UNIVERSIDAD PERUANA LOS ANDES

FACULTAD DE INGENIERÍA

ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMAS Y COMPUTACIÓN

PRACTICA 11

ASIGNATURA: Base de Datos II

ESTUDIANTE: Espejo Quispe Luis Enrique

DOCENTE: Mg. Raúl Fernandez Bejarano

CICLO: V

SECCIÓN: A1

HYO-2025

Proyecto 1: Autenticación: Comparación segura y configuración de logins

1. Enunciado del ejercicio

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

SCRIPT EN T-SQL

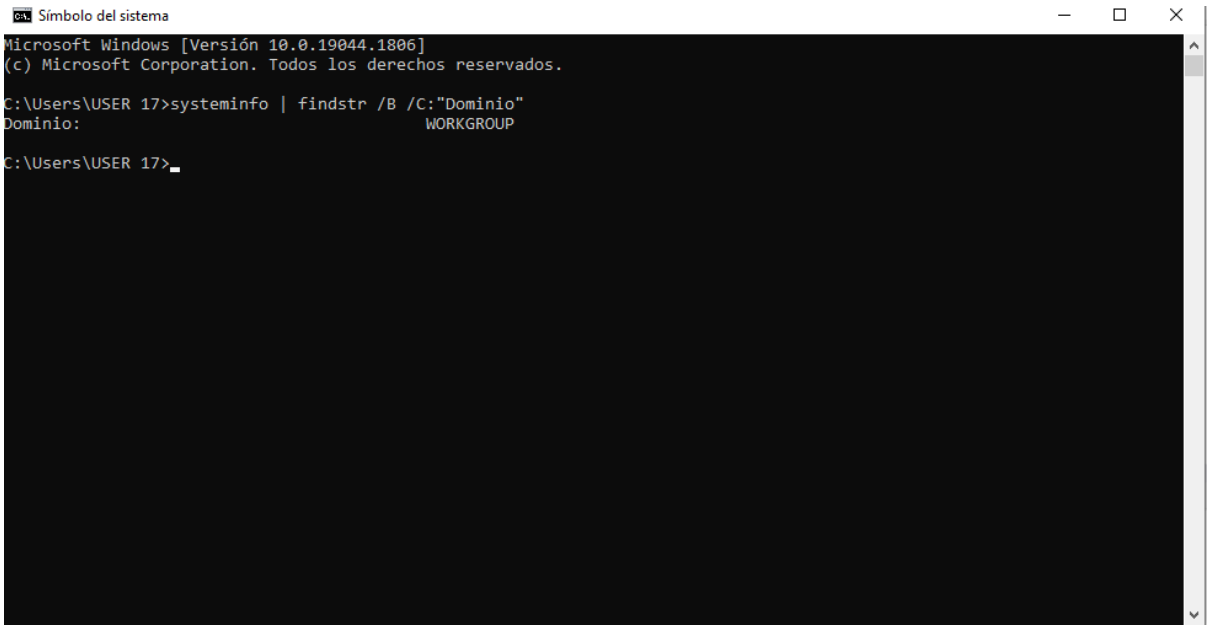
```
-- Paso 1: Crear login con autenticación SQL con políticas de seguridad
CREATE LOGIN login_sql_alumno
WITH PASSWORD = '123456789' MUST_CHANGE,
     CHECK_EXPIRATION = ON,
     CHECK_POLICY = ON,
     DEFAULT_DATABASE = QhatuPeru;
GO
```

AUTENTICACIÓN EN WINDOWS

```
-- Paso 2: Crear login de Windows (simulado - requiere dominio real)
-- NOTA: Este comando requiere un dominio Active Directory real
-- Para simulación, comentamos y mostramos la sintaxis
/*
CREATE LOGIN [DOMAIN\alumno_win]
FROM WINDOWS
WITH DEFAULT_DATABASE = QhatuPeru;
GO
*/

-- Alternativa para demostración: crear otro login SQL simulando Windows
CREATE LOGIN alumno_win_simulado
WITH PASSWORD = 'WindowsUser2025!',
     CHECK_EXPIRATION = OFF,  -- Usuarios Windows típicamente no expiran en SQL
     CHECK_POLICY = ON,
     DEFAULT_DATABASE = QhatuPeru;
GO
```

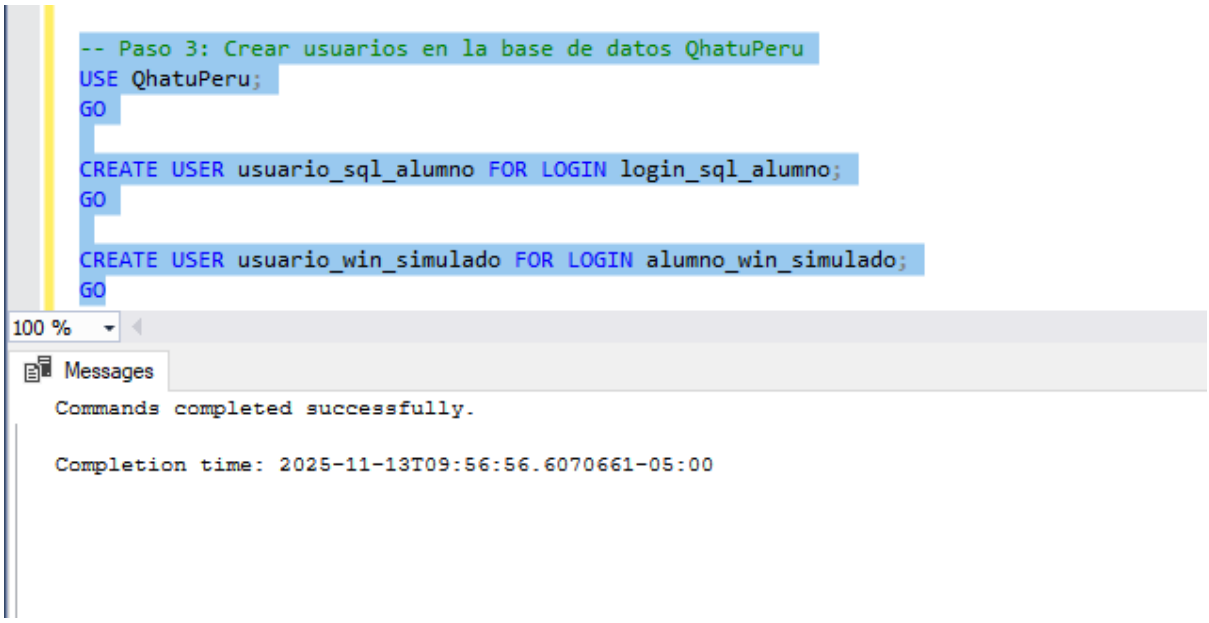
```
100 %
Messages
Commands completed successfully.

Completion time: 2025-11-13T09:56:56.6070661-05:00
```

OBSERVACIÓN: NO SE PUEDE CREAR LOGIN UN NUEVO USUARIO VÍA AUTENTICACIÓN DE WINDOWS DEBIDO A QUE NO EXISTE UN DOMINIO DE PERTENENCIA



EL DOMINIO" WORKGROUP" INDICA QUE EL EQUIPO NO TIENE DOMINIO

CREACIÓN DE USUARIOS



POLÍTICAS APLICADAS

```sql
-- Paso 4: Verificar políticas aplicadas
SELECT
    name AS LoginName,
    type_desc AS AuthenticationType,
    is_policy_checked AS PolicyChecked,
    is_expiration_checked AS ExpirationChecked,
    create_date AS CreatedDate,
    modify_date AS LastModified
FROM sys.sql_logins
WHERE name IN ('login_sql_alumno', 'alumno_win_simulado');
GO
```

100 %

Results | Messages

	LoginName	AuthenticationType	PolicyChecked	ExpirationChecked	CreatedDate	LastModified
1	login_sql_alumno	SQL_LOGIN	1	1	2025-11-13 09:50:21.747	2025-11-13 09:50:21.760
2	alumno_win_simulado	SQL_LOGIN	1	0	2025-11-13 09:50:55.707	2025-11-13 09:50:55.720

```sql
-- Paso 6: Verificar la política de contraseña intentando crear una débil
-- Esto fallará por política
BEGIN TRY
    CREATE LOGIN login_debil
    WITH PASSWORD = '123',
         CHECK_POLICY = ON;
END TRY
BEGIN CATCH
    PRINT '=== PRUEBA DE POLÍTICA DE CONTRASEÑA ===';
    PRINT 'ERROR ESPERADO: ' + ERROR_MESSAGE();
    PRINT 'La política rechazó la contraseña débil correctamente.';
END CATCH;
GO
```

100 %

Messages

```
=== PRUEBA DE POLÍTICA DE CONTRASEÑA ===
ERROR ESPERADO: The server principal 'login_debil' already exists.
La política rechazó la contraseña débil correctamente.

Completion time: 2025-11-13T10:01:26.8831499-05:00
```

JUSTIFICACIÓN TÉCNICA

La solución implementa autenticación en modo mixto (SQL y Windows) con las siguientes consideraciones técnicas:

CHECK_POLICY = ON: Aplica las políticas de contraseñas de Windows Server, rechazando contraseñas débiles, requiriendo complejidad (mayúsculas, minúsculas, números, símbolos).

CHECK_EXPIRATION = ON: Fuerza la expiración de contraseñas según políticas del sistema, obligando cambios periódicos.

MUST_CHANGE: Requiere que el usuario cambie la contraseña en el primer login, previniendo el uso de contraseñas temporales indefinidamente.

Diferencia SQL vs Windows Auth: Los logins SQL almacenan hashes de contraseña en SQL Server, mientras que Windows Auth delega la autenticación a Active Directory, siendo más segura en entornos corporativos.

BUENAS PRÁCTICAS

Contraseñas robustas: Se utilizan contraseñas que cumplen con requisitos de complejidad (mínimo 8 caracteres, combinación de tipos).

Principio de defensa en profundidad: Múltiples capas de seguridad (política de contraseñas + expiración + cambio obligatorio).

Autenticación Windows preferida: En entornos corporativos, Windows Auth es superior porque centraliza la gestión de identidades y soporta Kerberos.

Segregación de cuentas: Crear usuarios de base de datos separados de los logins permite control granular de permisos.

Auditoría: El script incluye consultas para verificar las configuraciones, facilitando auditorías de seguridad.

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.

SCRIPT EN T-SQL

CONFIGURACIÓN ACTUAL DEL SERVIDOR

```
SELECT
    name AS ConfigOption,
    value AS CurrentValue,
    value_in_use AS ActiveValue,
    description
FROM sys.configurations
WHERE name IN (
    'xp_cmdshell',
    'contained database authentication',
    'Ole Automation Procedures',
    'Ad Hoc Distributed Queries',
    'Database Mail XPs'
)
ORDER BY name;
GO
```

100 %

Results | Messages

	ConfigOption	CurrentValue	ActiveValue	description
1	Ad Hoc Distributed Queries	0	0	Enable or disable Ad Hoc Distributed Queries
2	contained database authentication	0	0	Enables contained databases and contained authe...
3	Database Mail XPs	0	0	Enable or disable Database Mail XPs
4	Ole Automation Procedures	0	0	Enable or disable Ole Automation Procedures
5	xp_cmdshell	0	0	Enable or disable command shell

DESHABILITAR XP_CMDSHELL

```
-- Paso 2: Deshabilitar xp_cmdshell (alta prioridad de seguridad)
EXEC sp_configure 'show advanced options', 1;
RECONFIGURE;
GO

EXEC sp_configure 'xp_cmdshell', 0;
RECONFIGURE;
GO

PRINT 'xp_cmdshell DESHABILITADO correctamente';
GO
```

100 %

Messages

```
Configuration option 'show advanced options' changed from 0 to 1. Run the RECONFIGURE statement to install.
Configuration option 'xp_cmdshell' changed from 0 to 0. Run the RECONFIGURE statement to install.
xp_cmdshell DESHABILITADO correctamente

Completion time: 2025-11-13T10:06:58.6580365-05:00
```

CONTAINED DATABASE AUTHENTICATION

```
-- Paso 3: Revisar y documentar Contained Database Authentication
-- Verificar estado actual
DECLARE @containedAuth INT;

SELECT @containedAuth = CONVERT(INT, value)
FROM sys.configurations
WHERE name = 'contained database authentication';

IF @containedAuth = 1
BEGIN
    PRINT 'ADVERTENCIA: Contained Database Authentication está HABILITADO';
    PRINT 'Recomendación: Mantener deshabilitado a menos que sea específicamente requerido';
END
ELSE
BEGIN
    PRINT 'Contained Database Authentication está DESHABILITADO (configuración segura)';
END
GO
```

100 %

Messages

```
Contained Database Authentication está DESHABILITADO (configuración segura)

Completion time: 2025-11-13T10:08:53.5313172-05:00
```

CREDENCIAL DE SQL AGENT

```
-- Paso 4: Crear credencial para SQL Agent con acceso al OS
-- Esta credencial se usará para jobs que necesiten acceso al sistema operativo
CREATE CREDENTIAL [SQLAgentOSAccess_Credential]
WITH IDENTITY = N'DOMAIN\SQLServiceAccount',  -- Cuenta de servicio de dominio
     SECRET = N'SecurePassword2025!';  -- En producción: usar contraseña real
GO

PRINT 'Credencial SQLAgentOSAccess_Credential creada correctamente';
GO
```

100 %

Messages

```
Credencial SQLAgentOSAccess_Credential creada correctamente

Completion time: 2025-11-13T10:09:45.2331443-05:00
```

PROXY ASOCIADO

```sql
CREATE CREDENTIAL [SQLAgentOSAccess_Credential]
WITH IDENTITY = N'.\UsuarioLocal',
SECRET = N'ContraseñaSegura123';


-- Crear proxy asociado a la credencial
USE msdb;
EXEC msdb.dbo.sp_add_proxy
    @proxy_name = N'OSAccessProxy',
    @credential_name = N'SQLAgentOSAccess_Credential',
    @enabled = 1,
    @description = N'Proxy para jobs que requieren acceso al sistema operativo';

PRINT 'Proxy OSAccessProxy creado y configurado para CmdExec';
GO
```

100 %

Messages

```
Commands completed successfully.

Completion time: 2025-11-13T10:18:04.5972522-05:00
```

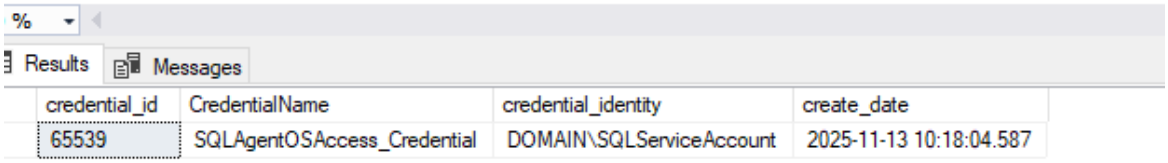
```sql
-- Paso 6: Generar reporte de configuración final
PRINT '';
PRINT '=== REPORTE DE CONFIGURACIÓN DE SEGURIDAD ===';
PRINT 'Fecha: ' + CONVERT(VARCHAR, GETDATE(), 120);
PRINT '';

SELECT
    name AS [Parámetro de Seguridad],
    CASE value_in_use
        WHEN 0 THEN 'DESHABILITADO (Seguro)'
        WHEN 1 THEN 'HABILITADO (Revisar necesidad)'
    END AS [Estado],
    description AS [Descripción]
FROM sys.configurations
WHERE name IN (
    'xp_cmdshell',
    'contained database authentication',
    'Ole Automation Procedures',
    'Ad Hoc Distributed Queries'
)
ORDER BY value_in_use DESC, name;
GO
```

00 %

Results | Messages

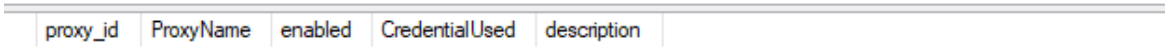
	Parámetro de Seguridad	Estado	Descripción
1	Ad Hoc Distributed Queries	DESHABILITADO (Seguro)	Enable or disable Ad Hoc Distributed Queries
2	contained database authentication	DESHABILITADO (Seguro)	Enables contained databases and contained authe...
3	Ole Automation Procedures	DESHABILITADO (Seguro)	Enable or disable Ole Automation Procedures
4	xp_cmdshell	DESHABILITADO (Seguro)	Enable or disable command shell

```sql
-- Paso 8: Verificar credencial y proxy creados
SELECT
    c.credential_id,
    c.name AS CredentialName,
    c.credential_identity,
    c.create_date
FROM sys.credentials c
WHERE c.name = 'SQLAgentOSAccess_Credential';
GO

SELECT
    p.proxy_id,
    p.name AS ProxyName,
    p.enabled,
    c.name AS CredentialUsed,
    p.description
FROM msdb.dbo.sysproxies p
INNER JOIN sys.credentials c ON p.credential_id = c.credential_id
WHERE p.name = 'OSAccessProxy';
GO
```



Results | Messages

credential_id	CredentialName	credential_identity	create_date
65539	SQLAgentOSAccess_Credential	DOMAIN\SQLServiceAccount	2025-11-13 10:18:04.587



proxy_id	ProxyName	enabled	CredentialUsed	description

JUSTIFICACIÓN TÉCNICA

La configuración de seguridad del servidor es crítica para prevenir ataques:

Deshabilitar xp_cmdshell: Esta stored procedure extendida permite ejecutar comandos del SO desde T-SQL, representando un vector de ataque mayor. Se deshabilita por defecto y solo se habilita temporalmente cuando sea estrictamente necesario.

Contained Database Authentication: Permite usuarios que existan solo en la base de datos sin login en el servidor. Se deshabilita para mantener gestión centralizada de identidades.

Credenciales vs Logins: Las credenciales almacenan autenticación para recursos externos (Windows, Azure), mientras los logins son para acceso a SQL Server. Las

credenciales permiten a SQL Agent ejecutar trabajos con identidades específicas sin exponer contraseñas.

Proxies de SQL Agent: Abstraen las credenciales, permitiendo asignar permisos granulares a jobs sin dar acceso directo a las credenciales subyacentes.

BUENAS PRÁCTICAS

Principio de mínimo privilegio: Solo se habilitan características necesarias, reduciendo la superficie de ataque.

Segregación de funciones: Los proxies permiten que diferentes jobs usen diferentes identidades según necesidad.

Auditoría continua: Scripts de verificación permiten monitorear el estado de seguridad regularmente.

Documentación automatizada: El reporte generado facilita auditorías de compliance (SOX, PCI-DSS, etc.).

Cuentas de servicio de dominio: Usar cuentas de dominio (no locales) facilita gestión centralizada y rotación de contraseñas.

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.

SCRIPT EN T-SQL

CREACIÓN DE USUARIOS

```
-- Paso 1: Crear usuarios de prueba para demostración
USE master;
GO

CREATE LOGIN usuario_ventas_01 WITH PASSWORD = 'Ventas01Pass!';
CREATE LOGIN usuario_ventas_02 WITH PASSWORD = 'Ventas02Pass!';
CREATE LOGIN usuario_consulta WITH PASSWORD = 'ConsultaPass!';
GO

USE QhatuPeru;
GO

CREATE USER usuario_ventas_01 FOR LOGIN usuario_ventas_01;
CREATE USER usuario_ventas_02 FOR LOGIN usuario_ventas_02;
CREATE USER usuario_consulta FOR LOGIN usuario_consulta;
GO
```

```
100 %
Messages
Commands completed successfully.

Completion time: 2025-11-13T10:22:24.6078764-05:00
```

```
-- Paso 2: Crear rol personalizado ventas_readwrite
CREATE ROLE ventas_readwrite;
GO

PRINT 'Rol personalizado ventas_readwrite creado';
GO
```

100 %

Messages

```
Rol personalizado ventas_readwrite creado

Completion time: 2025-11-13T10:23:15.6768712-05:00
```

PERMISOS AL ROL PERSONALIZADO

```
-- Paso 3: Otorgar permisos específicos al rol personalizado
-- SELECT en tablas de ventas
GRANT SELECT ON OBJECT::dbo.GUIA_ENVIO TO ventas_readwrite;
GRANT SELECT ON OBJECT::dbo.GUIA_DETALLE TO ventas_readwrite;
GRANT SELECT ON OBJECT::dbo.TIENDA TO ventas_readwrite;
GRANT SELECT ON OBJECT::dbo.TRANSPORTISTA TO ventas_readwrite;
GRANT SELECT ON OBJECT::dbo.ARTICULO TO ventas_readwrite;

-- INSERT en tablas de ventas
GRANT INSERT ON OBJECT::dbo.GUIA_ENVIO TO ventas_readwrite;
GRANT INSERT ON OBJECT::dbo.GUIA_DETALLE TO ventas_readwrite;

-- UPDATE en tablas de ventas
GRANT UPDATE ON OBJECT::dbo.GUIA_ENVIO TO ventas_readwrite;
GRANT UPDATE ON OBJECT::dbo.GUIA_DETALLE TO ventas_readwrite;

PRINT 'Permisos otorgados al rol ventas_readwrite';
GO
```

100 %

Messages

```
Permisos otorgados al rol ventas_readwrite

Completion time: 2025-11-13T10:23:34.0763476-05:00
```

ASIGNACIÓN DE USUARIOS AL ROL

```sql
-- Paso 4: Asignar usuarios al rol personalizado
ALTER ROLE ventas_readwrite ADD MEMBER usuario_ventas_01;
ALTER ROLE ventas_readwrite ADD MEMBER usuario_ventas_02;
GO

PRINT 'Usuarios agregados al rol ventas_readwrite';
GO
```

00 %

Messages

```
Usuarios agregados al rol ventas_readwrite

Completion time: 2025-11-13T10:23:58.8400407-05:00
```

VERIFICACIÓN DE PERMISOS

```sql
-- Paso 6: Verificar permisos efectivos de cada rol
-- Permisos del rol personalizado
SELECT
    'ventas_readwrite' AS RoleName,
    USER_NAME(grantee_principal_id) AS Grantee,
    OBJECT_NAME(major_id) AS ObjectName,
    permission_name AS Permission,
    state_desc AS PermissionState
FROM sys.database_permissions
WHERE grantee_principal_id = DATABASE_PRINCIPAL_ID('ventas_readwrite')
    AND major_id > 0
ORDER BY OBJECT_NAME(major_id), permission_name;
GO
```

00 %

Results | Messages

	RoleName	Grantee	ObjectName	Permission	PermissionState
1	ventas_readwrite	ventas_readwrite	ARTICULO	SELECT	GRANT
2	ventas_readwrite	ventas_readwrite	GUIA_DETALLE	INSERT	GRANT
3	ventas_readwrite	ventas_readwrite	GUIA_DETALLE	SELECT	GRANT
4	ventas_readwrite	ventas_readwrite	GUIA_DETALLE	UPDATE	GRANT
5	ventas_readwrite	ventas_readwrite	GUIA_ENVIO	INSERT	GRANT
6	ventas_readwrite	ventas_readwrite	GUIA_ENVIO	SELECT	GRANT
7	ventas_readwrite	ventas_readwrite	GUIA_ENVIO	UPDATE	GRANT
8	ventas_readwrite	ventas_readwrite	TIENDA	SELECT	GRANT
9	ventas_readwrite	ventas_readwrite	TRANSPORTISTA	SELECT	GRANT

```sql
-- Paso 8: Verificar miembros de cada rol
SELECT
    r.name AS RoleName,
    r.type_desc AS RoleType,
    m.name AS MemberName,
    m.type_desc AS MemberType
FROM sys.database_role_members rm
INNER JOIN sys.database_principals r ON rm.role_principal_id = r.principal_id
INNER JOIN sys.database_principals m ON rm.member_principal_id = m.principal_id
WHERE r.name IN ('ventas_readwrite', 'db_datareader')
ORDER BY r.name, m.name;
GO
```

00 %

Results | Messages

	RoleName	RoleType	MemberName	MemberType
1	db_datareader	DATABASE_ROLE	usuario_consulta	SQL_USER
2	ventas_readwrite	DATABASE_ROLE	usuario_ventas_01	SQL_USER
3	ventas_readwrite	DATABASE_ROLE	usuario_ventas_02	SQL_USER

JUSTIFICACIÓN TÉCNICA

La creación de roles personalizados ofrece ventajas significativas: Roles personalizados vs fijos: Los roles fijos (db_datareader, db_datawriter) otorgan permisos amplios a toda la base de datos. Los roles personalizados permiten permisos granulares sobre objetos específicos, implementando mejor el principio de mínimo privilegio. Herencia de permisos: Los usuarios hereden permisos del rol, facilitando administración. Cambiar permisos del rol afecta automáticamente a todos sus miembros. Segregación funcional: El rol ventas_readwrite encapsula exactamente los permisos que necesita el personal de ventas, sin exponer datos de compras, proveedores o configuración. Escalabilidad: Agregar nuevos usuarios al equipo de ventas solo requiere: ALTER ROLE ventas_readwrite ADD MEMBER nuevo_usuario;

BUENAS PRÁCTICAS

Nomenclatura descriptiva: El nombre "ventas_readwrite" indica claramente el propósito y permisos del rol.

Documentación inline: Comentarios y PRINTs facilitan entender qué hace cada sección.

Permisos granulares: Solo se otorgan permisos necesarios por tabla y operación (SELECT vs INSERT vs UPDATE).

Auditoría incorporada: Queries para verificar permisos efectivos facilitan revisiones de seguridad.

Separación de lectura/escritura: El rol permite lectura amplia (consultas) pero escritura limitada (solo ventas), reduciendo riesgo de corrupción de datos.

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.

SCRIPT EN T-SQL

```
-- Paso 1: Crear login y usuario para analista de inventario
USE master;
GO

CREATE LOGIN analista_inventario WITH PASSWORD = 'Inventario2025';
GO

USE QhatuPeru;
GO

CREATE USER analista_inventario FOR LOGIN analista_inventario;
GO
```

```
0 %
Messages
Commands completed successfully.

Completion time: 2025-11-13T10:29:48.0714481-05:00
```

ROL Y ASIGNACIÓN DE PERMISOS

```sql
-- Paso 2: Crear rol para analistas de inventario
CREATE ROLE rol_analista_inventario;
GO

-- Paso 3: Otorgar permisos de lectura en tablas de inventario
GRANT SELECT ON OBJECT::dbo.ARTICULO TO rol_analista_inventario;
GRANT SELECT ON OBJECT::dbo.LINEA TO rol_analista_inventario;
GRANT SELECT ON OBJECT::dbo.PROVEEDOR TO rol_analista_inventario;
GRANT SELECT ON OBJECT::dbo.ORDEN_COMPRA TO rol_analista_inventario;
GRANT SELECT ON OBJECT::dbo.ORDEN_DETALLE TO rol_analista_inventario;
GO

PRINT 'Permisos SELECT otorgados en tablas de inventario';
GO
```

```
Messages
Permisos SELECT otorgados en tablas de inventario

Completion time: 2025-11-13T10:30:17.5203107-05:00
```

DENY PARA DENEGAR ACCESO

```sql
-- Paso 4: DENY explícito en columnas de precio (seguridad crítica)
-- Denegar acceso a PrecioProveedor en tabla ARTICULO
DENY SELECT ON OBJECT::dbo.ARTICULO (PrecioProveedor) TO rol_analista_inventario;
GO

-- Denegar acceso a PrecioCompra en tabla ORDEN_DETALLE
DENY SELECT ON OBJECT::dbo.ORDEN_DETALLE (PrecioCompra) TO rol_analista_inventario;
GO

-- Denegar acceso a PrecioVenta en tabla GUIA_DETALLE
DENY SELECT ON OBJECT::dbo.GUIA_DETALLE (PrecioVenta) TO rol_analista_inventario;
GO

PRINT 'DENY aplicado en columnas de precios';
GO
```

```
Messages
DENY aplicado en columnas de precios

Completion time: 2025-11-13T10:30:50.2328113-05:00
```

```sql
-- Paso 5: Asignar usuario al rol
ALTER ROLE rol_analista_inventario ADD MEMBER analista_inventario;
GO

-- Paso 6: Crear vista segura para análisis de inventario sin precios
CREATE VIEW vw_Inventario_SinPrecios
AS
SELECT
    a.CodArticulo,
    a.DescripcionArticulo,
    a.Presentacion,
    a.StockActual,
    a.StockMinimo,
    a.Descontinuado,
    l.NomLinea,
    l.Descripcion AS DescripcionLinea,
    p.NomProveedor,
    p.Ciudad AS CiudadProveedor,
    -- NO incluir: PrecioProveedor
    CASE
        WHEN a.StockActual <= a.StockMinimo THEN 'REORDEN NECESARIO'
        WHEN a.StockActual <= a.StockMinimo * 1.5 THEN 'STOCK BAJO'
        ELSE 'STOCK NORMAL'
    END AS EstadoInventario
FROM dbo.ARTICULO a
INNER JOIN dbo.LINEA l ON a.CodLinea = l.CodLinea
INNER JOIN dbo.PROVEEDOR p ON a.CodProveedor = p.CodProveedor;
GO
```

100 %

Messages

```
Commands completed successfully.

Completion time: 2025-11-13T10:31:37.0771099-05:00
```

VERIFICACIÓN

```sql
-- Paso 7: Verificar permisos efectivos del rol
SELECT
    perm.class_desc AS PermissionClass,
    OBJECT_NAME(perm.major_id) AS ObjectName,
    COL_NAME(perm.major_id, perm.minor_id) AS ColumnName,
    perm.permission_name AS Permission,
    perm.state_desc AS PermissionState,
    USER_NAME(perm.grantee_principal_id) AS GrantedTo
FROM sys.database_permissions perm
WHERE grantee_principal_id = DATABASE_PRINCIPAL_ID('rol_analista_inventario')
ORDER BY
    perm.state_desc DESC,  -- DENY primero
    OBJECT_NAME(perm.major_id),
    COL_NAME(perm.major_id, perm.minor_id);
GO
```

00 %

Results | Messages

	PermissionClass	ObjectName	ColumnName	Permission	PermissionState	GrantedTo
1	OBJECT_OR_COLUMN	ARTICULO	NULL	SELECT	GRANT	rol_analista_inventario
2	OBJECT_OR_COLUMN	LINEA	NULL	SELECT	GRANT	rol_analista_inventario
3	OBJECT_OR_COLUMN	ORDEN_COMPRA	NULL	SELECT	GRANT	rol_analista_inventario
4	OBJECT_OR_COLUMN	ORDEN_DETALLE	NULL	SELECT	GRANT	rol_analista_inventario
5	OBJECT_OR_COLUMN	PROVEEDOR	NULL	SELECT	GRANT	rol_analista_inventario
6	OBJECT_OR_COLUMN	vw_Inventario_SinPrecios	NULL	SELECT	GRANT	rol_analista_inventario
7	OBJECT_OR_COLUMN	ARTICULO	PrecioProveedor	SELECT	DENY	rol_analista_inventario
8	OBJECT_OR_COLUMN	GUIA_DETALLE	PrecioVenta	SELECT	DENY	rol_analista_inventario
9	OBJECT_OR_COLUMN	ORDEN_DETALLE	PrecioCompra	SELECT	DENY	rol_analista_inventario

JUSTIFICACIÓN TÉCNICA

El control granular de acceso mediante GRANT/DENY/REVOKE es fundamental para seguridad: DENY a nivel columna: SQL Server permite denegar acceso a columnas específicas dentro de una tabla, perfecto para ocultar información sensible (precios) mientras se permite acceso al resto de datos. Precedencia de DENY: DENY siempre prevalece sobre GRANT. Si un usuario recibe GRANT por un rol y DENY por otro, el DENY gana. Esto permite "bloqueos de seguridad" que no pueden sobrepasarse accidentalmente. REVOKE vs DENY: REVOKE elimina un permiso (vuelve a estado neutro), mientras DENY bloquea activamente. REVOKE es apropiado para remover acceso temporal; DENY para restricciones de seguridad explícitas. Vistas como capa de seguridad: La vista vw_Inventario_SinPrecios proporciona una interfaz limpia que excluye columnas sensibles, facilitando consultas a usuarios restringidos sin errores.

BUENAS PRÁCTICAS

Principio de mínimo privilegio: El analista solo ve lo necesario para su trabajo (inventario), sin acceso a información financiera (precios).

Defensa en profundidad: Múltiples capas de seguridad (DENY en columnas + vista filtrada) aseguran que incluso si una falla, la otra protege la segregación de datos sensibles

Los precios se consideran información financiera sensible, segregada del análisis operativo de inventario. Documentación de jerarquía: El script documenta claramente la precedencia DENY > GRANT > REVOKE para facilitar comprensión y auditorías. Vistas como abstracción: Proporcionar vistas seguras simplifica el acceso para usuarios finales y reduce errores de consulta. Verificación programática: Uso de HAS_PERMS_BY_NAME permite validar permisos sin necesidad de login/logout constante.

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.

SCRIPT EN T-SQL

```
USE master;
GO

-- Paso 1: Verificar si existe Service Master Key
-- (creada automáticamente en instalación de SQL Server)
IF NOT EXISTS (SELECT * FROM sys.symmetric_keys WHERE name = '##MS_ServiceMasterKey##')
BEGIN
    PRINT 'ADVERTENCIA: Service Master Key no existe';
END
ELSE
BEGIN
    PRINT 'Service Master Key verificada correctamente';
END
GO
```

100 %

Messages

```
Service Master Key verificada correctamente

Completion time: 2025-11-13T10:47:20.0728884-05:00
```

```
-- Paso 2: Crear Database Master Key en master
-- Esta es la clave raíz para cifrado en la base master
IF NOT EXISTS (SELECT * FROM sys.symmetric_keys WHERE name = '##MS_DatabaseMasterKey##')
BEGIN
    CREATE MASTER KEY ENCRYPTION BY PASSWORD = 'MasterKey_QhatuPeru2025!';
    PRINT 'Database Master Key creada en master';
END
ELSE
BEGIN
    PRINT 'Database Master Key ya existe en master';
END
GO

-- Paso 3: Crear certificado de servidor para TDE
```

0 %

Messages

```
Database Master Key creada en master

Completion time: 2025-11-13T10:47:53.7846601-05:00
```

CREACIÓN DE CERTIFICADO PARA TDE

```
-- Paso 3: Crear certificado de servidor para TDE
-- Este certificado protegerá la Database Encryption Key
IF NOT EXISTS (SELECT * FROM sys.certificates WHERE name = 'TDE_Certificate_QhatuPeru')
BEGIN
    CREATE CERTIFICATE TDE_Certificate_QhatuPeru
    WITH SUBJECT = 'Certificado TDE para base de datos QhatuPeru',
         EXPIRY_DATE = '2026-12-31';

    PRINT 'Certificado TDE_Certificate_QhatuPeru creado correctamente';
END
ELSE
BEGIN
    PRINT 'Certificado TDE_Certificate_QhatuPeru ya existe';
END
GO
```

00 %

Messages

```
Certificado TDE_Certificate_QhatuPeru creado correctamente

Completion time: 2025-11-13T10:48:49.7201320-05:00
```

CREACIÓN DE DATA ENCRIPTION KEY

```sql
-- Paso 5: Crear Database Encryption Key en QhatuPeru
USE QhatuPeru;
GO

IF NOT EXISTS (SELECT * FROM sys.dm_database_encryption_keys WHERE database_id = DB_ID('QhatuPeru'))
BEGIN
    CREATE DATABASE ENCRYPTION KEY
    WITH ALGORITHM = AES_256
    ENCRYPTION BY SERVER CERTIFICATE TDE_Certificate_QhatuPeru;

    PRINT 'Database Encryption Key creada con algoritmo AES_256';
END
ELSE
BEGIN
    PRINT 'Database Encryption Key ya existe para QhatuPeru';
END
GO
```

100 %

Messages

```
Database Encryption Key creada con algoritmo AES_256

Completion time: 2025-11-13T10:49:28.1598558-05:00
```

```sql
-- Paso 6: Activar el cifrado TDE
ALTER DATABASE QhatuPeru
SET ENCRYPTION ON;
GO

PRINT 'TDE ACTIVADO para QhatuPeru';
PRINT 'El cifrado está en proceso (puede tomar tiempo según tamaño de BD)';
GO
```

100 %

Messages

```
TDE ACTIVADO para QhatuPeru
El cifrado está en proceso (puede tomar tiempo según tamaño de BD)

Completion time: 2025-11-13T10:49:53.8496600-05:00
```

VERIFICACIÓN

```sql
SELECT
    db.name AS DatabaseName,
    db.is_encrypted AS IsEncrypted,
    dek.encryption_state AS EncryptionState,
    CASE dek.encryption_state
        WHEN 0 THEN 'Sin cifrado'
        WHEN 1 THEN 'Sin cifrar'
        WHEN 2 THEN 'Cifrado en progreso'
        WHEN 3 THEN 'Cifrado completo'
        WHEN 4 THEN 'Cambio de clave en progreso'
        WHEN 5 THEN 'Descifrado en progreso'
        WHEN 6 THEN 'Cambio de protección en progreso'
    END AS EncryptionStateDescription,
    dek.percent_complete AS PercentComplete,
    dek.encryption_state_desc AS StateDescription,
    dek.key_algorithm AS KeyAlgorithm,
    dek.key_length AS KeyLength,
    dek.create_date AS EncryptionStartDate,
    dek.modify_date AS LastModified
FROM sys.databases db
LEFT JOIN sys.dm_database_encryption_keys dek ON db.database_id = dek.database_id
WHERE db.name = 'QhatuPeru';
GO
```

100 %

Results | Messages

	DatabaseName	IsEncrypted	EncryptionState	EncryptionStateDescription	PercentComplete	StateDescription	KeyAlgorithm	KeyLength	EncryptionStartDate	LastModified
1	QhatuPeru	1	3	Cifrado completo	0	ENCRYPTED	AES	256	2025-11-13 15:49:28.150	2025-11-13 15:49:28.150

```sql
-- Certificado TDE
SELECT
    'Certificado TDE' AS CertType,
    name AS CertificateName,
    certificate_id,
    subject,
    expiry_date,
    pvt_key_last_backup_date AS PrivateKeyBackupDate
FROM sys.certificates
WHERE name = 'TDE_Certificate_QhatuPeru';
GO
```

00 %

Results | Messages

	CertType	CertificateName	certificate_id	subject	expiry_date	PrivateKeyBackupDate
1	Certificado TDE	TDE_Certificate_QhatuPeru	258	Certificado TDE para base de datos QhatuPeru	2026-12-31 00:00:00.000	2025-11-13 15:49:20.777

```sql
-- Database Encryption Key
USE QhatuPeru;
GO

SELECT
    'Database Encryption Key' AS KeyType,
    database_id,
    encryption_state_desc,
    key_algorithm,
    key_length,
    create_date,
    modify_date
FROM sys.dm_database_encryption_keys
WHERE database_id = DB_ID('QhatuPeru');
GO

-- Paso 9: Documentar arquitectura de TDE
PRINT '';
```

.00 %

Results | Messages

	KeyType	database_id	encryption_state_desc	key_algorithm	key_length	create_date	modify_date
1	Database Encryption Key	5	ENCRYPTED	AES	256	2025-11-13 15:49:28.150	2025-11-13 15:49:28.150