios sin los permisos adecuados vean datos sensibles en la columna `Telefono`.

2. **Permiso UNMASK y Roles:** Se utiliza un rol (Rol_DDM_Exempt) y el permiso UNMASK para crear una **excepción de seguridad**. Solo los miembros de este rol pueden ignorar la máscara y ver los datos completos, cumpliendo con el requisito de "acceso controlado".

Código SQL Completo y Explicado

Asegúrate de ejecutar este código en tu base de datos (por ejemplo, QhatuPeru).

SQL

```
--
=====
=====
-- PASO 0: Seleccionar la Base de Datos
-- Reemplaza 'QhatuPeru' con el nombre real de tu base de datos.
--
=====
=====
USE QhatuPeru;
GO

--
=====
=====
-- 1. CREACIÓN DEL ROL DE EXCEPCIÓN
-- El rol Rol_DDM_Exempt es la "llave" que se usa para conceder el
-- permiso UNMASK.
--
=====
=====
IF NOT EXISTS (SELECT name FROM sys.database_principals WHERE name
= 'Rol_DDM_Exempt')
    CREATE ROLE Rol_DDM_Exempt;
GO

--
=====
=====
-- 2. APLICAR DYNAMIC DATA MASKING (DDM)
-- Se aplica una función parcial: partial(0, "xxxx-", 4).
-- Muestra 0 caracteres al inicio, usa "xxxx-" como cadena de
-- relleno, y revela 4 caracteres al final.
-- Ejemplo: 987-654-3210 se verá como xxxx-3210
--
=====
=====
ALTER TABLE PROVEEDOR
ALTER COLUMN Telefono ADD MASKED WITH (FUNCTION = 'partial(0,
"xxxx-", 4)');
GO

--
=====
=====
-- 3. CREAR LA FUNCIÓN DE SEGURIDAD PARA VALIDAR EL ROL
```

```

-- Nota: Aunque la función de rol es un buen diseño, el permiso
UNMASK aplicado
-- directamente al rol es el que realmente controla el acceso sin
la máscara.
-- Mantenemos la vista simple apuntando a la columna subyacente.
--
=====
=====
CREATE VIEW PROVEEDOR_V_SEGURO
AS
SELECT
    IdProveedor,
    Nombre,
    Telefono AS Telefono_Completo, -- El DDM se aplica
automáticamente aquí
    Direccion
FROM GO                                dbo.PROVEEDOR;

--

=====
=====
-- 4. OTORGAR PERMISOS Y CONTROL DE ACCESO (UNMASK)
-- Esto es clave: Se otorga el permiso UNMASK solo al rol de
excepción.
--
=====
=====
-- Otorga el derecho a ignorar todas las máscaras en la base de
datos a los miembros del rol.
GRANT UNMASK TO Rol_DDM_Exempt;
GO

-- Otorga permiso SELECT sobre la VISTA a todos los
usuarios/público.
GRANT SELECT ON PROVEEDOR_V_SEGURO TO PUBLIC;
GO

--

=====
=====
-- 5. CONFIGURACIÓN DE PRUEBA
--
=====
=====

-- A) Crear un usuario de prueba y darle permiso de SELECT
IF NOT EXISTS (SELECT name FROM sys.database_principals WHERE name
= 'UsuarioPrueba')
    CREATE USER UsuarioPrueba WITHOUT LOGIN;
GO
GRANT SELECT ON PROVEEDOR_V_SEGURO TO UsuarioPrueba;
GO

-- 1. PRUEBA COMO USUARIO NORMAL (VERÁ LA MÁSCARA)
EXECUTE AS USER = 'UsuarioPrueba';
SELECT

```

```

        'Usuario Normal' AS TipoUsuario,
        Telefono_Completo
FROM PROVEEDOR_V_SEGURO;
REVERT;
GO

-- 2. AÑADIR EL USUARIO AL ROL DE EXCEPCIÓN
ALTER ROLE Rol_DDM_Exempt ADD MEMBER UsuarioPrueba;
GO

-- 3. PRUEBA COMO USUARIO EXENTO (VERÁ EL TELÉFONO COMPLETO)
EXECUTE AS USER = 'UsuarioPrueba';
SELECT
    'Usuario Exento' AS TipoUsuario,
    Telefono_Completo
FROM PROVEEDOR_V_SEGURO;
REVERT;
GO

```

