UNIVERSIDAD PERUANA LOS ANDES FACULTAD DE INGENIERÍA

Asignatura: Base de Datos II

Docente: Dr. Raúl Enrique Fernández Bejarano

Alumno: SORIANO TIMOTEO Joel Kevin

Ciclo: V

huancayo-Perú-2025

Manual de Práctica Calificada 02

Proyecto 1: Autenticación: Comparación segura y configuración de logins

1. Enunciado del ejercicio

Se debe crear en el servidor dos cuentas de inicio de sesión para pruebas:
una que utilice autenticación SQL (**login_sql_joel_soriano_timoteo**) y otra que represente un usuario de Windows de manera simulada (**DOMAIN\joel_soriano_timoteo_win**).
A ambas cuentas se les deben aplicar las políticas de seguridad de contraseñas y posteriormente vincularlas con la base de datos **QhatuPeru**.
Finalmente, se solicita mostrar cómo activar la expiración obligatoria de la contraseña y verificar el cumplimiento de las políticas de seguridad configuradas.

2. Script de la solución en T-SQL

```
USE master;

GO

-- Crear Login con Autenticación SQL

CREATE LOGIN login_sql_joel_soriano_timoteo

WITH PASSWORD = 'Pass_12345!',

    CHECK_POLICY = ON,

    CHECK_EXPIRATION = ON;

GO
```

```
-- Crear Login con Autenticación Windows (Simulado)
CREATE LOGIN [DOMAIN\joel_soriano_timoteo_win]
FROM WINDOWS;
GO

USE QhatuPeru;
GO

-- Crear usuario en la base QhatuPeru para el Login SQL
CREATE USER usuario_sql_joel_soriano_timoteo
FOR LOGIN login_sql_joel_soriano_timoteo;
GO

-- Crear usuario en la base QhatuPeru para el Login Windows
CREATE USER usuario_win_joel_soriano_timoteo
FOR LOGIN [DOMAIN\joel_soriano_timoteo_win];
GO

-- Comprobación del estado de políticas del login SQL
SELECT
```

```
    name AS LoginName,
    is_policy_checked AS PolicyApplied,
    is_expiration_checked AS ExpirationForced
FROM sys.sql_logins
WHERE name = 'login_sql_joel_soriano_timoteo';
GO
```

3. Justificación técnica de la solución aplicada

Comando / Opción	Justificación Técnica
CREATE LOGIN login_sql_joel_soriano_timoteo	Genera un *security principal* a nivel de servidor empleando autenticación nativa de SQL Server.
CHECK_POLICY = ON	Obliga a que la contraseña cumpla con las reglas de complejidad definidas por SQL Server o Windows.
CHECK_EXPIRATION = ON	Indica que la contraseña debe renovarse al iniciar sesión, reforzando las prácticas seguras de administración de cuentas.
CREATE LOGIN [DOMAIN\joel_soriano_timoteo_win] FROM WINDOWS	Crea un login cuya validación es gestionada por el sistema operativo, evitando almacenar la contraseña en SQL Server.

CREATE USER … FOR LOGIN …	Realiza el vínculo entre el login del servidor y un usuario dentro de la base QhatuPeru, permitiendo asignar permisos a nivel de base de datos.

4. Explicación de las buenas prácticas utilizadas en el proyecto

- **Nomenclatura clara:**
  Los nombres de logins y usuarios (por ejemplo: *login_sql_joel_soriano_timoteo* y *usuario_sql_joel_soriano_timoteo*) se estructuran de forma que identifiquen fácilmente al propietario o propósito, mejorando la trazabilidad y la gestión administrativa.

- **Separación de credenciales:**
  La creación independiente del Login (acceso al servidor) y del Usuario (permisos dentro de la base) cumple con el **Principio de Privilegio Mínimo**, permitiendo un control más preciso sobre qué puede hacer cada cuenta.

- **Autenticación reforzada:**
  Las opciones **CHECK_POLICY = ON** y **CHECK_EXPIRATION = ON** aseguran el uso de contraseñas seguras, complejas y sujetas a renovación periódica, lo cual disminuye la exposición a ataques por credenciales débiles.

Código de consulta de creación de login

```
-- Consulta de creación
USE master;
GO

SELECT
    name AS LoginName,
    type_desc AS AutenticationType,
    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_joel_soriano_timoteo',
               'DOMAIN\joel_soriano_timoteo_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_joel_soriano_timoteo',
                  'usuario_win_joel_soriano_timoteo');
GO
```

Proyecto 2: Cuentas de servicio y configuración segura del servidor

1. Enunciado del ejercicio

Se solicita revisar y registrar la configuración de seguridad del servidor utilizada en **QhatuPeru**.
Esto incluye desactivar la característica **xp_cmdshell**, verificar el estado de la opción **contained database authentication**, y crear una **credencial y un proxy** que permitan a los trabajos del SQL Agent ejecutar acciones controladas en el sistema operativo cuando sea necesario.

2. Script de la solución en T-SQL

A continuación se presenta el código T-SQL que implementa todas las configuraciones de seguridad requeridas.

```
USE master;
GO

-----------------------------------------------------------------------
------------
-- 1. DESHABILITAR XP_CMDSHELL (Aseguramiento del servidor)
-----------------------------------------------------------------------
------------

-- Activar opciones avanzadas para poder modificar xp_cmdshell
EXEC sp_configure 'show advanced options', 1;
RECONFIGURE;
GO

-- Deshabilitar xp_cmdshell para evitar ejecución directa de
comandos del OS
EXEC sp_configure 'xp_cmdshell', 0;
RECONFIGURE;
GO

-----------------------------------------------------------------------
------------
-- 2. VERIFICAR ESTADO DE 'CONTAINED DATABASE AUTHENTICATION'
-----------------------------------------------------------------------
------------

-- Comprobar si la autenticación contenida está habilitada en el
servidor
EXEC sp_configure 'contained database authentication';
GO

-- Si estuviera habilitada, podría desactivarse (recomendado si no
se usa)
-- EXEC sp_configure 'contained database authentication', 0;
-- RECONFIGURE;
-- GO
```

```
-------------------------------------------------------------------
------------
-- 3. CREACIÓN DE CREDENCIAL Y PROXY PARA JOBS DE SQL AGENT
-------------------------------------------------------------------
------------

-- 3.1 Crear una credencial con usuario del sistema operativo
CREATE CREDENTIAL [Credential_OS_Access]
WITH IDENTITY = 'DOMAIN\OS_User_for_Agent',     -- Usuario del SO
con permisos mínimos
     SECRET   = 'ContrasenaSegura!';            -- Contraseña del
usuario del SO
GO

-- 3.2 Crear un proxy que utilice la credencial anterior
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 sistema operativo.';
GO

-- 3.3 Conceder permisos al proxy para su uso desde subsistemas del
SQL Agent
-- EXEC msdb.dbo.sp_grant_proxy_to_subsystem
--     @proxy_name = 'Proxy_QhatuPeru_OS_Access',
--     @subsystem_id = 3;  -- CmdExec (comandos del OS)

-- Asociar un login con el proxy
-- EXEC msdb.dbo.sp_grant_login_to_proxy
--     @proxy_name = 'Proxy_QhatuPeru_OS_Access',
--     @login_name = 'login_sql_joel_soriano_timoteo';
-- GO
```

3. Justificación técnica de la solución aplicada

Elemento T-SQL	Justificación Técnica de Seguridad

sp_configure 'xp_cmdshell', 0	Evita que SQL Server ejecute comandos del sistema operativo directamente. Esta característica es riesgosa porque, si un atacante accede a SQL, podría ejecutar acciones en el OS. Desactivarla reduce la superficie de ataque.
contained database authentication	Permite que las bases de datos manejen usuarios independientes de los logins del servidor. Aunque útil en migraciones, introduce otro nivel de administración de seguridad. Se debe revisar su uso y mantenerlo desactivado si no es necesario.
CREATE CREDENTIAL	Proporciona un medio seguro para almacenar credenciales de Windows. Aísla las credenciales del servicio de SQL Agent y permite aplicar el Principio de Privilegio Mínimo.
sp_add_proxy	Crea un mecanismo para que los Jobs del SQL Agent ejecuten tareas con permisos controlados, evitando usar la cuenta del servicio del SQL Agent, que normalmente tiene más privilegios de los necesarios.

4. Explicación de las buenas prácticas utilizadas en el proyecto

1. **Reducción de la superficie de ataque:**
   Desactivar **xp_cmdshell** es una medida esencial de endurecimiento del servidor, ya que evita que procesos malintencionados ejecuten comandos del sistema operativo desde SQL.

2. **Aplicación del Principio de Mínimo Privilegio (PoLP):**

   - La cuenta del SQL Agent debe tener solo los permisos básicos para que el servicio funcione.

   - Si un job requiere ejecutar comandos del sistema operativo, se usa una **Credencial + Proxy**, asignando únicamente los permisos mínimos necesarios a ese usuario del sistema operativo.
     Esto limita el alcance de cualquier posible compromiso.

3. **Control sobre usuarios contenidos:**
   Revisar la opción **contained database authentication** es clave, ya que permite usuarios internos que no dependen de los logins de servidor. Aunque mejora la portabilidad, puede complicar la seguridad si se habilita sin una estrategia clara.

Consultas de lo ya realizado

```
USE master;
GO

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

Proyecto 3 – Roles fijos y roles personalizados (Server & DB)

Versión reformulada por Joel Soriano Timoteo

1. Enunciado del ejercicio (modificado)

Se solicita crear un rol de base de datos específico llamado **ventas_readwrite**, cuyo propósito es conceder únicamente los permisos necesarios para trabajar con las tablas vinculadas al proceso de ventas (por ejemplo: **GUIA_ENVIO** y **GUIA_DETALLE**).
El usuario asignado debe tener permiso para **leer (SELECT), insertar (INSERT) y actualizar (UPDATE)** registros en dichas tablas.
Adicionalmente, se debe comparar este rol personalizado con un rol fijo de SQL Server, como **db_datareader**, para evidenciar la diferencia en alcance y seguridad.

2. Script T-SQL de la solución (versión mejorada)

```
USE QhatuPeru;
GO
-------------------------------------------------------------
-- PARTE 1: CREACIÓN DEL ROL PERSONALIZADO
-------------------------------------------------------------

-- 1.1 Crear el rol a nivel de base de datos
CREATE ROLE ventas_readwrite;
GO
```

```
-- 1.2 Otorgar permisos específicos al rol sobre las tablas de
ventas
-- Permisos de lectura
GRANT SELECT ON GUIA_ENVIO TO ventas_readwrite;
GRANT SELECT ON GUIA_DETALLE TO ventas_readwrite;

-- Permisos de inserción
GRANT INSERT ON GUIA_ENVIO TO ventas_readwrite;
GRANT INSERT ON GUIA_DETALLE TO ventas_readwrite;

-- Permisos de actualización
GRANT UPDATE ON GUIA_ENVIO TO ventas_readwrite;
GRANT UPDATE ON GUIA_DETALLE TO ventas_readwrite;
GO

-- (Opcional) Permitir lectura a las tablas de referencia para
consultas relacionadas
GRANT SELECT ON ARTICULO TO ventas_readwrite;
GRANT SELECT ON TIENDA TO ventas_readwrite;
GO

-------------------------------------------------------------
-- PARTE 2: ASIGNACIÓN DE USUARIOS A LOS ROLES
-------------------------------------------------------------

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

-- 2.2 Para comparación: añadir el usuario Windows a un rol fijo
ALTER ROLE db_datareader ADD MEMBER usuario_win_chuquiyauri;
GO
```

3. Justificación técnica de la solución (redacción nueva)

Elemento T-SQL	Justificación técnica

CREATE ROLE ventas_readwrite	Permite definir un rol adaptado a las necesidades reales. Facilita la administración de seguridad agrupando permisos y asignándolos a varios usuarios sin manejarlos individualmente.
GRANT SELECT/INSERT/UPDATE ON tabla TO rol	Implementa el *Principio del Privilegio Mínimo* al otorgar únicamente los permisos indispensables sobre tablas específicas. Se evita conceder permisos globales como db_datawriter.
ALTER ROLE ... ADD MEMBER	Asocia al usuario con los permisos definidos en el rol, simplificando la gestión y garantizando consistencia en la seguridad de la base.
db_datareader (rol fijo)	Permite lecturas en todas las tablas de la base de datos, lo cual sirve para comparar lo restrictivo del rol personalizado frente al alcance amplio de un rol fijo.

4. Buenas prácticas aplicadas (nuevo enfoque)

1. Uso de roles personalizados (Recomendado)

Los roles específicos como **ventas_readwrite** permiten:

- Diseñar privilegios alineados al área funcional (ventas).
- Evitar acceso innecesario a otras áreas como inventario o proveedores.
- Gestionar permisos de forma centralizada y más segura.

2. Comparación con roles fijos (db_datareader)

- **db_datareader** entrega permisos *SELECT* en *todas* las tablas del esquema, lo cual puede dar más acceso del necesario.
- En cambio, **ventas_readwrite** otorga permisos solo sobre las tablas que realmente necesitan manipulación.

3. Aplicación del Principio de Mínimo Privilegio (PoLP)

Se asegura que un usuario solo puede realizar operaciones esenciales para su labor, reduciendo el riesgo de modificaciones accidentales o intencionales en tablas críticas.

4. Administración eficiente de permisos

Al usar roles:

- Se evitan permisos directos usuario-objeto.
- Se facilita la auditoría y la trazabilidad.
- Los cambios se aplican al rol y afectan a todos los usuarios asociados.

Consultas de Verificación – Proyecto 3 (Roles y Permisos)

Base de datos: `QhatuPeru`
Objetivo: comprobar que el rol, permisos y asignaciones fueron aplicados correctamente.

---

1. Verificar la existencia del rol personalizado

Confirma que el rol **ventas_readwrite** fue creado.

```
USE QhatuPeru;
GO

SELECT
    name AS RoleName,
    type_desc AS RoleType
FROM sys.database_principals
WHERE name = 'ventas_readwrite';
GO
```

---

2. Verificar los permisos asignados al rol

Muestra los permisos otorgados a **ventas_readwrite** sobre cada tabla.

```
USE QhatuPeru;
GO
```

```
SELECT
    dp.name AS RoleName,
    o.name AS ObjectName,
    p.permission_name AS Permission,
    p.state_desc AS State
FROM sys.database_permissions p
JOIN sys.objects o ON p.major_id = o.object_id
JOIN sys.database_principals dp ON p.grantee_principal_id =
dp.principal_id
WHERE dp.name = 'ventas_readwrite'
ORDER BY o.name, p.permission_name;
GO
```

---

3. Verificar los miembros del rol (usuarios asignados)

Confirma que el usuario SQL fue agregado al rol personalizado.

```
USE QhatuPeru;
GO

SELECT
    r.name AS RoleName,
    m.name AS MemberUser
FROM sys.database_role_members drm
JOIN sys.database_principals r ON drm.role_principal_id =
r.principal_id
JOIN sys.database_principals m ON drm.member_principal_id =
m.principal_id
WHERE r.name = 'ventas_readwrite';
GO
```

---

4. Verificar el usuario agregado al rol fijo (db_datareader)

Comprueba que el usuario Windows está como miembro del rol fijo.

```
USE QhatuPeru;
GO
```

```
SELECT
    r.name AS RoleName,
    m.name AS MemberUser
FROM sys.database_role_members drm
JOIN sys.database_principals r ON drm.role_principal_id =
r.principal_id
JOIN sys.database_principals m ON drm.member_principal_id =
m.principal_id
WHERE r.name = 'db_datareader';
GO
```

2. Verificar la Asignación de Usuarios a Roles (Versión Joel Soriano Timoteo)

```
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_joel', 'usuario_win_joel');
GO
```

3. Verificación de Permisos Granulares del Rol ventas_readwrite

```
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
```

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

1. Enunciado del ejercicio (Versión modificada)

Se requiere simular un escenario donde un analista pueda consultar información de inventario, pero sin tener acceso a datos sensibles relacionados con precios. Para ello, se deberán crear usuarios y roles específicos, aplicar permisos de lectura y luego emplear **DENY** para bloquear la visualización de las columnas **PrecioProveedor** y **PrecioVenta**.

2. Script de la solución en T-SQL (Versión modificada)

Este script define un rol especializado para el analista, asigna permisos de consulta general sobre tablas de inventario y finalmente utiliza la instrucción **DENY** para restringir el acceso solo a las columnas que contienen información confidencial.

```
USE QhatuPeru;
GO

-------------------------------------------------------------------
------------
-- PARTE 1: CREACIÓN DE USUARIO Y ROL ESPECÍFICO
-------------------------------------------------------------------
------------
-- 1.1 Crear un Login y Usuario para el Analista (solo si no
existen)
-- Se genera un login SQL que representará al 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 que manejará los permisos del Analista
CREATE ROLE rol_analista_inventario;
GO

-------------------------------------------------------------------
------------
-- PARTE 2: CONCESIÓN DE PERMISOS AMPLIOS (GRANT)
-------------------------------------------------------------------
------------
-- Permisos de lectura general sobre tablas de inventario y guías.
GRANT SELECT ON ARTICULO TO rol_analista_inventario;
GRANT SELECT ON GUIA_DETALLE TO rol_analista_inventario;

-- Lectura adicional de tablas auxiliares.
GRANT SELECT ON TIENDA TO rol_analista_inventario;
GRANT SELECT ON LINEA TO rol_analista_inventario;
GO

-------------------------------------------------------------------
------------
-- PARTE 3: RESTRICCIÓN DE PERMISOS SENSIBLES (DENY)
-------------------------------------------------------------------
------------
-- Impedir que el analista vea la columna PrecioProveedor de
ARTICULO.
DENY SELECT ON ARTICULO(PrecioProveedor)
TO rol_analista_inventario;
GO

-- Impedir acceso a la columna PrecioVenta de GUIA_DETALLE.
DENY SELECT ON GUIA_DETALLE(PrecioVenta)
TO rol_analista_inventario;
GO

-------------------------------------------------------------------
------------
-- PARTE 4: ASIGNACIÓN DEL USUARIO AL ROL
```

```
-------------------------------------------------------------------
------------
ALTER ROLE rol_analista_inventario
ADD MEMBER usuario_analista;
GO
```

3. Justificación técnica de la solución aplicada (Versión modificada)

Elemento T-SQL	Justificación Técnica de Seguridad
CREATE ROLE rol_analista_inventario	Permite agrupar y administrar permisos desde un único punto, simplificando el control de acceso.
GRANT SELECT ON Tabla TO Role	Brinda la capacidad de consultar datos de inventario que el analista realmente necesita para su función.
DENY SELECT ON Tabla(Columna)	Es la pieza fundamental del control. DENY siempre tiene prioridad sobre cualquier GRANT. Esto asegura que, aunque el rol tenga permiso para consultar la tabla completa, no podrá ver las columnas marcadas como confidenciales.
Simulación de acceso	Cualquier intento de ejecutar `SELECT *` sobre estas tablas fallará, debido al DENY aplicado a nivel de columna, garantizando protección de información sensible.

4. Explicación de las buenas prácticas utilizadas en el proyecto (Versión modificada)

1. **Regla de denegación explícita (DENY primero):**
   SQL Server evalúa primero los permisos más restrictivos. Esto hace que aplicar **DENY a nivel de columna** sea una estrategia sólida para proteger campos críticos incluso si existen otros permisos amplios a nivel de tabla.

2. **Aplicación del Principio de Privilegios Mínimos (PoLP):**
   El rol creado (rol_analista_inventario) otorga únicamente los accesos necesarios para las tareas del analista. No se emplean roles fijos, lo que evita otorgar permisos innecesarios sobre todo el esquema.

3. **Control de Seguridad Granular:**
   Al trabajar directamente con permisos por columna, se garantiza que solo se limita

el acceso estrictamente necesario, permitiendo que el usuario siga consultando inventario sin exponer información monetaria o de proveedores.

Consultas de Verificación — Proyecto 4 (GRANT / DENY / REVOKE)

```
USE QhatuPeru;

GO



---------------------------------------------

-- 1A. Verificar la existencia del ROL

---------------------------------------------

SELECT

    name AS RoleName,

    type_desc

FROM sys.database_principals

WHERE name = 'rol_analista_inventario';

GO



---------------------------------------------

-- 1B. Verificar 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

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  -- Columna
específica con DENY

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
```

3. Prueba de Acceso (Demostración del DENY)

Estas pruebas demuestran que el DENY funciona correctamente para el usuario **usuario_analista**.

Puedes ejecutarlas tal cual.

🔹 A. Prueba Exitosa – Acceso PERMITIDO

El analista **sí puede** ver las columnas que NO están bloqueadas:
 CodArticulo, DescripcionArticulo, StockActual, etc.

```
EXECUTE AS USER = 'usuario_analista';

GO


-- ✔️ Debe funcionar (solo columnas NO denegadas)

SELECT

    CodArticulo,

    DescripcionArticulo,

    StockActual

FROM ARTICULO;

GO


REVERT;

GO
```

✅ Resultado esperado

La consulta debe mostrar filas sin problema.
No deben aparecer las columnas PrecioProveedor ni PrecioVenta.

🔹 B. Prueba Fallida – Acceso DENEGADO (DEBE FALLAR)

Aquí el analista intenta acceder a la columna **PrecioProveedor**, que tiene un **DENY SELECT**.

```
EXECUTE AS USER = 'usuario_analista';

GO


-- ❌ DEBE FALLAR: DENY SELECT en PrecioProveedor

SELECT

    CodArticulo,

    DescripcionArticulo,

    PrecioProveedor   -- Causa el error

FROM ARTICULO;

GO


REVERT;

GO
```

❌ Resultado esperado

SQL Server debe devolver un error como:

Msg 230, Level 14, State 1

The SELECT permission was denied on the column 'PrecioProveedor' of the object 'ARTICULO'.

Esto demuestra que:

- **DENY tiene prioridad sobre GRANT**
- El analista **no puede** ver precios aunque tenga SELECT en la tabla completa
- La implementación del PoLP es correcta

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

---

1. Enunciado del ejercicio

Activar Transparent Data Encryption en la base de datos **QhatuPeru** para proteger los archivos físicos (MDF/LDF) en reposo. Crear la clave maestra de la base de datos, generar un certificado en el servidor, crear la Database Encryption Key cifrada por ese certificado y finalmente encender el cifrado de la base.

---

2. Script de la solución en T-SQL

Nota: TDE sigue una jerarquía de cifrado: Service Master Key → Database Master Key → Certificado → Database Encryption Key (DEK). A

continuación el script que crea los objetos necesarios y habilita TDE.

```
-- PARTE 0: Ejecutar en master

USE master;

GO



-------------------------------------------------------------------------
------------

-- PARTE 1: CREAR LA MASTER KEY DE LA BASE DE DATOS (CMK)

-- (Protege certificados y claves; hacer backup inmediatamente)

-------------------------------------------------------------------------
------------

CREATE MASTER KEY ENCRYPTION BY PASSWORD =
'TuContraseñaMaestraFuerte!';

GO



-------------------------------------------------------------------------
------------

-- PARTE 2: CREAR EL CERTIFICADO DEL SERVIDOR PARA TDE

-- (El certificado protegerá la DEK y se guarda en master)

-------------------------------------------------------------------------
------------

CREATE CERTIFICATE TdeCert_QhatuPeru

    WITH SUBJECT = 'Certificado TDE para QhatuPeru';

GO
```

```
-------------------------------------------------------------------
------------

-- PARTE 3: CREAR LA DATABASE ENCRYPTION KEY (DEK) EN LA BASE
QhatuPeru

-- (Clave simétrica que cifra las páginas usando AES_256; está
protegida por el certificado)

-------------------------------------------------------------------
------------

USE QhatuPeru;

GO



CREATE DATABASE ENCRYPTION KEY

WITH ALGORITHM = AES_256

ENCRYPTION BY SERVER CERTIFICATE TdeCert_QhatuPeru;

GO



-------------------------------------------------------------------
------------

-- PARTE 4: ACTIVAR TDE EN LA BASE DE DATOS

-- (Inicia el proceso de cifrado en segundo plano)

-------------------------------------------------------------------
------------

ALTER DATABASE QhatuPeru

SET ENCRYPTION ON;

GO
```

```
-------------------------------------------------------------------------
------------

-- PARTE 5: CONSULTA PARA VERIFICAR EL PROGRESO/ESTADO

-------------------------------------------------------------------------
------------

USE master;

GO


SELECT

    db.name,

    db.is_encrypted,

    dek.encryption_state,

    dek.encryption_state_desc,

    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
```

---

3. Justificación técnica de la solución aplicada

- **CREATE MASTER KEY**
  Establece la clave maestra que protege claves privadas y certificados en la instancia; es la raíz para la jerarquía de cifrado. Es crítico hacer copia de seguridad de esta clave.

- **CREATE CERTIFICATE**
  Genera un certificado en la base master que se utiliza para proteger la DEK. Mantener el certificado seguro/respaldado permite restaurar bases cifradas en otra instancia.

- **CREATE DATABASE ENCRYPTION KEY (DEK)**
  Crea la clave simétrica que cifra realmente las páginas de datos y transacciones (niveles de página). Se emplea un algoritmo robusto (AES_256).

- **ALTER DATABASE ... SET ENCRYPTION ON**
  Activa TDE: inicia una tarea que recorre y reescribe las páginas cifradas. El cifrado es transparente para las aplicaciones.

- **sys.dm_database_encryption_keys**
  DMV útil para auditar el estado del cifrado y monitorizar el progreso (percent_complete).

---

4. Explicación de las buenas prácticas utilizadas en el proyecto

1. **Jerarquía de cifrado correcta:** SMK → CMK → Certificado → DEK. Esta estructura garantiza que, para restaurar una BD cifrada, se requieren los elementos superiores (archivo de copia de seguridad de la master key y del certificado).

2. **Separación de claves:** La DEK vive en la base de datos; el certificado que la protege está en master. Así, una copia de seguridad de la BD sola no permite descifrar sin el certificado/clave del servidor.

3. **Protección de datos en reposo:** TDE evita que los archivos MDF/LDF y backups sean legibles si se obtienen por medios no autorizados (p. ej. robo de discos o backups).

4. **Copia de seguridad de claves:** Siempre respaldar el certificado y la master key inmediatamente después de crearlos —práctica crítica—; su pérdida puede volver los datos irrecuperables.

Consultas de Verificación para el Proyecto 5: TDE

1. Verificar la existencia de la Master Key y del Certificado

Ejecuta en la base master:

```
USE master;

GO


-- 1A. Verificar la existencia de la Database Master Key (indicador en symmetric keys)

SELECT

    name AS MasterKeyName,

    key_algorithm AS EncryptionAlgorithm

FROM sys.symmetric_keys

WHERE name LIKE '##MS_DatabaseMasterKey##';

GO


-- 1B. Verificar que el certificado creado exista y cómo se protege su clave privada
```

```
SELECT

    name AS CertificateName,

    pvt_key_encryption_type_desc AS KeyProtection

FROM sys.certificates

WHERE name = 'TdeCert_QhatuPeru';

GO
```

2. Verificar la DEK y el estado de TDE

Ejecuta en QhatuPeru:

```
USE QhatuPeru;

GO


SELECT

    DB_NAME(dek.database_id) AS DatabaseName,

    CASE WHEN db.is_encrypted = 1 THEN 'Yes' ELSE 'No' END AS
IsDatabaseEncrypted,

    dek.encryption_state,           -- 3 = Encrypted, 2 =
Decrypting, 1 = Encrypting

    dek.encryption_state_desc,

    dek.percent_complete,

    dek.key_algorithm,

    dek.key_length

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**
- encryption_state: **3**
- encryption_state_desc: **Encrypted**
- percent_complete: **100** (o cerca si consultas inmediatamente tras activar)

Proyecto 6: Implementación de Always Encrypted

(columna de datos sensibles)**

---

1. Enunciado del ejercicio

Configurar Always Encrypted para proteger la columna **PrecioProveedor** (o una nueva columna **PrecioProveedor_ENC**) usando:

- Una **Column Master Key (CMK)** almacenada en el Certificate Store de Windows o Azure Key Vault.
  (Esta parte NO puede hacerse con T-SQL).
- Una **Column Encryption Key (CEK)** generada en SQL Server.
- Alteración de la columna para que almacene los datos cifrados.

---

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

Always Encrypted funciona en 3 etapas:

Fase 1 – Creación de la CMK (Fuera de T-SQL)

La CMK se genera en:

- Windows Certificate Store, **o**
- Azure Key Vault.

Esta parte NO se puede crear con T-SQL, pero sí se registra en SQL Server.

Ejemplo (solo registro dentro de SQL Server):

```
ALTER DATABASE SCOPED CONFIGURATION
SET PARAMETERIZATION FORCED;
```

(La creación real de la CMK se hace con SQL Server Management Studio o PowerShell).

Fase 2 – Creación de la CEK (Dentro de SQL Server)

Primero se limpia cualquier intento anterior:

```
USE QhatuPeru;
GO

------------------------------------------------------------

-- 0. LIMPIEZA PREVIA
```

```
-----------------------------------------------------------------


-- Eliminar CEK si existe

IF EXISTS (SELECT 1 FROM sys.column_encryption_keys WHERE name =
'CEK_PrecioProveedor')

BEGIN

    DROP COLUMN ENCRYPTION KEY [CEK_PrecioProveedor];

END

GO


-- Eliminar CMK si existe (solo si no hay CEKs asociadas)

IF EXISTS (SELECT 1 FROM sys.column_master_keys WHERE name =
'CMK_PrecioProveedor')

BEGIN

    DROP COLUMN MASTER KEY [CMK_PrecioProveedor];

END

GO
```

Registrar la CMK (ya creada fuera del servidor)

```
Ejemplo:

CREATE COLUMN MASTER KEY [CMK_PrecioProveedor]

WITH (

    KEY_STORE_PROVIDER_NAME = 'MSSQL_CERTIFICATE_STORE',
```

```
    KEY_PATH =
'CurrentUser/My/1234567890ABCDEF1234567890ABCDEF12345678'
);
GO
```

> **KEY_PATH** debe ser reemplazado por el *Thumbprint real del certificado*.

Crear la CEK que cifra los datos

```
CREATE COLUMN ENCRYPTION KEY [CEK_PrecioProveedor]
WITH VALUES (
    COLUMN_MASTER_KEY = [CMK_PrecioProveedor],
    ALGORITHM = 'RSA_OAEP',
    ENCRYPTED_VALUE = 0x00 -- Este valor lo genera automáticamente
SSMS
);
GO
```

> En la práctica, este script lo genera automáticamente SSMS al configurar Always Encrypted mediante el asistente.

Fase 3 – Crear o Alterar la columna cifrada

Si deseas **crear una nueva columna**:

```
ALTER TABLE ARTICULO
ADD PrecioProveedor_ENC VARBINARY(256)
```

```
    COLLATE Latin1_General_BIN2
    ENCRYPTED WITH (
        ENCRYPTION_TYPE = DETERMINISTIC,
        ALGORITHM = 'AEAD_AES_256_CBC_HMAC_SHA_256',
        COLUMN_ENCRYPTION_KEY = CEK_PrecioProveedor
    );
GO
```

Si deseas reemplazar la columna original

Primero eliminas:

```
ALTER TABLE ARTICULO DROP COLUMN PrecioProveedor;
GO
```

Luego agregas la columna cifrada:

```
ALTER TABLE ARTICULO
ADD PrecioProveedor VARBINARY(256)
    COLLATE Latin1_General_BIN2
    ENCRYPTED WITH (
        ENCRYPTION_TYPE = DETERMINISTIC,
        ALGORITHM = 'AEAD_AES_256_CBC_HMAC_SHA_256',
        COLUMN_ENCRYPTION_KEY = CEK_PrecioProveedor
    );
GO
```

---

3. Justificación técnica de la solución aplicada

Elemento T-SQL	Justificación Técnica
Column Master Key (CMK)	Se almacena fuera de SQL Server (Windows Certificate Store o Key Vault). Garantiza separación de funciones: el DBA no puede ver los datos en texto claro.
Column Encryption Key (CEK)	Es la clave simétrica que cifra la columna. SQL Server la almacena cifrada por la CMK.
ENCRYPTED WITH (…)	Define el tipo de cifrado de la columna. Se usa VARBINARY porque los valores cifrados son binarios.
DETERMINISTIC	Ideal cuando se necesita usar la columna en filtros (WHERE), comparaciones o JOINs porque siempre cifra igual un mismo valor.

---

4. Buenas Prácticas del Proyecto

1. **Cifrado de Datos en Uso (Data In Use)**
   Always Encrypted mantiene los datos cifrados durante toda la ejecución.
   SQL Server *nunca ve los datos reales*.

2. **Separación de Responsabilidades (SoD)**
   El DBA no tiene acceso a la CMK.
   La aplicación cliente (con el certificado) es la única que puede descifrar.

3. **Cifrado a Nivel de Columna**
   Ideal para columnas con información altamente sensible (precios secretos, DNI, salarios, tarjetas).

Consultas de Verificación del Proyecto 6

---

1. Verificar la existencia de CMK y CEK

```
USE QhatuPeru;

GO


-- A. Verificar la CMK

SELECT

    cmk.name AS CMK_Name,

    cmk.key_store_provider_name AS Key_Source,

    cmk.key_path AS Certificado_Path

FROM sys.column_master_keys cmk

WHERE cmk.name = 'CMK_PrecioProveedor';

GO
```

```
-- B. Verificar la CEK
SELECT
    cek.name AS CEK_Name,
    cek.column_master_key_id AS CMK_ID
FROM sys.column_encryption_keys cek
WHERE cek.name = 'CEK_PrecioProveedor';
GO
```

---

2. Verificar el estado de cifrado de la columna

```
USE QhatuPeru;
GO

SELECT
    t.name AS TableName,
    c.name AS ColumnName,
    c.encryption_type_desc AS EncryptionType,
    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