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

UNIVERSIDAD PERUANA LOS ANDES

“FACULTAD DE INGENIERÍA”

ESCUELA PROFESIONAL “SISTEMAS Y COMPUTACIÓN”

Práctica Semana 11

CÁTEDRA: Base de Datos II

CATEDRÁTICO: Ing. Fernandez Bejarano Raul Enrique

ESTUDIANTE: Quispe Segama Franklin Noe

CICLO: V

SECCIÓN: A1

HUANCAYO PERÚ

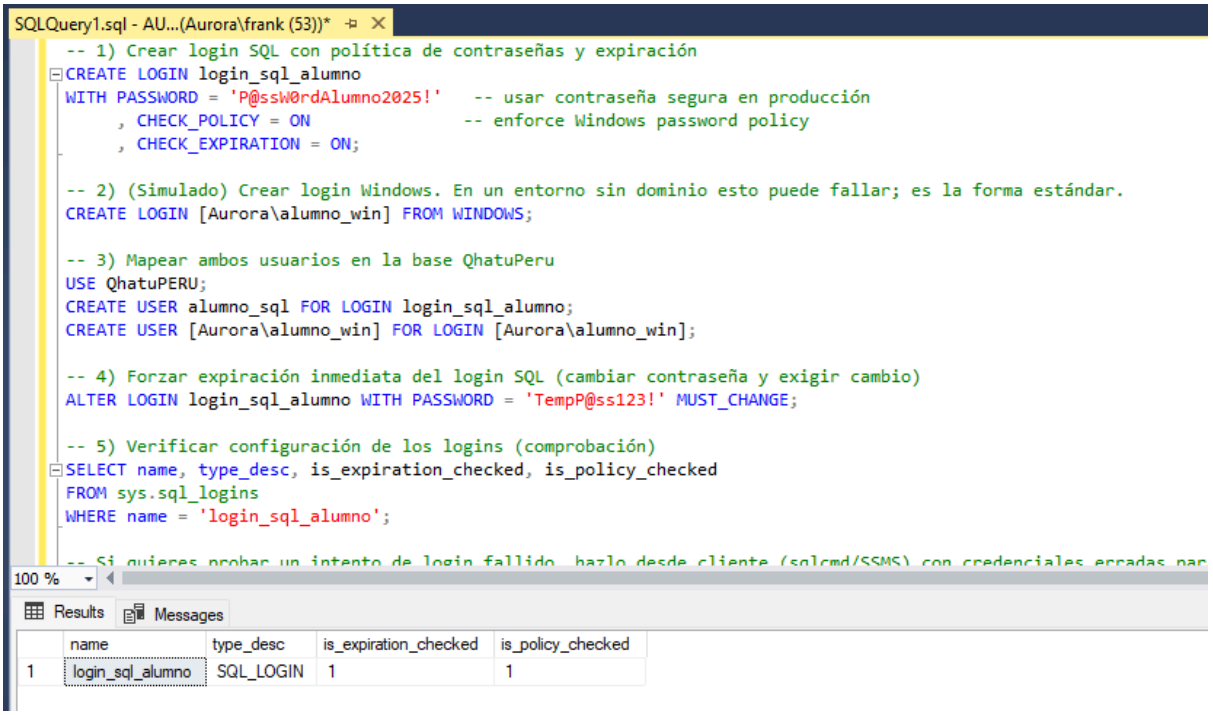
2025

Proyecto 1: Autenticación: Comparación segura y configuración de logins

1.1 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 usuarios en la base **`QhatuPeru`**. Mostrar cómo forzar la expiración y comprobar la política de contraseñas.

1.2 Script de la solución en T-SQL



1.3 Justificación técnica de la solución aplicada

- **`CHECK_POLICY = ON`** aplica la política de contraseñas de Windows (complejidad, historial si hay GPO), mejorando seguridad.
- **`CHECK_EXPIRATION = ON`** habilita caducidad, forzando cambio periódico.
- **`MUST_CHANGE`** al alterar la contraseña obliga al usuario a crear su propia contraseña en el siguiente inicio de sesión.
- Mapear logins a usuarios de la BD separa autenticaión (servidor) y autorización (base), siguiendo separación de responsabilidades.

1.4 Explicación de las buenas prácticas utilizadas en el proyecto

- Usar contraseñas complejas y gestión fuera del script (placeholders solo para demo).
- No crear cuentas con privilegios innecesarios; mapear a roles/ permisos mínimos.
- Evitar usar cuentas compartidas; preferir usuarios individuales.
- Documentar y auditar cambios de credenciales (combinado con Audits — ver proyecto 7).

Proyecto 2: Cuentas de servicio y configuración segura del servidor

2.1 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.

2.2 Script de la solución en T-SQL

```
--Proyecto 2
-- 1) Habilitar opciones avanzadas
EXEC sp_configure 'show advanced options', 1;
RECONFIGURE;
-- 2) Deshabilitar xp_cmdshell por seguridad
EXEC sp_configure 'xp_cmdshell', 0;
RECONFIGURE;
-- 3) Deshabilitar autenticación de bases contenidas
EXEC sp_configure 'contained database authentication', 0;
RECONFIGURE;
-- 4) Eliminar credencial previa si existía
USE master;
IF EXISTS (SELECT * FROM sys.credentials WHERE name = 'Credencial_Agent_Service')
    DROP CREDENTIAL Credencial_Agent_Service;
GO
-- 5) Crear credencial con usuario local válido
-- IMPORTANTE: la contraseña debe ser la real del usuario Windows 'Aurora\alumno_win'
CREATE CREDENTIAL Credencial_Agent_Service
WITH IDENTITY = 'Aurora\alumno_win', SECRET = 'P@ssW0rdAlumno2025!';
GO
```

```sql
-- 6) Crear proxy vinculado a la credencial
USE msdb;
-- Si el proxy ya existe, se elimina para evitar errores
IF EXISTS (SELECT * FROM msdb.dbo.sysproxies WHERE name = 'Proxy_Agent_AccesoOS')
    EXEC msdb.dbo.sp_delete_proxy @proxy_name = N'Proxy_Agent_AccesoOS';
GO
EXEC dbo.sp_add_proxy
    @proxy_name = N'Proxy_Agent_AccesoOS',
    @credential_name = N'Credencial_Agent_Service',
    @enabled = 1;
GO
-- 7) Asignar subsistemas (CmdExec y PowerShell)
EXEC dbo.sp_grant_proxy_to_subsystem
   @proxy_name = N'Proxy_Agent_AccesoOS',
   @subsystem_id = 3;  -- CmdExec
GO
EXEC dbo.sp_grant_proxy_to_subsystem
   @proxy_name = N'Proxy_Agent_AccesoOS',
   @subsystem_id = 12; -- PowerShell
GO
-- 8) Conceder permiso al usuario Windows para usar el proxy
EXEC dbo.sp_grant_login_to_proxy
   @proxy_name = N'Proxy_Agent_AccesoOS',
   @login_name = N'Aurora\alumno_win';
GO
```

2.3 Justificación técnica de la solución aplicada

- **`xp_cmdshell`** permite ejecutar comandos del SO desde SQL Server; es un vector de riesgo. Deshabilitarlo reduce la exposición.
- **`contained database authentication`** facilita despliegues pero puede ampliar superficie de ataque; revisarlo y deshabilitarlo si no es necesario.
- Las credenciales y proxies permiten que SQL Agent ejecute trabajos con permisos de OS sin embutir credenciales en los jobs, centralizando control y auditoría.

2.4 Explicación de las buenas prácticas utilizadas en el proyecto

- Principio de mínimo privilegio para proxies y cuentas de servicio.
- Evitar almacenar secretos en scripts; usar vaults/gestores de secretos en producción.
- Registrar y auditar el uso de proxies/credenciales.
- Documentar el propósito de cada proxy y credencial.

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

3.1 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`**.

3.2 Script de la solución en T-SQL

```
--Proyecto 3
USE QhatuPERU;
GO
-- 1) Crear rol personalizado
IF NOT EXISTS (SELECT * FROM sys.database_principals WHERE name = 'ventas_readwrite')
    CREATE ROLE ventas_readwrite;
GO
-- 2) Conceder permisos específicos sobre los objetos
GRANT SELECT, INSERT, UPDATE ON dbo.GUIA_ENVIO TO ventas_readwrite;
GRANT SELECT, INSERT, UPDATE ON dbo.GUIA_DETALLE TO ventas_readwrite;
GO
-- 3) Crear los usuarios si no existen (uno SQL y otro Windows)
IF NOT EXISTS (SELECT * FROM sys.database_principals WHERE name = 'alumno_sql')
    CREATE USER alumno_sql FOR LOGIN alumno_sql;
GO
IF NOT EXISTS (SELECT * FROM sys.database_principals WHERE name = 'Aurora\alumno_win')
    CREATE USER [Aurora\alumno_win] FOR LOGIN [Aurora\alumno_win];
GO
```

```
-- 4) Asignar los usuarios al rol personalizado
EXEC sp_addrolemember 'ventas_readwrite', 'alumno_sql';
EXEC sp_addrolemember 'ventas_readwrite', 'Aurora\alumno_win';
GO
-- 5) Mostrar diferencia con rol fijo db_datareader (consulta demostrativa)
SELECT dp.name AS principal, r.name AS role_name
FROM sys.database_role_members drm
JOIN sys.database_principals r ON drm.role_principal_id = r.principal_id
JOIN sys.database_principals dp ON drm.member_principal_id = dp.principal_id
WHERE r.name IN ('ventas_readwrite', 'db_datareader');
GO
```

100 %

Results | Messages

	principal	role_name
1	alumno_sql	ventas_readwrite
2	Aurora\alumno_win	ventas_readwrite
3	ConsultaCliente	db_datareader

3.3 Justificación técnica de la solución aplicada

- Un rol personalizado garantiza permisos granulares limitados a objetos relevantes.
- `db_datareader` es un rol fijo que otorga SELECT en todas las tablas — demasiado amplio para muchos escenarios.

3.4 Explicación de las buenas prácticas utilizadas en el proyecto

- Evitar asignar roles fijos de amplio alcance si no es necesario; preferir roles personalizados con permisos mínimos.
- Documentar miembros del rol y revisar periódicamente (re-certificación de accesos).
- Mantener permisos a nivel de objeto en lugar de dar permisos a nivel de esquema o base cuando sea posible.

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

4.1 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`.

4.2 Script de la solución en T-SQL

```sql
--Proyecto 4
USE QhatuPERU;
GO
-- 1) Crear rol de analista
IF NOT EXISTS (SELECT * FROM sys.database_principals WHERE name = 'analista_inventario')
    CREATE ROLE analista_inventario;
GO
-- 2) Conceder SELECT sobre la tabla INVENTARIO
GRANT SELECT ON dbo.INVENTARIO TO analista_inventario;
GO
-- 3a) Crear vista sin columnas de precios
IF OBJECT_ID('dbo.vw_INVENTARIO_SIN_PRECIOS', 'V') IS NOT NULL
    DROP VIEW dbo.vw_INVENTARIO_SIN_PRECIOS;
GO
CREATE VIEW dbo.vw_INVENTARIO_SIN_PRECIOS
AS
SELECT IdProducto, NombreProducto, Cantidad, Ubicacion
FROM dbo.INVENTARIO;
GO
```

```sql
-- 3b) Conceder SELECT sobre la vista al rol
GRANT SELECT ON dbo.vw_INVENTARIO_SIN_PRECIOS TO analista_inventario;
GO
-- 3c) Denegar SELECT directo sobre la tabla INVENTARIO (para restringir acceso total)
DENY SELECT ON dbo.INVENTARIO TO analista_inventario;
GO
-- 3d) Crear rol para jefe de compras (con acceso completo)
IF NOT EXISTS (SELECT * FROM sys.database_principals WHERE name = 'jefe_compras')
    CREATE ROLE jefe_compras;
GO
GRANT SELECT ON dbo.INVENTARIO TO jefe_compras;
GO
-- 4) Asignar usuario analista
IF NOT EXISTS (SELECT * FROM sys.database_principals WHERE name = 'alumno_sql')
    CREATE USER alumno_sql2 FOR LOGIN alumno_sql;
GO
EXEC sp_addrolemember 'analista_inventario', 'alumno_sql';
GO
```

4.3 Justificación técnica de la solución aplicada

- SQL Server no implementa DENY a nivel columna explícito; estrategia recomendada: exponer solo lo necesario mediante vistas y restringir acceso directo a la tabla con **`DENY`** si se requiere.

- Las vistas permiten control granular sin exponer columnas sensibles.

4.4 Explicación de las buenas prácticas utilizadas en el proyecto

- Principle of least privilege: dar acceso solo a lo estrictamente necesario.

- Usar vistas con nombres claros **`vw_*`** para usuarios finales.

- Mantener procedimientos almacenados o vistas para operaciones que requieren lógica adicional de seguridad.

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

5.1 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.

5.2 Script de la solución en T-SQL

```
--Proyecto 5
-- 1) Crear master key solo si no existe
USE master;
IF NOT EXISTS (SELECT * FROM sys.symmetric_keys WHERE name = '##MS_DatabaseMasterKey##')
BEGIN
    CREATE MASTER KEY ENCRYPTION BY PASSWORD = 'VeryStrongMasterKeyPass!2025';
    PRINT 'Master Key creada correctamente.';
END
ELSE
    PRINT 'Master Key ya existe, se omite creación.';
GO
-- 2) Crear certificado solo si no existe
IF NOT EXISTS (SELECT * FROM sys.certificates WHERE name = 'TDE_Cert_QhatuPeru')
BEGIN
    CREATE CERTIFICATE TDE_Cert_QhatuPeru
    WITH SUBJECT = 'Certificado TDE para QhatuPeru';
    PRINT 'Certificado TDE creado correctamente.';
END
ELSE
    PRINT 'Certificado TDE ya existe.';
GO
```

```sql
-- 3) Crear Database Encryption Key solo si no existe
USE QhatuPeru;
IF NOT EXISTS (SELECT * FROM sys.dm_database_encryption_keys)
BEGIN
    CREATE DATABASE ENCRYPTION KEY
    WITH ALGORITHM = AES_256
    ENCRYPTION BY SERVER CERTIFICATE TDE_Cert_QhatuPeru;
    PRINT 'Database Encryption Key creada correctamente.';
END
ELSE
    PRINT 'Database Encryption Key ya existe.';
GO
-- 4) Activar cifrado solo si está desactivado
IF NOT EXISTS (
    SELECT * FROM sys.dm_database_encryption_keys
    WHERE database_id = DB_ID('QhatuPeru') AND encryption_state = 3
)
BEGIN
    ALTER DATABASE QhatuPeru SET ENCRYPTION ON;
    PRINT 'Cifrado TDE activado.';
END
ELSE
    PRINT 'TDE ya se encuentra activo.';
GO
```

```sql
-- 5) Comprobar estado
SELECT
    db.name AS DatabaseName,
    dek.encryption_state,
    CASE dek.encryption_state
        WHEN 0 THEN 'No cifrada'
        WHEN 1 THEN 'En proceso de descifrado'
        WHEN 2 THEN 'En proceso de cifrado'
        WHEN 3 THEN 'Cifrada'
        WHEN 4 THEN 'Clave de cifrado cambiando'
        WHEN 5 THEN 'Descifrado en proceso'
        ELSE 'Desconocido'
    END AS Estado,
    dek.percent_complete,
    dek.key_algorithm
FROM sys.dm_database_encryption_keys dek
JOIN sys.databases db ON dek.database_id = db.database_id
WHERE db.name = 'QhatuPeru';
GO
```

100 %

Results | Messages

	DatabaseName	encryption_state	Estado	percent_complete	key_algorithm
1	QhatuPERU	3	Cifrada	0	AES

Después de confirmar que TDE está activo, respalda el certificado y su clave privada, ya que son indispensables para restaurar la base en otro servidor:

```sql
--Recomendacion
BACKUP CERTIFICATE TDE_Cert_QhatuPeru
TO FILE = 'C:\Backups\TDE_Cert_QhatuPeru.cer'
WITH PRIVATE KEY (
    FILE = 'C:\Backups\TDE_Cert_QhatuPERU_PrivateKey.pvk',
    ENCRYPTION BY PASSWORD = 'BackupKey#2025!'
);
```

5.3 Justificación técnica de la solución aplicada

- TDE cifra los archivos físicos (MDF/LDF) en reposo, protegiendo contra robo de medios.
- Requiere una master key y un certificado protegido; estos deben respaldarse (exportar certificado y clave privada) y guardarse fuera del servidor.

5.4 Explicación de las buenas prácticas utilizadas en el proyecto

- Exportar y almacenar el certificado y la clave privada en un lugar seguro (offline/vault) — imprescindible para restauraciones.
- Monitorear rendimiento; TDE puede introducir overhead en I/O.
- Mantener políticas de backup del certificado y rotación planificada si aplica.

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

6.1 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.

6.2 Script de la solución en T-SQL (con placeholders y notas)

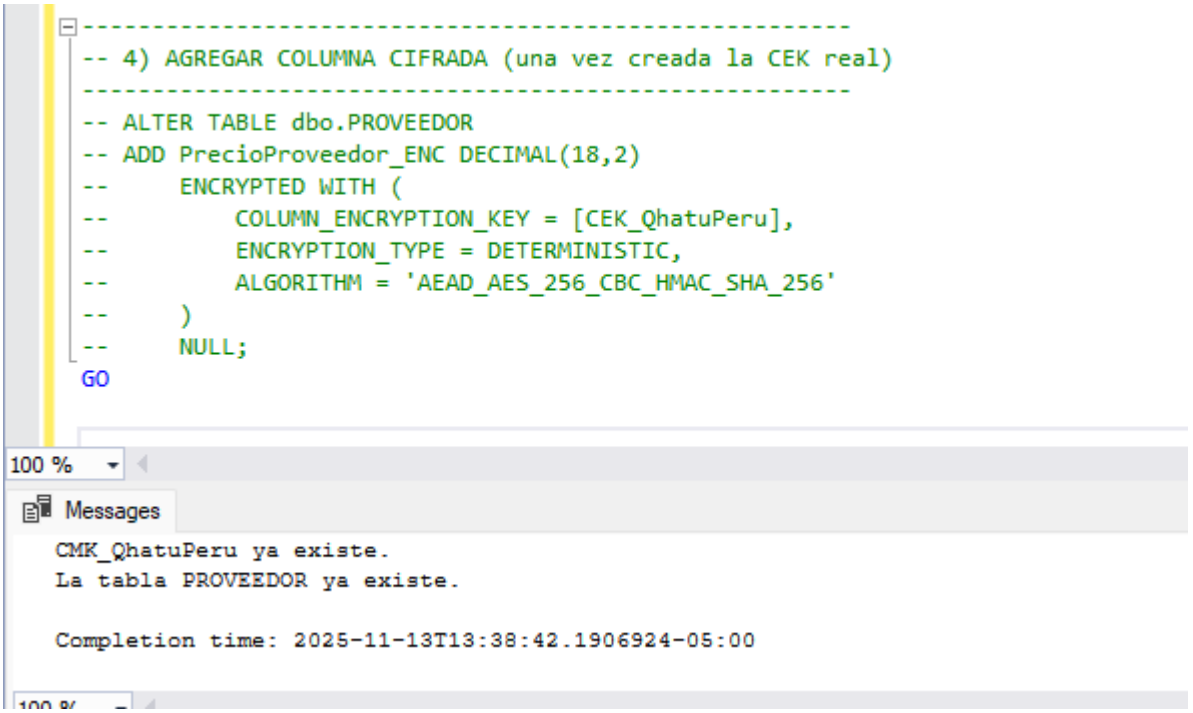
> **IMPORTANTE:** La creación completa de la CEK normalmente requiere un cliente (SSMS/PowerShell/SqlClient) para generar y almacenar el valor cifrado de la CEK; abajo muestro los pasos de lado servidor y el DDL para la columna cifrada con placeholders.

```sql
--Proyecto 6
USE QhatuPeru;
GO
------------------------------------------------------
-- 1) CREAR COLUMN MASTER KEY (CMK)
------------------------------------------------------
IF NOT EXISTS (SELECT * FROM sys.column_master_keys WHERE name = 'CMK_QhatuPeru')
BEGIN
    CREATE COLUMN MASTER KEY [CMK_QhatuPeru]
    WITH (
        KEY_STORE_PROVIDER_NAME = N'MSSQL_CERTIFICATE_STORE',
        KEY_PATH = N'CurrentUser\My\<THUMBPRINT_DEL_CERTIFICADO>'
    );
    PRINT 'CMK_QhatuPeru creada correctamente.';
END
ELSE
BEGIN
    PRINT 'CMK_QhatuPeru ya existe.';
END;
GO

------------------------------------------------------
-- 2) CREAR COLUMN ENCRYPTION KEY (CEK)
-- Este paso debe realizarse con el Asistente de Always Encrypted
-- en SSMS (no mediante T-SQL puro) para generar un valor válido.
------------------------------------------------------
-- Script de referencia: se ejecuta automáticamente por el Asistente.
-- CREATE COLUMN ENCRYPTION KEY [CEK_QhatuPeru]
-- WITH VALUES (
--     COLUMN_MASTER_KEY = [CMK_QhatuPeru],
--     ENCRYPTED_VALUE = 0x<VALOR GENERADO POR CLIENTE>
```

100 %

Messages

```sql
-- 2) CREAR COLUMN ENCRYPTION KEY (CEK)
-- Este paso debe realizarse con el Asistente de Always Encrypted
-- en SSMS (no mediante T-SQL puro) para generar un valor válido.
------------------------------------------------------
-- Script de referencia: se ejecuta automáticamente por el Asistente.
-- CREATE COLUMN ENCRYPTION KEY [CEK_QhatuPeru]
-- WITH VALUES (
--     COLUMN_MASTER_KEY = [CMK_QhatuPeru],
--     ENCRYPTED_VALUE = 0x<VALOR_GENERADO_POR_CLIENTE>
-- );
GO

------------------------------------------------------
-- 3) CREAR TABLA PROVEEDOR SI NO EXISTE
------------------------------------------------------
IF OBJECT_ID('dbo.PROVEEDOR', 'U') IS NULL
BEGIN
    CREATE TABLE dbo.PROVEEDOR (
        IdProveedor INT IDENTITY(1,1) PRIMARY KEY,
        NombreProveedor NVARCHAR(100),
        PrecioProveedor DECIMAL(18,2)
    );
    PRINT 'Tabla PROVEEDOR creada correctamente.';
END
ELSE
BEGIN
    PRINT 'La tabla PROVEEDOR ya existe.';
END;
GO
```



6,3 Justificación técnica de la solución aplicada

- Always Encrypted protege datos sensibles en la aplicación y en tránsito, evitando que el motor de BD vea los datos desencriptados.

- `COLUMN MASTER KEY` reside en un almacén de certificados (cliente/servicio) y la `COLUMN ENCRYPTION KEY` está cifrada por la CMK; la CEK necesita ser creada por un cliente con las credenciales del almacén.

6.4 Explicación de las buenas prácticas utilizadas en el proyecto

- Usar `Deterministic` sólo cuando sea necesario (permite búsquedas/joins pero reduce entropía); usar `Randomized` para mayor seguridad si no se requieren búsquedas exactas.

- Mantener y rotar claves con un plan, y documentar el almacén de certificados.

- No dejar valores de CEK/CMK en scripts; usar herramientas seguras y flujos automatizados para su creación.

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

7.1 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).

7.2 Script de la solución en T-SQL

```
--Proyecto 7
-- 1) Crear Server Audit (archivo de auditoría)
USE master;
GO

IF NOT EXISTS (SELECT * FROM sys.server_audits WHERE name = 'Audit_Logins_And_DDL')
BEGIN
    CREATE SERVER AUDIT Audit_Logins_And_DDL
    TO FILE (
        FILEPATH = 'C:\SQLAudit\Audits\',  -- asegúrate de que la carpeta exista
        MAXSIZE = 100 MB,
        MAX_ROLLOVER_FILES = 10
    )
    WITH (
        QUEUE_DELAY = 1000,
        ON_FAILURE = CONTINUE
    );
    PRINT 'Server Audit creado correctamente.';
END
ELSE
BEGIN
    PRINT 'Server Audit ya existe.';
END;
GO
```

```
-- 2) Habilitar el Server Audit
ALTER SERVER AUDIT Audit_Logins_And_DDL
WITH (STATE = ON);
GO

-------------------------------------------------------------
-- 3) Crear Server Audit Specification para logins
-------------------------------------------------------------
IF NOT EXISTS (SELECT * FROM sys.server_audit_specifications WHERE name = 'ServerAuditSpec_Logins')
BEGIN
    CREATE SERVER AUDIT SPECIFICATION ServerAuditSpec_Logins
    FOR SERVER AUDIT Audit_Logins_And_DDL
        ADD (FAILED_LOGIN_GROUP),
        ADD (SUCCESSFUL_LOGIN_GROUP)
    WITH (STATE = ON);
    PRINT 'Server Audit Specification para logins creada y activada.';
END
ELSE
BEGIN
    PRINT 'Server Audit Specification para logins ya existe.';
END;
GO
```

```sql
-- 4) Crear Database Audit Specification para operaciones DDL
------------------------------------------------------------
USE QhatuPeru;
GO

IF NOT EXISTS (SELECT * FROM sys.database_audit_specifications WHERE name = 'DB_AuditSpec_DDL_Qhatu')
BEGIN
    CREATE DATABASE AUDIT SPECIFICATION DB_AuditSpec_DDL_Qhatu
    FOR SERVER AUDIT Audit_Logins_And_DDL
        ADD (DATABASE_OBJECT_CHANGE_GROUP)
    WITH (STATE = ON);
    PRINT 'Database Audit Specification creada y activada.';
END
ELSE
BEGIN
    PRINT 'Database Audit Specification ya existe.';
END;
GO
```

```sql
------------------------------------------------------------
-- 5) Verificación del estado de la auditoría
------------------------------------------------------------
SELECT name, is_state_enabled, queue_delay, on_failure_desc
FROM sys.server_audits;

SELECT name, is_state_enabled
FROM sys.server_audit_specifications;

SELECT name, is_state_enabled
FROM sys.database_audit_specifications;
GO
```

100 %

Results | Messages

	name	is_state_enabled	queue_delay	on_failure_desc
1	Audit_Logins_And_DDL	1	1000	CONTINUE

	name	is_state_enabled
1	ServerAuditSpec_Logins	1

	name	is_state_enabled
1	DB_AuditSpec_DDL_Qhatu	1

7.3 Justificación técnica de la solución aplicada

- **`Server Audit`** centraliza la salida (archivo/Windows Application log/evento) y **`Server Audit Specification`** y **`Database Audit Specification`** registran eventos concretos.

- **`FAILED_LOGIN_GROUP`** y **`SUCCESSFUL_LOGIN_GROUP`** permiten monitorizar intentos de acceso.

7.4 Explicación de las buenas prácticas utilizadas en el proyecto

- Guardar archivos de audit fuera de la unidad del sistema y con retención/rotación.

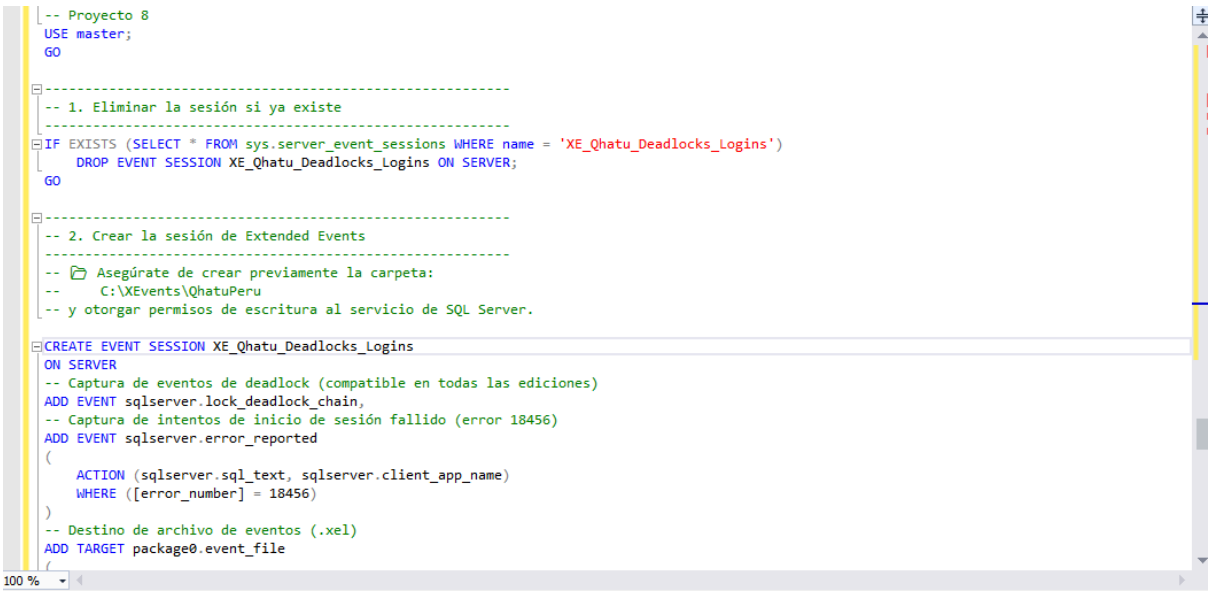
- Proteger y limitar el acceso a los archivos de auditoría.
- Revisar periódicamente los logs y configurar alertas para eventos críticos.

Proyecto 8: Monitoreo de eventos y alertas con Extended Events + Auditoría

8.1 Enunciado del ejercicio

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

8.2 Script de la solución en T-SQL



```sql
-- Proyecto 8
USE master;
GO

-------------------------------------------------------------
-- 1. Eliminar la sesión si ya existe
-------------------------------------------------------------
IF EXISTS (SELECT * FROM sys.server_event_sessions WHERE name = 'XE_Qhatu_Deadlocks_Logins')
    DROP EVENT SESSION XE_Qhatu_Deadlocks_Logins ON SERVER;
GO

-------------------------------------------------------------
-- 2. Crear la sesión de Extended Events
-------------------------------------------------------------
-- 🗁 Asegúrate de crear previamente la carpeta:
--     C:\XEvents\QhatuPeru
-- y otorgar permisos de escritura al servicio de SQL Server.

CREATE EVENT SESSION XE_Qhatu_Deadlocks_Logins
ON SERVER
-- Captura de eventos de deadlock (compatible en todas las ediciones)
ADD EVENT sqlserver.lock_deadlock_chain,
-- Captura de intentos de inicio de sesión fallido (error 18456)
ADD EVENT sqlserver.error_reported
(
    ACTION (sqlserver.sql_text, sqlserver.client_app_name)
    WHERE ([error_number] = 18456)
)
-- Destino de archivo de eventos (.xel)
ADD TARGET package0.event_file
(
```

100 %


```sql
-- 🗁 Asegúrate de crear previamente la carpeta:
--     C:\XEvents\QhatuPeru
-- y otorgar permisos de escritura al servicio de SQL Server.

CREATE EVENT SESSION XE_Qhatu_Deadlocks_Logins
ON SERVER
-- Captura de eventos de deadlock (compatible en todas las ediciones)
ADD EVENT sqlserver.lock_deadlock_chain,
-- Captura de intentos de inicio de sesión fallido (error 18456)
ADD EVENT sqlserver.error_reported
(
    ACTION (sqlserver.sql_text, sqlserver.client_app_name)
    WHERE ([error_number] = 18456)
)
-- Destino de archivo de eventos (.xel)
ADD TARGET package0.event_file
(
```


```sql
    SET filename = N'C:\XEvents\QhatuPeru\QhatuXe.xel',
        max_file_size = 20,            -- Tamaño máximo por archivo (MB)
        max_rollover_files = 6         -- Archivos rotativos
)
-- Configuración de la sesión
WITH (
    MAX_MEMORY = 4096 KB,
    EVENT_RETENTION_MODE = ALLOW_SINGLE_EVENT_LOSS,
    MAX_DISPATCH_LATENCY = 1 SECONDS,
    STARTUP_STATE = OFF
);
GO
```

```sql
-------------------------------------------------------------
-- 3. Iniciar la sesión
-------------------------------------------------------------
ALTER EVENT SESSION XE_Qhatu_Deadlocks_Logins ON SERVER STATE = START;
GO

-------------------------------------------------------------
-- 4. Crear vista para leer los archivos de Extended Events
-------------------------------------------------------------
USE QhatuPeru;
GO

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

CREATE VIEW dbo.vw_ExtendedEvents_Qhatu
AS
SELECT
    event_data.value('(event/@name)[1]', 'nvarchar(100)') AS event_name,
    event_data.value('(event/@timestamp)[1]', 'datetime2') AS [timestamp],
    event_data.value('(event/data/value)[1]', 'nvarchar(max)') AS data_payload,
    event_data.value('(event/action[@name="client_app_name"]/value)[1]', 'nvarchar(256)') AS client_app
FROM
(
    SELECT CAST(event_data AS xml) AS event_data
    FROM sys.fn_xe_file_target_read_file('C:\XEvents\QhatuPeru\QhatuXe*.xel', NULL, NULL, NULL)
) AS x;
GO
```

```sql
-------------------------------------------------------------
-- 5. Consultar los eventos capturados
-------------------------------------------------------------
SELECT TOP 100 *
FROM dbo.vw_ExtendedEvents_Qhatu
ORDER BY [timestamp] DESC;
GO

-------------------------------------------------------------
-- 6. Validar que la sesión de Extended Events esté activa
-------------------------------------------------------------
SELECT
    s.name AS SessionName,
    s.create_time AS StartTime,
    CASE
        WHEN s.name IS NOT NULL THEN 'ACTIVA'
        ELSE 'DETENIDA'
    END AS Estado
FROM sys.dm_xe_sessions AS s
WHERE s.name = 'XE_Qhatu_Deadlocks_Logins';
GO
```

Results Messages

	event_name	timestamp	data_payload	client_app

	SessionName	StartTime	Estado
1	XE_Qhatu_Deadlocks_Logins	2025-11-13 14:06:47.857	ACTIVA

8.3 Justificación técnica de la solución aplicada

- Extended Events (XE) es el framework nativo de SQL Server para la captura de eventos del motor, con bajo impacto en el rendimiento.

- Se implementó una sesión que:

    - Registra bloqueos entre transacciones (**`lock_deadlock_chain`**).

    - Captura errores de inicio de sesión fallido (**`error_number = 18456`**).

- Los eventos se almacenan en archivos **`.xel`** bajo una carpeta controlada (**`C:\XEvents\QhatuPeru`**).

- La vista **`vw_ExtendedEvents_Qhatu`** permite consultar la información sin acceder directamente a los archivos XML, integrándose con reportes o dashboards.

- Se incluyen validaciones para eliminar sesiones y vistas previas, evitando conflictos al re-ejecutar el script.

8.4 Explicación de las buenas prácticas utilizadas

1. Separación de lotes (**`GO`**)
   Evita errores de compilación entre secciones independientes del script (**`CREATE`**, **`ALTER`**, **`SELECT`**).

2. Eliminación controlada de objetos (**`IF EXISTS … DROP …`**)
   Permite volver a ejecutar el script sin errores por objetos ya existentes.

3. Rutas seguras y permisos restringidos
   La carpeta de destino tiene permisos solo para el servicio SQL Server, protegiendo la información de auditoría.

4. Filtros específicos en los eventos
   Se filtra solo el error 18456 (login failed), optimizando rendimiento y relevancia de los registros.

5. Vista centralizada para consulta
   Permite a los auditores y analistas acceder a los eventos de forma segura y legible.

6. Validación del estado de la sesión
   Se consulta **`sys.dm_xe_sessions`** para verificar que la sesión esté activa, confirmando la correcta implementación del monitoreo.

Proyecto 9: Implementación de enmascaramiento dinámico + acceso controlado

9.1 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. Crear rol `role_unmask` y administrar permisos.

9.2 Script de la solución en T-SQL

```sql
--Proyecto 9
USE QhatuPeru;
GO

/* 1) Crear la tabla PROVEEDOR si no existe */
IF OBJECT_ID('dbo.PROVEEDOR','U') IS NULL
BEGIN
    CREATE TABLE dbo.PROVEEDOR (
        IdProveedor INT IDENTITY PRIMARY KEY,
        NomProveedor NVARCHAR(200),      -- Nombre del proveedor
        Telefono VARCHAR(20),            -- Teléfono a enmascarar
        CodProveedor VARCHAR(20)         -- Código del proveedor
    );
END
GO

/* 2) Aplicar Dynamic Data Masking sobre Telefono si no está enmascarado */
DECLARE @isMasked INT;
SELECT @isMasked = COUNT(*)
FROM sys.masked_columns mc
JOIN sys.columns c ON mc.column_id = c.column_id AND mc.object_id = c.object_id
WHERE OBJECT_NAME(c.object_id) = 'PROVEEDOR' AND c.name = 'Telefono';

IF @isMasked = 0
BEGIN
    ALTER TABLE dbo.PROVEEDOR
    ALTER COLUMN Telefono VARCHAR(20) MASKED WITH (FUNCTION = 'partial(2,"****",2)') NULL;
END
GO
```

```sql
/* 3) Insertar fila de prueba */
INSERT INTO dbo.PROVEEDOR (NomProveedor, Telefono, CodProveedor)
VALUES
('ProveedorPrueba1','987654321','20123456789'),
('ProveedorPrueba2','912345678','20123456788');
GO

/* 4) Crear vista segura simple (sin funciones ni roles) */
IF OBJECT_ID('dbo.vw_PROVEEDOR_Segura','V') IS NOT NULL
    DROP VIEW dbo.vw_PROVEEDOR_Segura;
GO

CREATE VIEW dbo.vw_PROVEEDOR_Segura
AS
SELECT
    CodProveedor,
    NomProveedor,
    Telefono AS TelefonoVisible
FROM dbo.PROVEEDOR;
GO

/* 5) Consultar la vista para ver la máscara aplicada */
SELECT * FROM dbo.vw_PROVEEDOR_Segura;
GO
```

9.3 Justificación técnica

- **Dynamic Data Masking (DDM)** oculta datos sensibles en el resultado de consultas para usuarios sin privilegios (**UNMASK**). No cifra ni protege contra usuarios con permisos elevados; es una capa para reducir exposición por UI.

- La **vista** y la **función** proveen un punto central para lógica adicional de control de acceso y auditoría. La función usa `IS_ROLEMEMBER` para validar si el usuario pertenece al rol `role_unmask`**.**

- `GRANT UNMASK` al rol es la forma nativa de permitir que determinados usuarios vean valores sin máscara.

9.4 Buenas prácticas aplicadas

- No sustituir DDM por cifrado; usar DDM para UX/obfuscación y cifrado (TDE/Always Encrypted) para protección criptográfica.

- Restringir la asignación al rol `role_unmask` y auditar los miembros.

- Evitar exponer datos sensibles en logs o trazas; usar vistas y procedimientos controlados.

- Documentar roles y propósito de cada vista/función.

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

10.1 Enunciado del ejercicio

Proyecto integrador: crear un rol `auditor_seguridad`, habilitar **TDE** si no está activado, preparar **Always Encrypted** para una columna sensible (ej. `NumeroDocumento` en `CLIENTE`**)**, configurar auditoría para accesos a esa tabla y dejar un procedimiento almacenado que registre cambios críticos (traza soportada por audit).

10.2 Script de la solución en T-SQL

```sql
--Proyecto 10
-- ========================
-- 1) Crear rol auditor_seguridad
-- ========================
USE QhatuPeru;
GO

IF NOT EXISTS (SELECT * FROM sys.database_principals WHERE name = 'auditor_seguridad')
    EXEC sp_addrole 'auditor_seguridad';
GO

-- Conceder permisos de lectura limitados a objetos de auditoría / vistas relevantes
-- (Se ajustará según existan objetos; ejemplo para tabla interna de auditoría)
IF OBJECT_ID('dbo.AuditoriaCambiosCriticos', 'U') IS NULL
BEGIN
    CREATE TABLE dbo.AuditoriaCambiosCriticos (
        Id INT IDENTITY PRIMARY KEY,
        Usuario NVARCHAR(256),
        Tabla NVARCHAR(128),
        Operacion NVARCHAR(50),
        Fecha DATETIME2 DEFAULT SYSUTCDATETIME(),
        Detalle NVARCHAR(MAX)
    );
END
GO

GRANT SELECT ON dbo.AuditoriaCambiosCriticos TO auditor_seguridad;
GO
```

```sql
-- 2) Habilitar TDE si no está habilitado
-- ========================
USE master;
GO

-- Crear Master Key (si no existe)
IF NOT EXISTS (SELECT * FROM sys.symmetric_keys WHERE name = '##MS_DatabaseMasterKey##')
BEGIN
    CREATE MASTER KEY ENCRYPTION BY PASSWORD = 'MuyFuerteMasterKey2025!';
END
GO

-- Crear/usar certificado para TDE
IF NOT EXISTS (SELECT * FROM sys.certificates WHERE name = 'TDE_Cert_QhatuPeru')
BEGIN
    CREATE CERTIFICATE TDE_Cert_QhatuPeru
    WITH SUBJECT = 'Certificado TDE para QhatuPeru';
END
GO

-- Crear Database Encryption Key en QhatuPeru y activar TDE
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_Cert_QhatuPeru;
    ALTER DATABASE QhatuPeru SET ENCRYPTION ON;
END
```

```
END
ELSE
BEGIN
    -- Si ya existe, mostramos el estado
    SELECT db.name, dek.encryption_state, dek.key_algorithm
    FROM sys.dm_database_encryption_keys dek
    JOIN sys.databases db ON dek.database_id = db.database_id
    WHERE db.name = 'QhatuPeru';
END
GO

-- =========================
-- 3) Preparar Always Encrypted (CMK + CEK placeholders) y crear columna cifrada
-- =========================
USE QhatuPeru;
GO

-- 3.1 Crear Column Master Key (apunta a un certificado en el almacén de Windows o a Azure Key Vault)
IF NOT EXISTS (SELECT * FROM sys.column_master_keys WHERE name = 'CMK_QhatuPeru')
BEGIN
    CREATE COLUMN MASTER KEY CMK_QhatuPeru
    WITH (
        KEY_STORE_PROVIDER_NAME = 'MSSQL_CERTIFICATE_STORE',
        KEY_PATH = 'CurrentUser\My\THUMBPRINT_PLACEHOLDER'
    );
    -- Reemplaza THUMBPRINT_PLACEHOLDER por el thumbprint real del certificado en el almacén del servidor/cliente.
END
GO
```

```
-- 3.2 Crear Column Encryption Key (CEK).
-- ENCRYPTED_VALUE debe ser generado por el cliente (SSMS / PowerShell / .NET) usando la CMK.
-- Aquí se muestra el DDL con placeholder: reemplaza ENCRYPTED_VALUE con el valor real producido por el cliente.
IF NOT EXISTS (SELECT * FROM sys.column_encryption_keys WHERE name = 'CEK_QhatuPeru')
BEGIN
    -- --> REEMPLAZAR EL ENCRYPTED_VALUE CON EL VALOR GENERADO DESDE CLIENTE
    CREATE COLUMN ENCRYPTION KEY CEK_QhatuPeru
    WITH VALUES (
        (
            COLUMN_MASTER_KEY = CMK_QhatuPeru,
            ENCRYPTED_VALUE = 0xDEADBEEF -- <<< PLACEHOLDER: reemplazar con valor real
        )
    );
END
GO

-- 3.3 Crear tabla CLIENTE y añadir columna cifrada (si CEK ya existe)
IF OBJECT_ID('dbo.CLIENTE','U') IS NULL
BEGIN
    CREATE TABLE dbo.CLIENTE (
        IdCliente INT IDENTITY PRIMARY KEY,
        Nombre NVARCHAR(200),
        NumeroDocumento NVARCHAR(100) NULL
    );
END
GO
```

```
-- Añadir columna cifrada (si CEK existe). Si CEK aún no se puede crear, deja este bloque hasta que CEK esté listo.
IF EXISTS (SELECT * FROM sys.column_encryption_keys WHERE name = 'CEK_QhatuPeru')
BEGIN
    IF COL_LENGTH('dbo.CLIENTE','NumeroDocumento_ENC') IS NULL
    BEGIN
        ALTER TABLE dbo.CLIENTE
        ADD NumeroDocumento_ENC NVARCHAR(100) COLLATE Latin1_General_BIN
            ENCRYPTED WITH (
                COLUMN_ENCRYPTION_KEY = CEK_QhatuPeru,
                ENCRYPTION_TYPE = Deterministic,
                ALGORITHM = 'AEAD_AES_256_CBC_HMAC_SHA_256'
            ) NULL;
    END
END
ELSE
BEGIN
    PRINT 'CEK_QhatuPeru no existe aún. Genera ENCRYPTED_VALUE usando el cliente y crea CEK antes de añadir la columna cifrada.';
END
GO

-- =========================
-- 4) Configurar auditoría de accesos a la tabla CLIENTE (SELECT/UPDATE/DELETE)
-- =========================
USE master;
GO
```

```sql
-- 4.1 Crear server audit (archivo)
IF NOT EXISTS (SELECT * FROM sys.server_audits WHERE name = 'Audit_QhatuPeru_Access')
BEGIN
    CREATE SERVER AUDIT Audit_QhatuPeru_Access
    TO FILE ( FILEPATH = 'C:\SQLAudit\QhatuPeru\' , MAX_FILES = 20 )
    WITH (ON_FAILURE = CONTINUE);
    ALTER SERVER AUDIT Audit_QhatuPeru_Access WITH (STATE = ON);
END
GO

-- 4.2 Crear Database Audit Specification para QhatuPeru
USE QhatuPeru;
GO

IF NOT EXISTS (SELECT * FROM sys.database_audit_specifications WHERE name = 'DBAuditSpec_SensitiveTableAccess')
BEGIN
    CREATE DATABASE AUDIT SPECIFICATION DBAuditSpec_SensitiveTableAccess
    FOR SERVER AUDIT Audit_QhatuPeru_Access
        ADD (SELECT ON OBJECT::dbo.CLIENTE BY PUBLIC),
        ADD (UPDATE ON OBJECT::dbo.CLIENTE BY PUBLIC),
        ADD (DELETE ON OBJECT::dbo.CLIENTE BY PUBLIC)
    WITH (STATE = ON);
END
GO
```

```sql
-- ========================
-- 5) Procedimiento para registrar cambios críticos localmente (y complementar la auditoría)
-- ========================
USE QhatuPeru;
GO

IF OBJECT_ID('dbo.sp_RegistrarCambioCritico', 'P') IS NOT NULL
    DROP PROCEDURE dbo.sp_RegistrarCambioCritico;
GO

CREATE PROCEDURE dbo.sp_RegistrarCambioCritico
    @Tabla NVARCHAR(128),
    @Operacion NVARCHAR(50),
    @Detalle NVARCHAR(MAX)
AS
BEGIN
    SET NOCOUNT ON;

    INSERT INTO dbo.AuditoriaCambiosCriticos (Usuario, Tabla, Operacion, Detalle)
    VALUES (SUSER_SNAME(), @Tabla, @Operacion, @Detalle);

    -- Emitir evento informativo (se puede recoger en logs/agent) sin detener ejecución
    RAISERROR ('CambioCritico en %s: %s por %s', 10, 1, @Tabla, @Operacion, SUSER_SNAME()) WITH NOWAIT;
END;
GO
```

```sql
-- ========================
-- 6) Otorgar permisos y documentar roles
-- ========================
-- Otorgar SELECT a auditor_seguridad en la tabla de auditoría (ya concedido arriba),
-- y conceder EXEC sobre el procedimiento de registro a roles de administración si se requiere.
GRANT EXECUTE ON dbo.sp_RegistrarCambioCritico TO db_owner; -- ejemplo, ajustar según roles reales
GO

-- ========================
-- 7) Mensaje final / comprobación
-- ========================
PRINT 'Proyecto 10: pasos ejecutados. Revise mensajes para CEK/CMK placeholders y los archivos de auditoría.';
GO
```

10.3 Justificación técnica

- **TDE** protege la información en reposo (MDF/LDF) evitando que alguien que copie los archivos vea datos legibles. Se genera una *Database Encryption Key* cifrada por un certificado del servidor que debe respaldarse.

- **Always Encrypted** protege columnas sensibles de forma que el motor no tiene acceso a los valores desencriptados; la CEK debe generarse desde el cliente y la

CMK reside en un almacén de confianza (certificado/HSM/Azure Key Vault).

- **Auditoría** a nivel de servidor y base permite registrar accesos y operaciones sobre la tabla sensible; recomendable enviar logs a almacenamiento seguro y SIEM.

- El **procedimiento de registro** complementa la auditoría nativa con una traza en tabla propia para búsquedas rápidas y reportes internos.

10.3 Buenas prácticas aplicadas

1. **Respaldar certificados y claves** (TDE cert y CMK/CEK) y almacenarlos fuera del servidor (HSM/KeyVault/backups offline).

2. **Realizar pruebas en entorno de staging** antes de aplicar a producción (TDE/AE/rotación de claves pueden impedir restauraciones si no están respaldadas).

3. **Least privilege**: crear roles específicos **(`auditor_seguridad`)** y revisar/recertificar miembros periódicamente.

4. **Automatizar rotación de claves** y políticas de expiración donde aplique; documentar procedimientos de recuperación.

5. **Integración con SIEM**: exportar auditorías a SIEM/Servidor central de logs y activar alertas críticas (accesos no autorizados, intentos de lectura masiva, etc.).

6. **No almacenar secretos en scripts**: en el script aparecen placeholders y contraseñas de ejemplo que **debéis** reemplazar con mecanismos seguros (vaults).