Universidad Peruana Los Andes

Facultad de Ingeniería

Escuela profesional de Ingeniería de Sistemas

Curso: Base de datos II

Docente: Raul Enrique Fernandez Bejarano

Estudiante: Sarmiento Mosquera Yeims Abraham

Ciclo: V - Código: s03807f

Huancayo - 2025

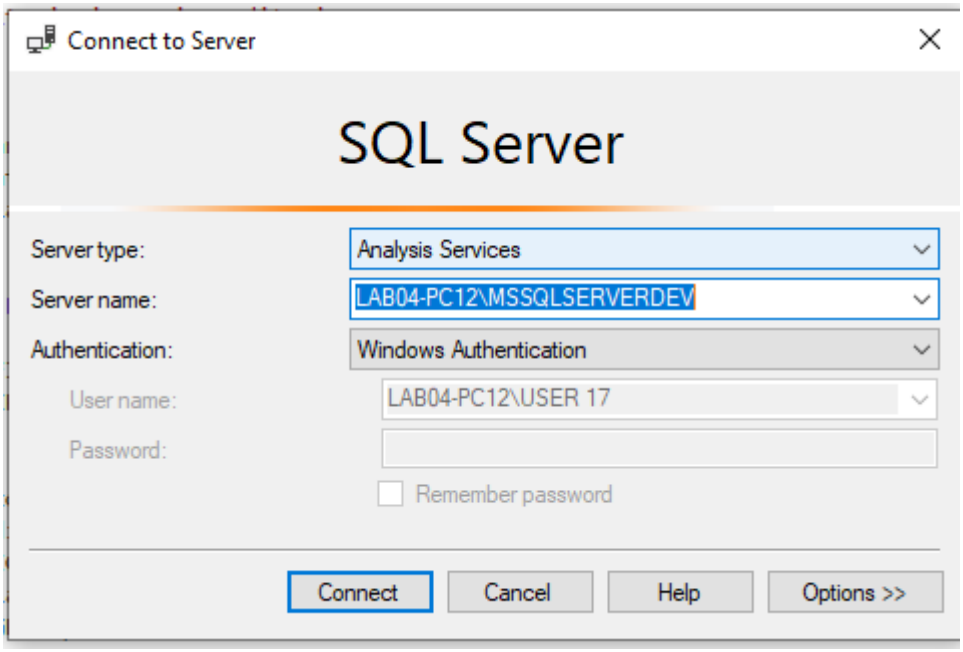
1. **<u>Autenticación SQL y Windows: diferencias y prácticas seguras</u>**
   Diferencias:

Autenticación de Windows	Autenticación de SQL
• Usa las credenciales del sistema operativo.	• Requiere usuario y contraseña definidos en SQL Server.
• Más segura porque integra políticas de dominio, contraseñas y auditoría.	• Útil cuando no se puede usar Active Directory (por ejemplo, en entornos aislados).
• Ideal para entornos corporativos con Active Directory.	• Debe configurarse con contraseñas fuertes y políticas de expiración.

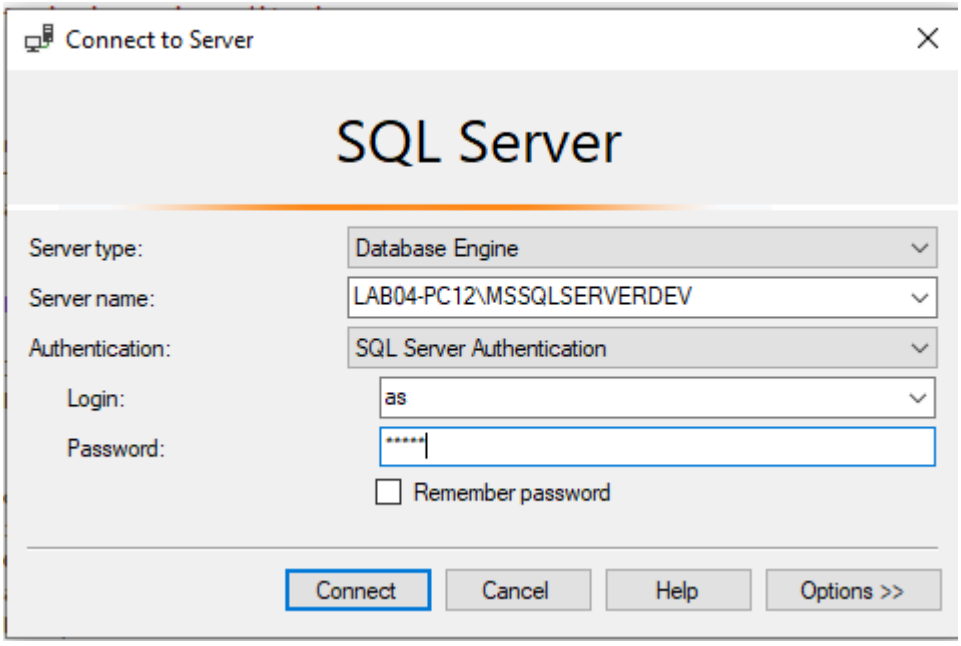
Prácticas seguras

- Preferir autenticación Windows cuando sea posible.
- Deshabilitar la cuenta `sa` o cambiarle el nombre.
- Usar conexiones cifradas (SSL/TLS) y limitar accesos por IP.

Resultado



La autenticación de windows no te pide credenciales



La autenticación de SQL si te pide credenciales para conectarte a la base de datos

2. **<u>Cuentas de servicio y configuración del servidor</u>**

Cuentas de servicio:

Son las cuentas bajo las cuales se ejecutan los servicios de SQL Server (Motor, Agente, etc.). Deben tener los permisos mínimos necesarios (principio de menor privilegio).

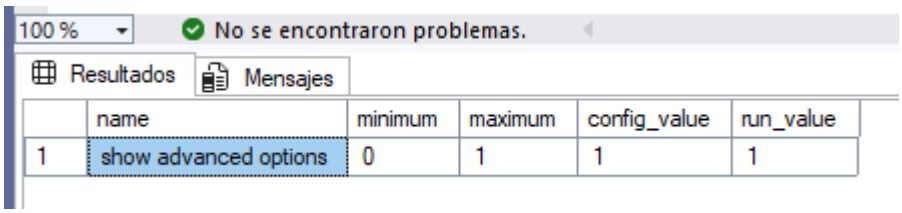
Configuración clave del servidor:

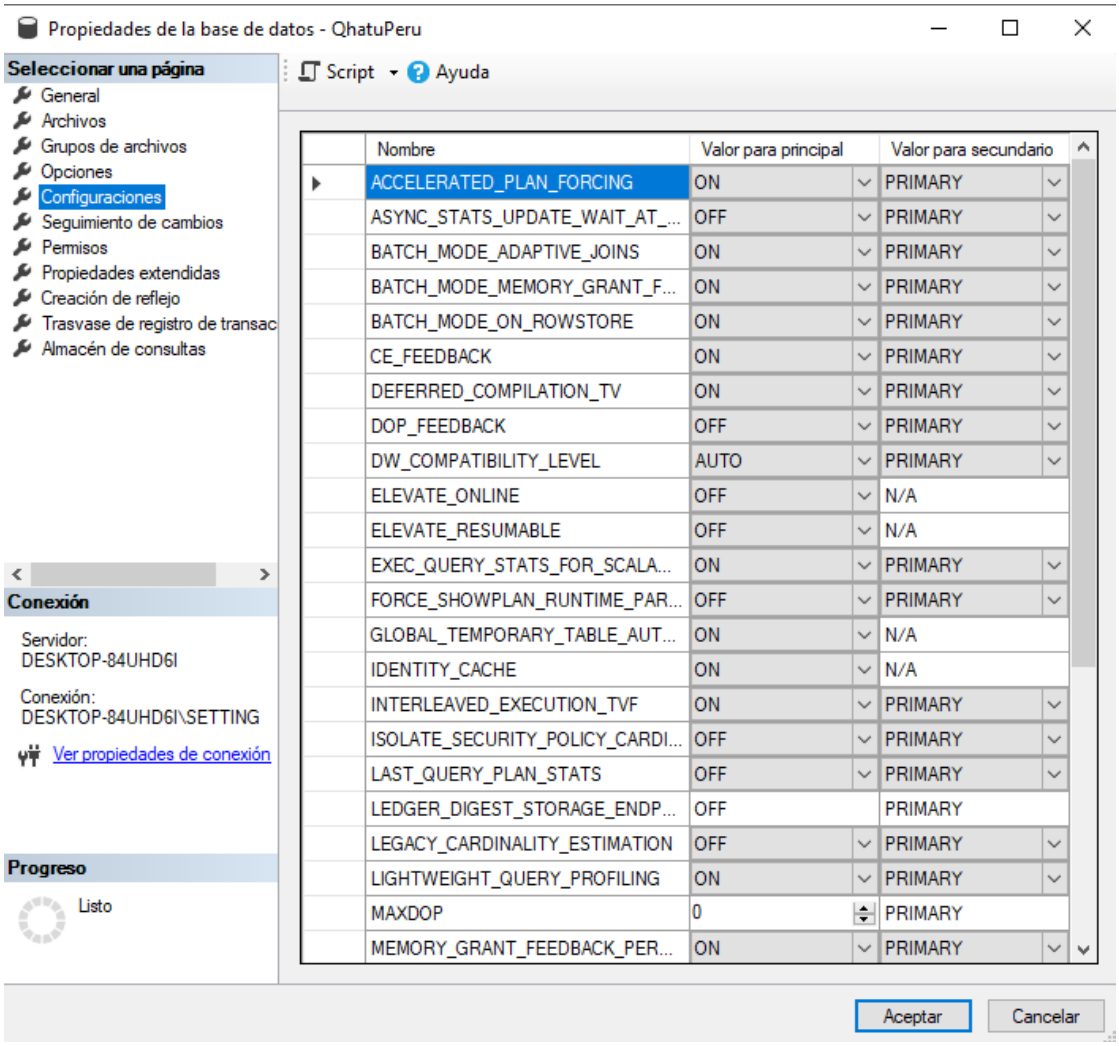
- Revisar opciones como `xp_cmdshell`, `Ad Hoc Distributed Queries`, y `remote access`.
- Usar `sp_configure` para ajustar parámetros de seguridad.

Ejemplo

```
EXEC sp_configure 'show advanced options', 1;
RECONFIGURE;
EXEC sp_configure 'xp_cmdshell', 0;
RECONFIGURE;
```

Resultado





3. Creación de roles fijos y personalizados

Roles fijos:

- SQL Server incluye roles predefinidos como `db_datareader`, `db_datawriter`, `db_owner`, etc.
- Se asignan según el nivel de acceso requerido.

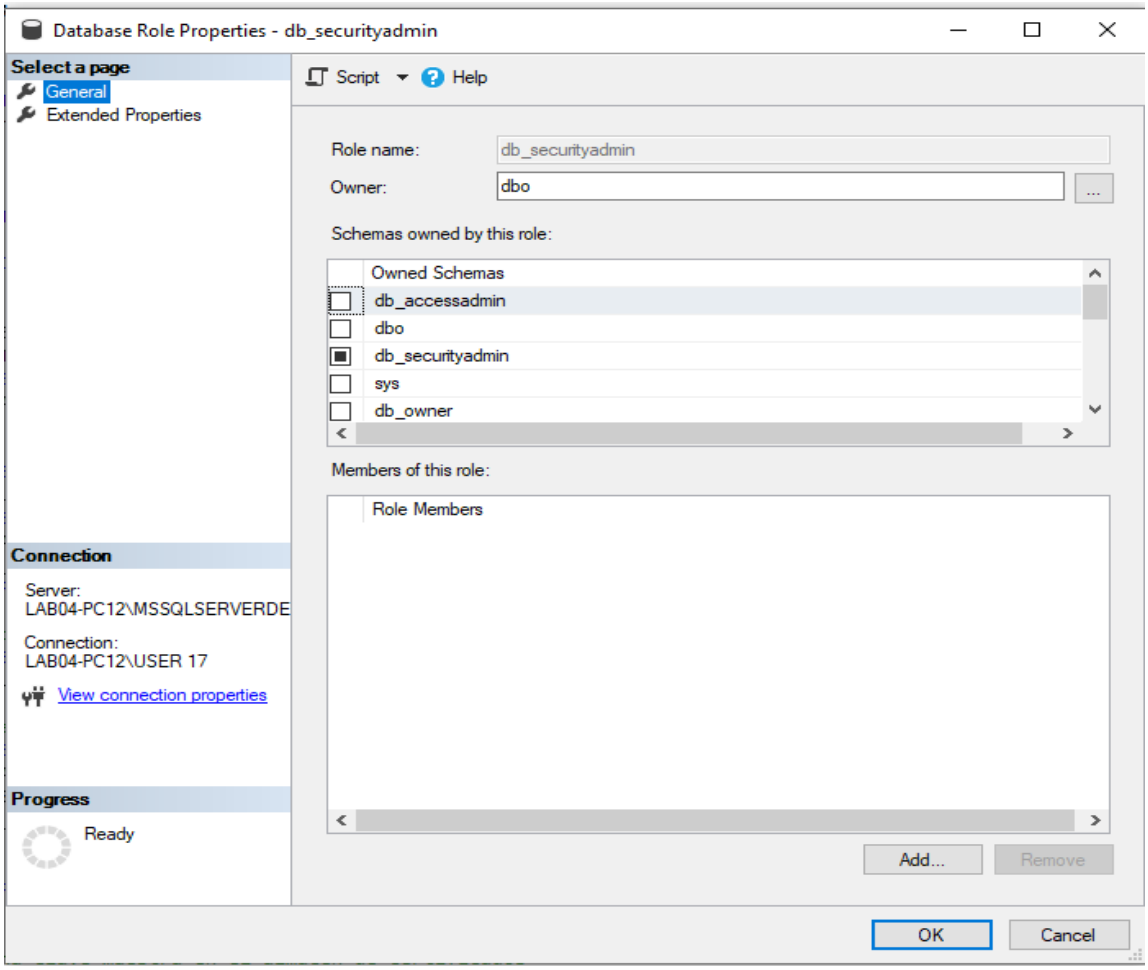
Roles personalizados:

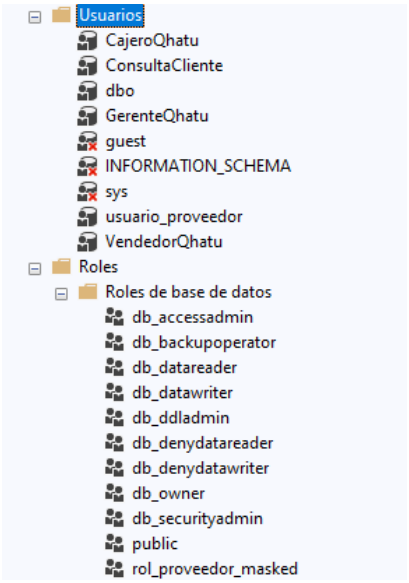
- Puedes crear roles específicos para tu aplicación o usuarios.
- Permiten una administración más granular.

Ejemplo

```
CREATE ROLE Auditor;
GRANT SELECT ON SCHEMA::dbo TO Auditor;
EXEC sp_addrolemember 'Auditor', 'usuario_auditor';
```

Resultado





4. Control de acceso mediante GRANT, DENY y REVOKE

GRANT: Otorga permisos.

DENY: Niega permisos explícitamente (tiene prioridad sobre GRANT).

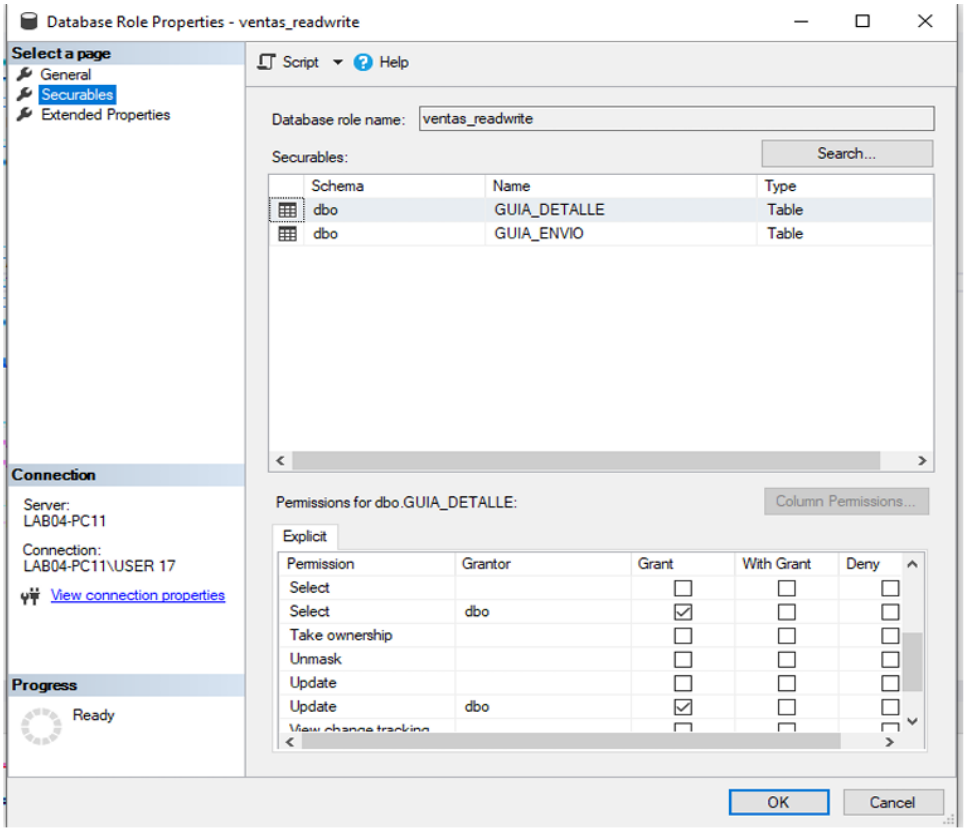
REVOKE: Elimina permisos previamente otorgados.

Ejemplo:

```
GRANT SELECT ON dbo.Estudiantes TO Auditor;
DENY DELETE ON dbo.Estudiantes TO Auditor;
REVOKE SELECT ON dbo.Estudiantes FROM Auditor;
```

```
-- tabla con precios
USE QhatuPeru;

CREATE TABLE dbo.INVENTARIO (
    ID_PRODUCTO INT PRIMARY KEY IDENTITY(1,1),
    NOMBRE NVARCHAR(100),
    STOCK INT,
    PrecioProveedor DECIMAL(10,2),
    PrecioVenta DECIMAL(10,2)
);

-- Crear login y usuario para el analista
CREATE LOGIN login_analista WITH PASSWORD = 'AnalistaSeguro2024!';
CREATE USER usuario_analista FOR LOGIN login_analista;

-- Crear rol de analista
CREATE ROLE rol_analista;
EXEC sp_addrolemember 'rol_analista', 'usuario_analista';

-- Conceder SELECT general sobre la tabla
GRANT SELECT ON dbo.INVENTARIO TO rol_analista;

-- Negar acceso a columnas sensibles
DENY SELECT ON dbo.INVENTARIO (PrecioProveedor, PrecioVenta) TO rol_analista;
```

RESULTADO



5. **Cifrado y protección de datos (TDE, Always Encrypted)**

TDE (Transparent Data Encryption):

- Cifra archivos de base de datos (.mdf, .ldf) en disco.
- Protege contra robo físico del archivo.

Always Encrypted:

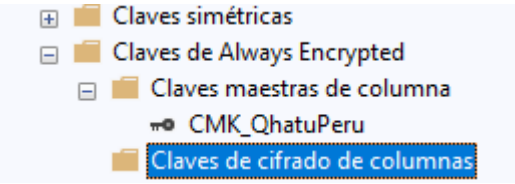
- Cifra datos sensibles (como DNI, tarjetas) en columnas específicas.
- Los datos se cifran/desencriptan en el cliente, no en el servidor.

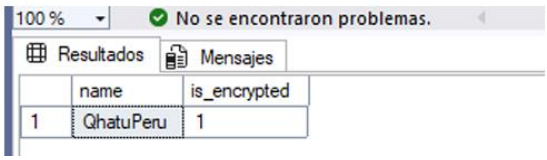
Ejemplo:

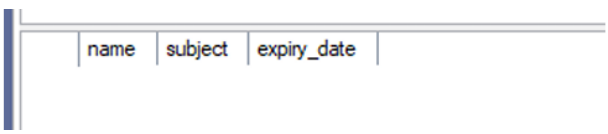
```
CREATE DATABASE ENCRYPTION KEY
WITH ALGORITHM = AES_256
ENCRYPTION BY SERVER CERTIFICATE MiCertificado;
ALTER DATABASE MiBD SET ENCRYPTION ON;
```

```
-- Paso 1: Crear master key en la base master
USE master;
CREATE MASTER KEY ENCRYPTION BY PASSWORD = 'ClaveSeguraMaster2024!';

-- Paso 2: Crear certificado de servidor
CREATE CERTIFICATE Certificado_QhatuPeru
WITH SUBJECT = 'Certificado para TDE en QhatuPeru';

-- Paso 3: Crear la Database Encryption Key en la base QhatuPeru
USE QhatuPeru;
CREATE DATABASE ENCRYPTION KEY
WITH ALGORITHM = AES_256
ENCRYPTION BY SERVER CERTIFICATE Certificado_QhatuPeru;

-- Paso 4: Activar el cifrado
ALTER DATABASE QhatuPeru
SET ENCRYPTION ON;

--
-- Verificar bases cifradas
SELECT name, is_encrypted
FROM sys.databases
WHERE name = 'QhatuPeru';

-- Verificar certificados
SELECT name, subject, expiry_date
FROM sys.certificates
WHERE name = 'Certificado_QhatuPeru';
```

RESULTADO







6. **Auditoría y monitoreo de eventos con SQL Server Audit**

SQL Server Audit:

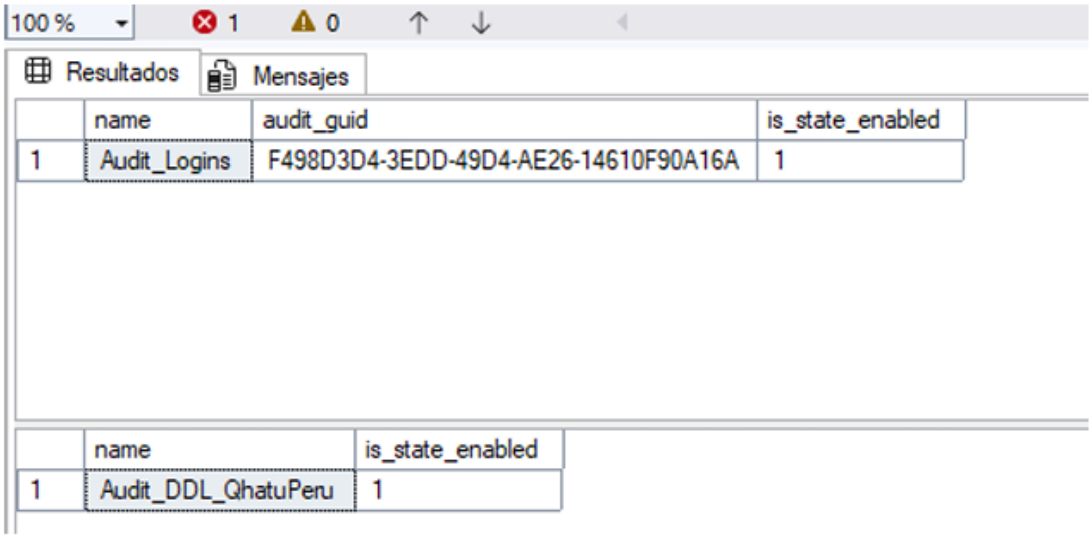
- Permite registrar eventos como acceso a objetos, cambios de configuración, etc.
- Se configura a nivel de servidor o base de datos.

Pasos básicos:

1. Crear un objeto de auditoría.
2. Definir especificaciones (qué auditar).
3. Activar y revisar los logs.

Ejemplo

```
-- verificaciones
-- Auditorías a nivel servidor
SELECT name, audit_guid, is_state_enabled
FROM sys.server_audit_specifications;

-- Auditorías a nivel base de datos
SELECT name, is_state_enabled
FROM sys.database_audit_specifications;


CREATE SERVER AUDIT MiAuditoria
TO FILE (FILEPATH = 'C:\Auditorias\');
CREATE SERVER AUDIT SPECIFICATION MiEspecificacion
FOR SERVER AUDIT MiAuditoria
ADD (SCHEMA_OBJECT_ACCESS_GROUP);
ALTER SERVER AUDIT MiAuditoria WITH (STATE = ON);
```



Prácticas:

- Configurar auditoría para seguimiento de inicios de sesión.
- Implementar un esquema de cifrado para una columna sensible (ej. DNI o tarjeta).

1. Configurar auditoría para seguimiento de inicios de sesión

Registrar quién accede al servidor, cuándo y desde dónde, para detectar accesos no autorizados o patrones sospechosos.

```
//crear el objeto de auditoria
CREATE SERVER AUDIT AuditoriaLogins
TO FILE (FILEPATH = 'C:\Auditorias\Logins');

//creamos la especificacion de la auditoria
CREATE SERVER AUDIT SPECIFICATION EspecificacionLogins
FOR SERVER AUDIT AuditoriaLogins
ADD (SUCCESSFUL_LOGIN_GROUP),
ADD (FAILED_LOGIN_GROUP);

//activamos la auditoria
ALTER SERVER AUDIT AuditoriaLogins WITH (STATE = ON);
```

```
USE master;
--paso 1 creamos el server audit
-- Crear el audit en el servidor, con archivo de salida
CREATE SERVER AUDIT Audit_QhatuPeru
TO FILE (FILEPATH = 'E:\SQLAudit\')
WITH (ON_FAILURE = CONTINUE);

-- Habilitar el audit
ALTER SERVER AUDIT Audit_QhatuPeru
WITH (STATE = ON);

--paso 2
-- Registrar intentos de login
CREATE SERVER AUDIT SPECIFICATION Audit_Logins
FOR SERVER AUDIT Audit_QhatuPeru
ADD (SUCCESSFUL_LOGIN_GROUP),
ADD (FAILED_LOGIN_GROUP)
WITH (STATE = ON);

--paso 3
-- crear la database para cambios DDL
USE QhatuPeru;

CREATE DATABASE AUDIT SPECIFICATION Audit_DDL_QhatuPeru
FOR SERVER AUDIT Audit_QhatuPeru
ADD (SCHEMA_OBJECT_CHANGE_GROUP)
WITH (STATE = ON);
```

Con el código escrito ya estaría configurado y se generarían los archivos .sqlaudit

RESULTADO

2. Implementar un esquema de cifrado para una columna sensible (ej. DNI o tarjeta)

Proteger datos sensibles como el número de DNI o tarjeta de crédito, incluso si alguien accede directamente a la base de datos. Usamos **Always Encrypted**, que cifra los datos en el cliente antes de enviarlos al servidor.

```
//creamos una clave maestra en el almacen de certificados
//creamos una clave de columna
CREATE COLUMN MASTER KEY MiClaveMaestra
WITH (
  KEY_STORE_PROVIDER_NAME = 'MSSQL_CERTIFICATE_STORE',
  KEY_PATH = 'CurrentUser/My/MiCertificado'
);

CREATE COLUMN ENCRYPTION KEY ClaveDNI
WITH VALUES (
  COLUMN_MASTER_KEY = MiClaveMaestra,
  ALGORITHM = 'RSA_OAEP',
  ENCRYPTED_VALUE = <valor_encriptado>
);

//Creamos la tabla columna cifrada
CREATE TABLE Ciudadanos (
  Id INT PRIMARY KEY,
  Nombre NVARCHAR(100),
  DNI CHAR(8) COLLATE Latin1_General_BIN2 ENCRYPTED WITH (
    COLUMN_ENCRYPTION_KEY = ClaveDNI,
    ENCRYPTION_TYPE = DETERMINISTIC,
    ALGORITHM = 'AEAD_AES_256_CBC_HMAC_SHA_256'
  )
);
```

```
CREATE TABLE dbo.PROVEEDORES (
    ID_PROVEEDOR INT PRIMARY KEY IDENTITY(1,1),
    NOMBRE NVARCHAR(100),
    PrecioProveedor_ENC DECIMAL(10,2)
        COLLATE Latin1_General_BIN2
        ENCRYPTED WITH (
            COLUMN_ENCRYPTION_KEY = CEK_QhatuPeru,
            ENCRYPTION_TYPE = Randomized,
            ALGORITHM = 'AEAD_AES_256_CBC_HMAC_SHA_256'
        )
);
```

El cifrado **determinístico** permite búsquedas exactas (`WHERE DNI = '12345678'`).

El cifrado **aleatorio** es más seguro, pero no permite búsquedas directas.

RESULTADO