

# Tarea 4: SEGURIDAD



## 1.- Encriptación

[Veracrypt](#)

[Encriptación de procedimientos](#)

[Encriptación de datos en SSMS](#)

[Encriptación de columnas](#)

[Ejemplo 2 \(encriptación columnas\)](#)

[Encriptar usando frase](#)

[Encriptación de Backups](#)

[Encriptación de TDE \(Transparent data encryption\)](#)

[Hashing](#)

[DDM Dynamic Data Masking](#)

[Secuencias](#)

[ROW LEVEL SECURITY \(RLS\)](#)

[Otro ejemplo de RLS](#)

[Always Encrypted](#)

## 2.- Auditorías

[Auditoría de servidor](#)

[Desde el interfaz gráfico:](#)

[Desde el script](#)

[Auditoría de base de datos](#)

## 3.- Ledger

[Updatable Ledger Table](#)

[Append-only Ledger Table](#)

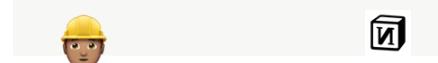
[Prueba detección de modificaciones](#)

## 4.- SQL Data Discovery and Classification in SSMS

## 5.- Ataques contra bases de datos

[Inyección SQL](#)

[Ransomware](#)



TAREA3: Modelado -Adm. SQL ...

Creado con Notion



<https://github.com/Bcoast84/SXBBDD.git>

## 1.- Encriptación

La **encriptación** (o mejor dicho, **cifrado**) es el proceso de convertir información en un formato ilegible para que solo personas autorizadas puedan acceder a ella. Se utiliza para proteger datos y garantizar su confidencialidad.

En nuestro caso, lo que realmente nos interesa es que nadie pueda acceder a nuestra información, ya sea desde dentro de las bases de datos o bien desde fuera, mediante acceso a las mismas o mediante acceso a la información en circulación, capturando esta con sniffers. Para ello, el propio Windows trae un software de encriptación llamado Bitlocker. Para este ejercicio, nosotros hemos decidido usar Veracrypt que viene a ser lo mismo pero externo y de código abierto.

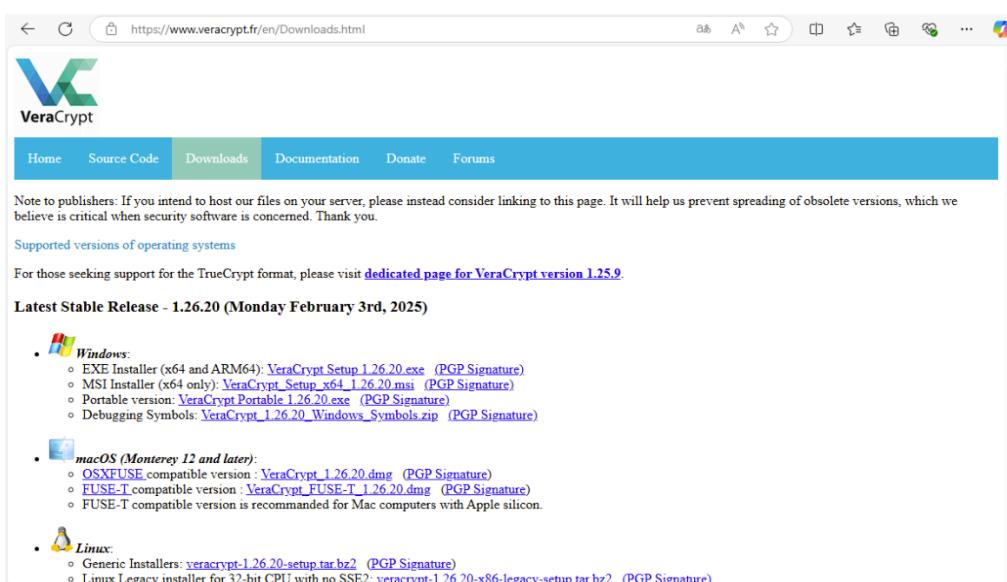
### Veracrypt

VeraCrypt es un software de **cifrado de datos** que permite crear volúmenes cifrados para proteger archivos y discos completos. Es un sucesor mejorado de **TrueCrypt**, diseñado para proporcionar mayor seguridad y resistencia contra ataques de fuerza bruta. Es útil para proteger información confidencial en entornos personales y profesionales, sobre todo en el ámbito de la **ciberseguridad**, donde la privacidad y la integridad de los datos son críticas.

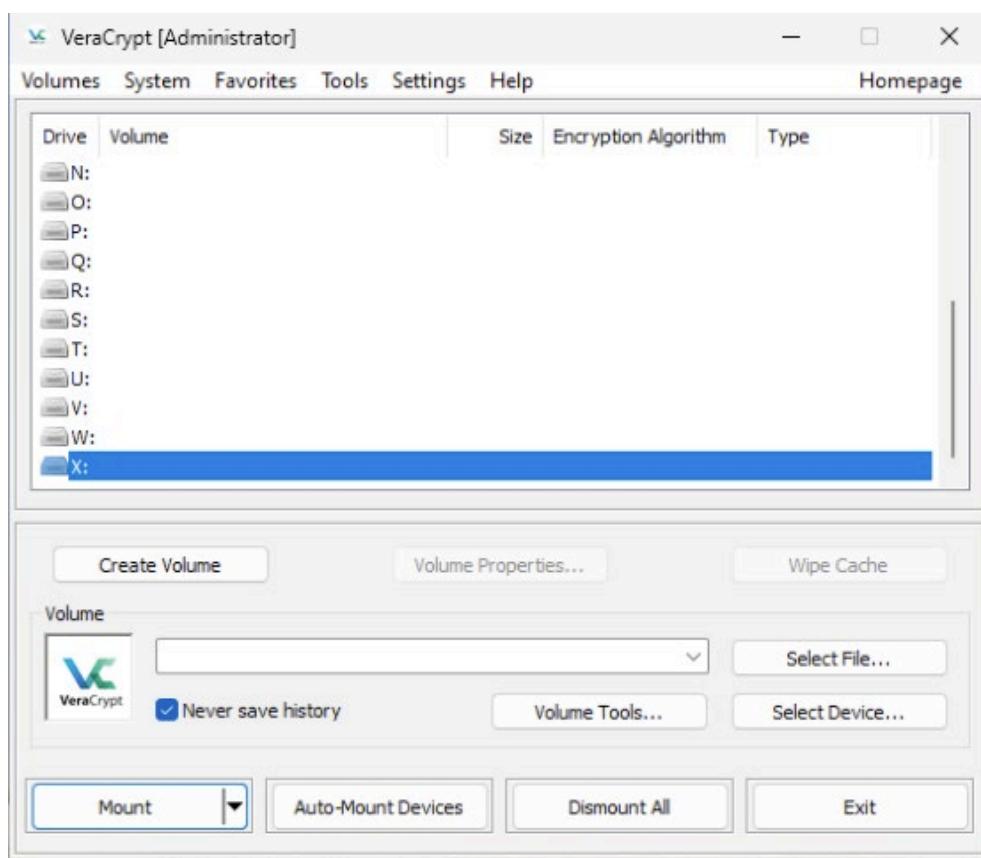
Nos permitirá guardar bases de datos completas o copias de seguridad de las mismas.

Lo primero que haremos será acceder a la web oficial de Veracrypt para descargar el software, como podemos ver, tenemos el software disponible para las plataformas mas usadas. Lo descargamos (en nuestro caso para Windows) y lo instalamos, no hacemos referencia a la instalación puesto que tan solo tendremos que presionar “siguiente” y dejar que la instalación finalice.

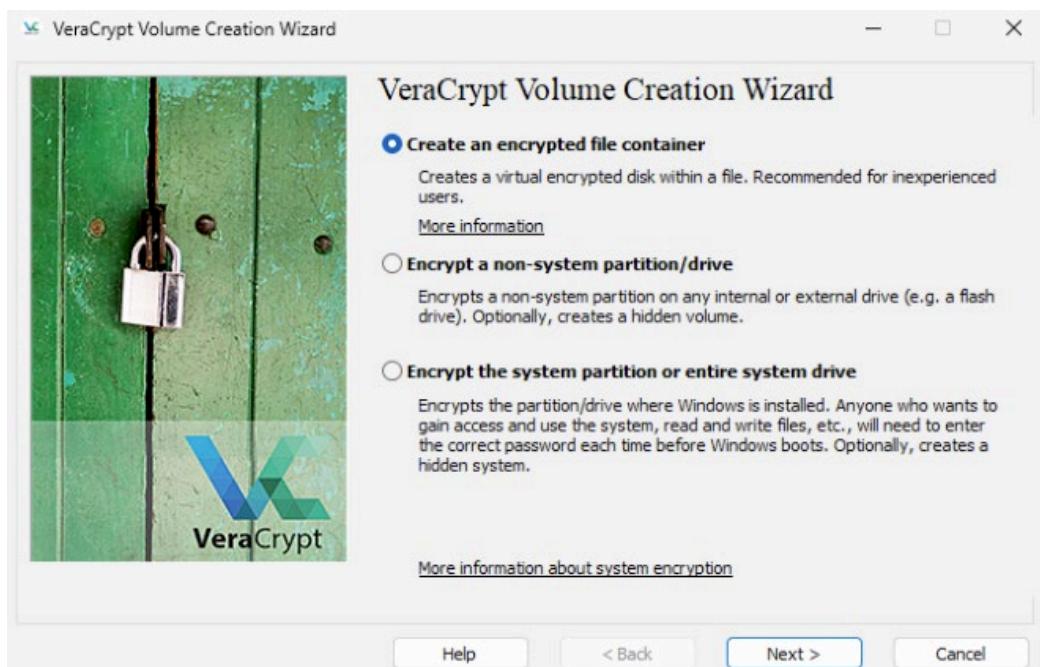
<https://www.veracrypt.fr/en/Downloads.html>



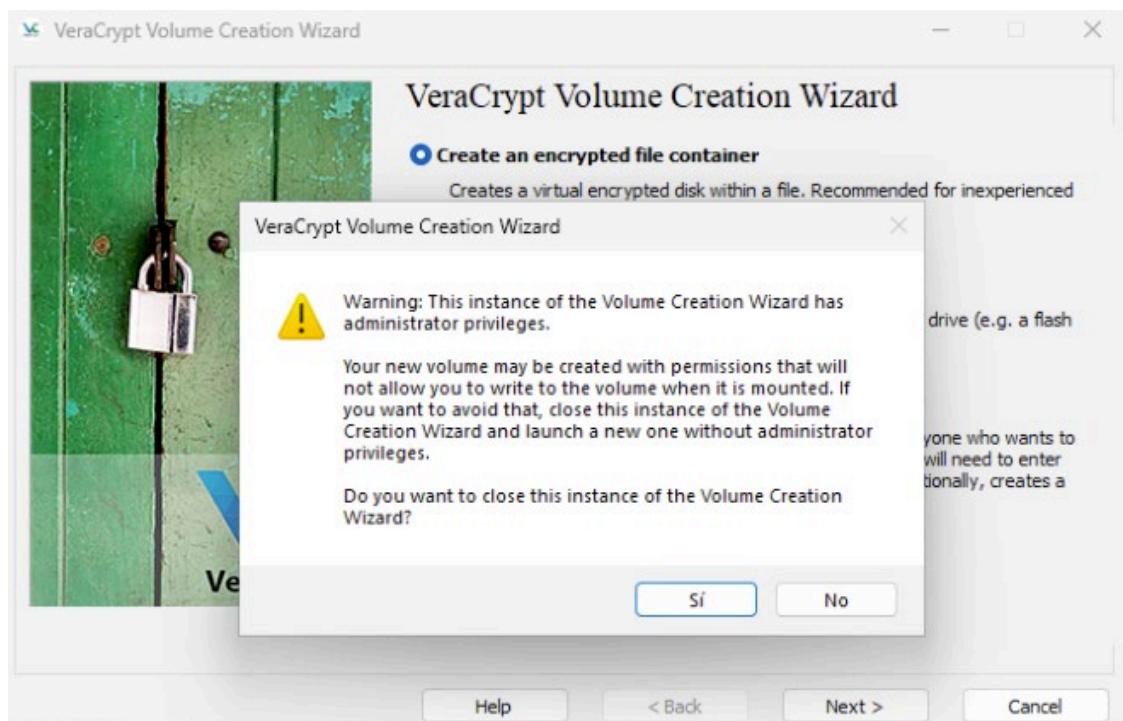
Al abrir el software nos encontramos con la siguiente ventana, tendremos que escoger una letra de unidad y luego presionar en “Create Volume”.



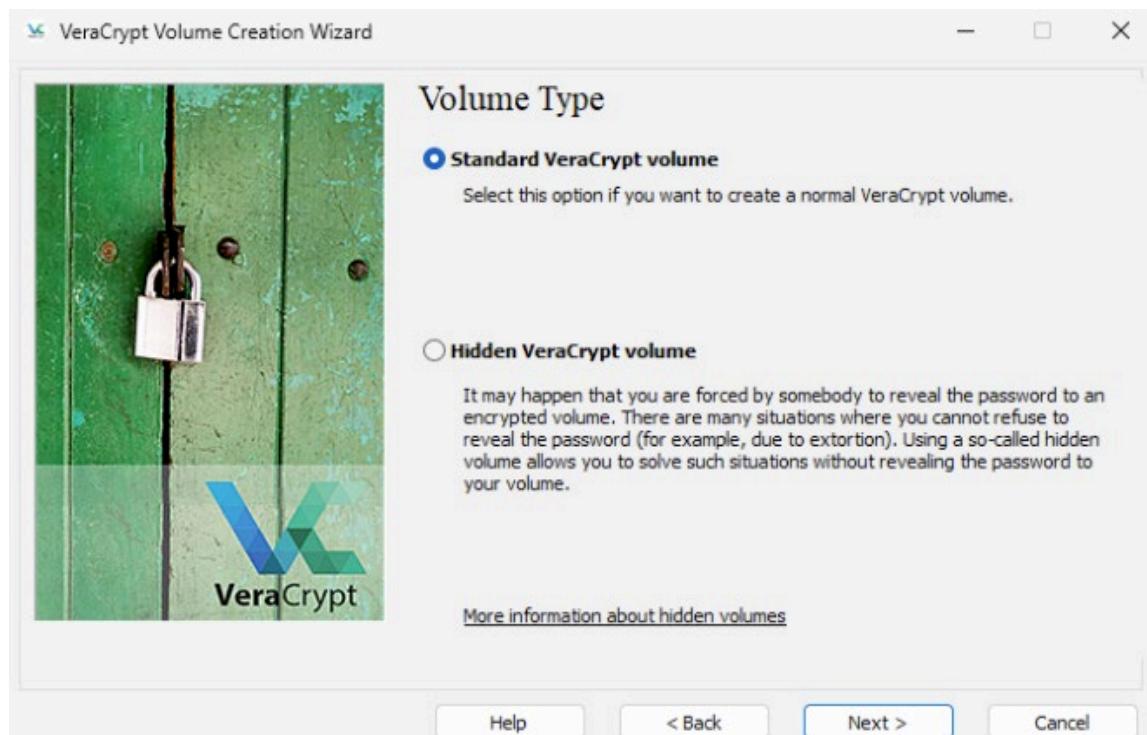
Se nos mostrarán tres opciones, una para crear y encriptar un archivo contenedor, otra para encriptar un driver intero o externo (disco duro entero) y otra opción para encriptar particiones enteras o sistemas operativos. Escogemos la primera opción.



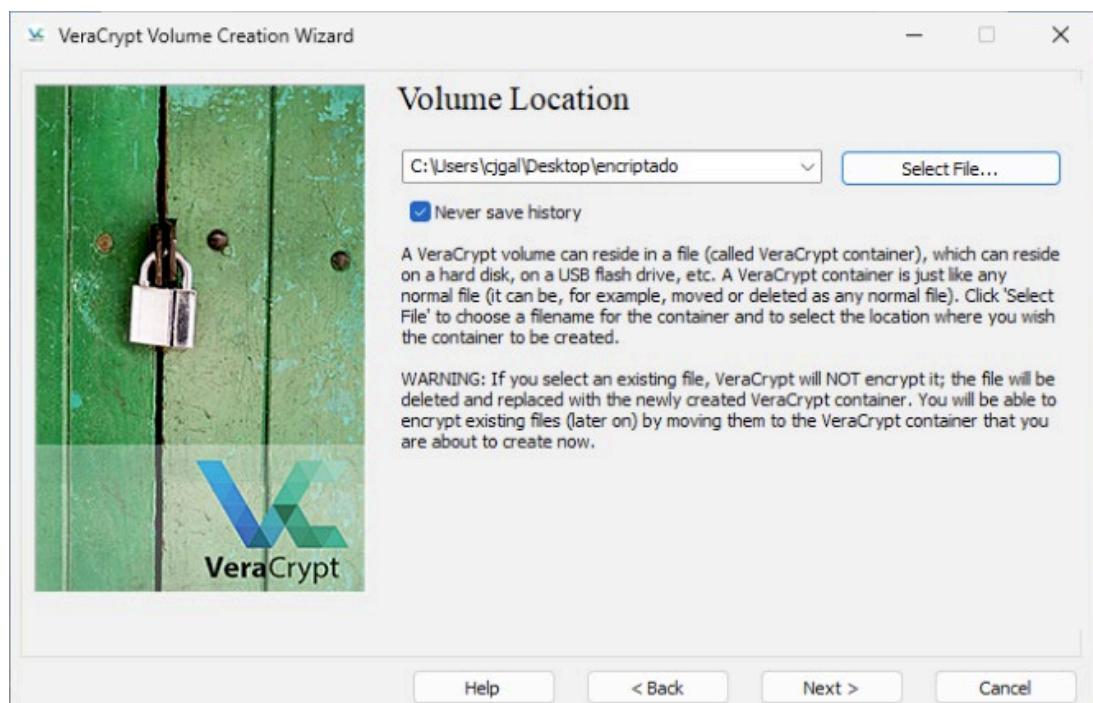
Nos mostrará un warning advirtiéndonos de que vamos a crear un volumen y que si no tenemos privilegios de administrador, puede que luego no podamos acceder a dicho volumen. Presionamos en “No”.



Ahora se nos mostrarán dos opciones, la primera una encriptación normal y la segunda una encriptación doble para que dependiendo de la contraseña que se introduzca al desencriptar, se muestren unos archivos u otros. Seleccionamos la primera opción.



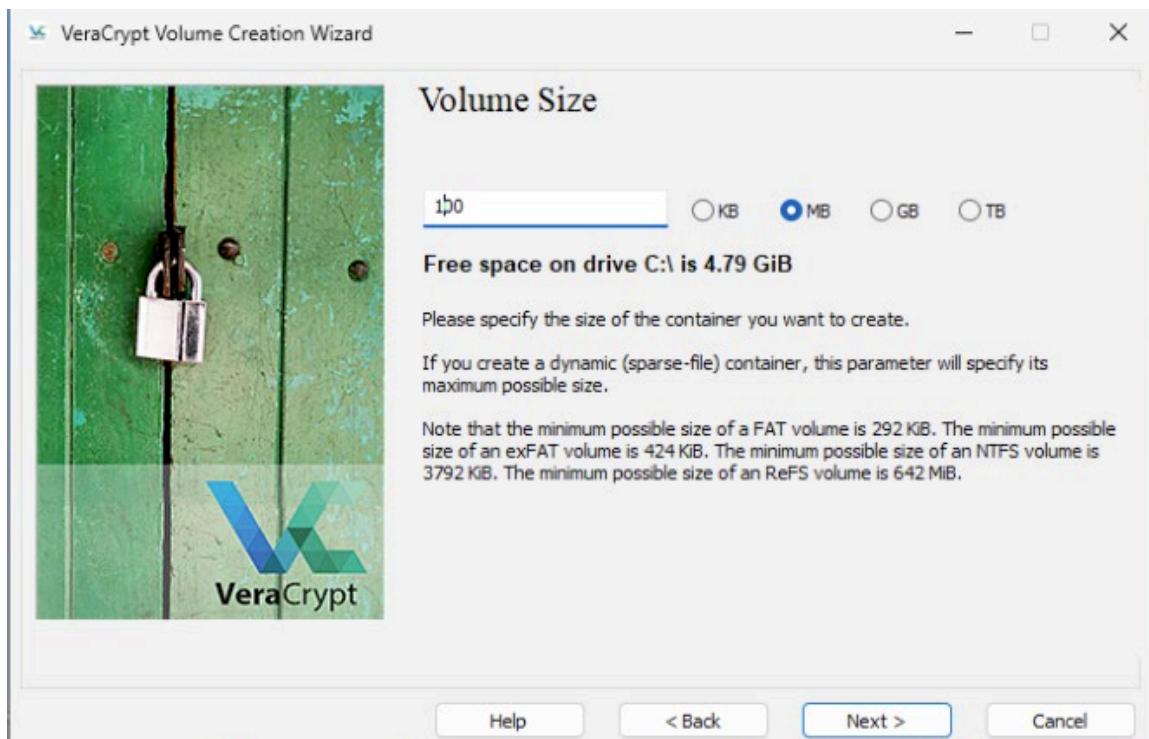
Ahora seleccionamos el directorio del archivo donde se va a guardar la información encriptada.



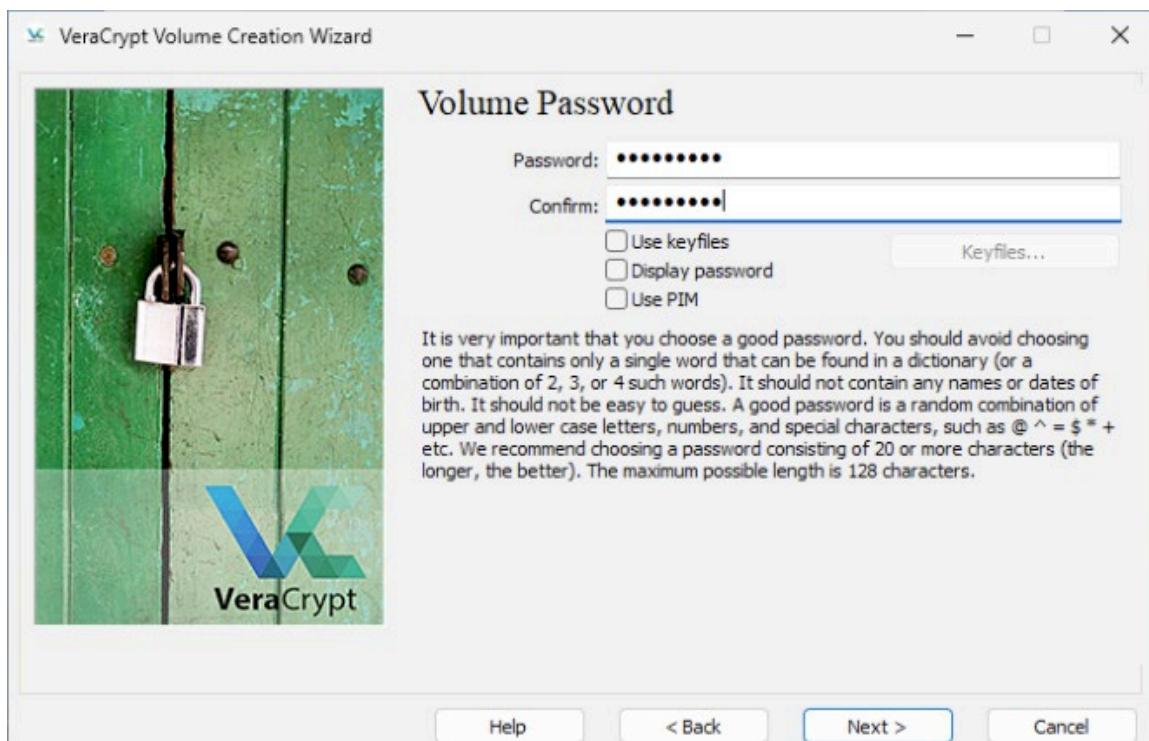
En el siguiente paso seleccionaremos el método de encriptación, en este caso dejaremos las opciones tal cual están ya que AES y 512 es más que suficiente para dar seguridad a los archivos.



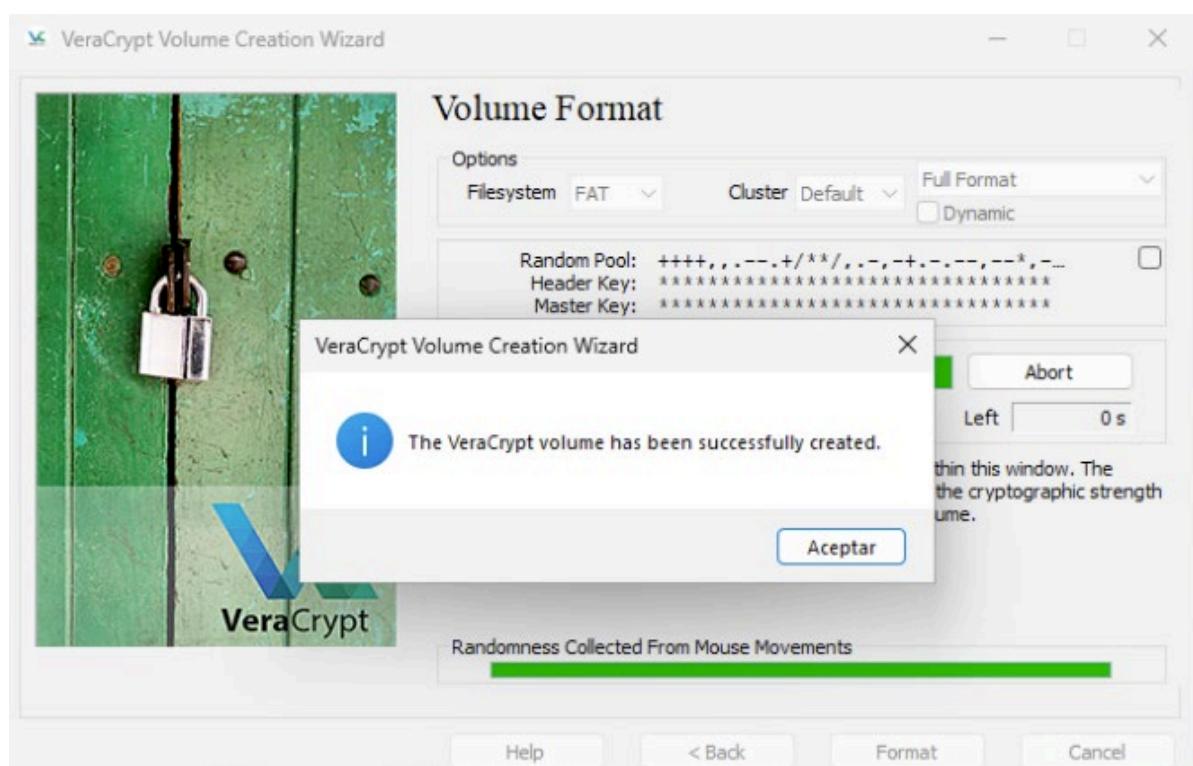
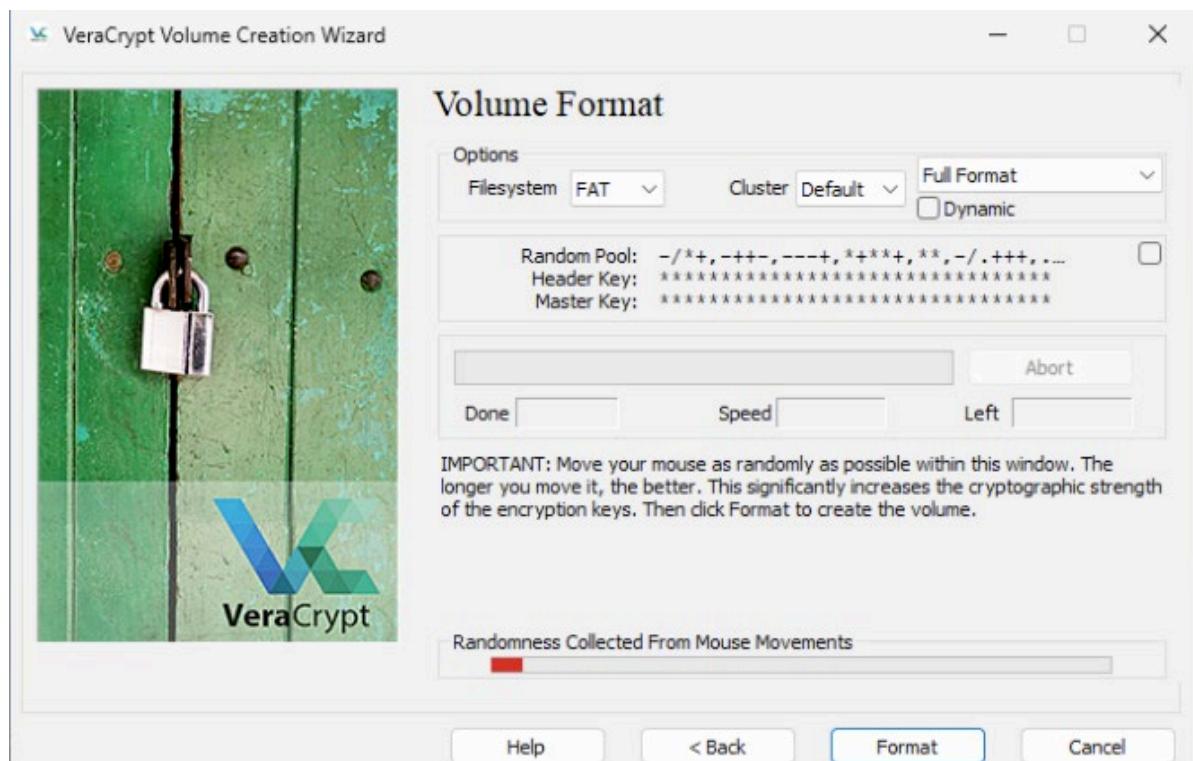
Por último, escogeremos el tamaño que le queremos dar al volumen encriptado, debemos tener en cuenta que nuestros archivos podrán crecer de tamaño, por este motivo debemos crear el volumen con algo de margen dependiendo de lo que queramos encriptar. En este caso como vamos a usar archivos pequeños, crearemos un volumen de 100Mb.



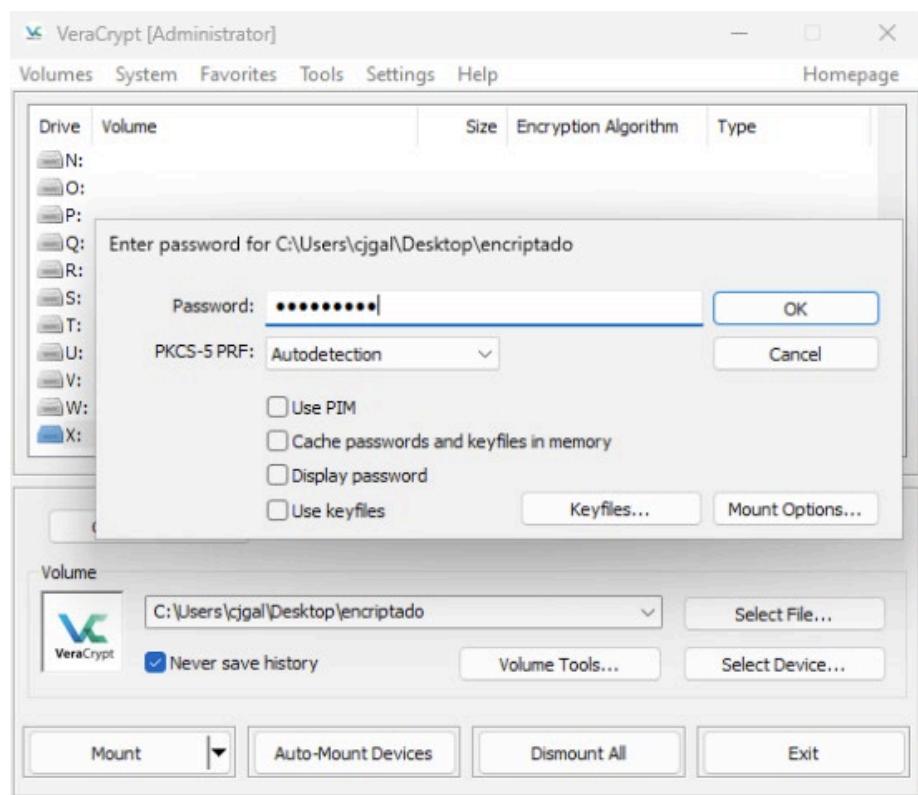
Añadimos un password, hemos puesto la que venimos usando en todas las contraseñas desde principio de curso ‘Abcd1234.’ Aunque nos mostrará un warning diciendo que la contraseña es demasiado corta para ser segura, presionamos continuar.



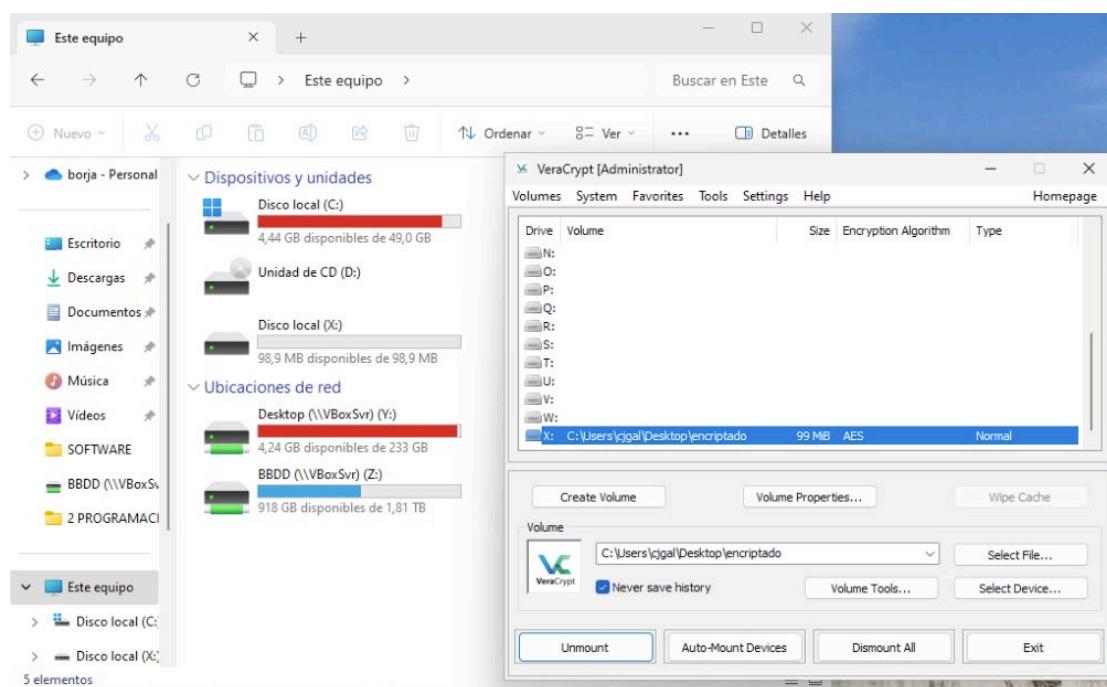
Ahora se encriptará el volumen, debemos tener en cuenta que cuanto mayor tamaño tenga el volumen, mas tiempo tardará en encriptarse. Debemos mover el ratón continuamente hasta que se llene la barra inferior. Una vez termine presionaremos en “Format”.



Ahora tendremos nuestro volumen encriptado en la ubicación que hemos designado, como podemos ver, se trata de un archivo sin extensión que no podremos abrir. Para abrirlo, usaremos de nuevo el software de Veracrypt. Entramos y esta vez seleccionamos el archivo y presionaremos en “mount”, nos pedirá la contraseña que hemos configurado.



Como podemos ver, se ha creado una unidad del tamaño que hemos designado, ahora bastará con soltar dentro de la unidad los archivos que queremos encriptar y luego, desde el software de Veracrypt presionar en Unmount para desmontar y que el archivo quede encriptado.



## Encriptación de procedimientos

Encriptar un procedimiento es algo tan sencillo como añadir “WITH ENCRYPTION” al código del mismo. De esta forma, nadie podrá ver el contenido del mismo. Vamos a generar un procedimiento que al introducir una matrícula, se elimine la misma de la base de datos de nuestra autoescuela si existiese y de lo contrario muestre un error (aunque en este caso no tiene mucho sentido encriptar el procedimiento, podría interesarnos que nos supiese, por ejemplo, el método que usamos para subir el sueldo a los empleados o la fórmula con la que un procedimiento calcula la productividad de un empleado).

```
CREATE PROCEDURE sp_EliminarVehiculo
    @Matricula NVARCHAR(10)
    WITH ENCRYPTION -- Con encriptación
    AS
    BEGIN
        SET NOCOUNT ON;
        -- Verificar si la matrícula existe en la tabla
        IF EXISTS (SELECT 1 FROM vehiculo WHERE Matricula_vehiculo = @Matricula)
        BEGIN
            DELETE FROM vehiculo WHERE Matricula_vehiculo = @Matricula;
            PRINT 'El vehículo con matrícula ' + @Matricula + ' ha sido eliminado
correctamente.';
        END
        ELSE
        BEGIN
            PRINT 'La matrícula introducida no existe en la base de datos.';
        END
    END;
```

Ahora, si intentamos acceder al código del mismo, comprobamos que se nos muestra un error.

```
-- Intentamos ver el código del procedimiento:
SP_HELPTEXT sp_EliminarVehiculo;
GO
100 %
Messages
The text for object 'sp_EliminarVehiculo' is encrypted.
Completion time: 2025-02-10T17:32:42.6587293+01:00
```

## Encriptación de datos en SSMS

### Encriptación de columnas

El sentido de encriptar una columnas no es más que cifrar ciertos datos de una tabla que no queremos que se muestren a un determinado usuario o grupos de usuarios. (muy útil en la actualidad para la protección de datos)

Necesitaremos realizar los siguientes paso:

- Crear una clave “master key”
- Crear los certificados
- Crear la “Symmetric key”.

En este ejemplo vamos a usar la encriptación de columnas para que cada profesor de la autoescuela, solo pueda acceder a los datos de sus alumnos y no a los datos del resto de alumnos del centro.

Para crear la master key, usaremos:

```
DROP MASTER KEY
GO
CREATE master KEY encryption BY password = 'Abcd1234.'
```

Comprobamos que la hemos creado correctamente:

```
SELECT name KeyName,
       symmetric_key_id KeyID,
       key_length KeyLength,
       algorithm_desc KeyAlgorithm
  FROM sys.symmetric_keys;
GO
87  -- Se comprueba
88  SELECT name KeyName,
89    symmetric_key_id KeyID,
90    key_length KeyLength,
91    algorithm_desc KeyAlgorithm
92  FROM sys.symmetric_keys;
93  GO
94
```

Results Messages

KeyName	KeyID	KeyLength	KeyAlgorithm
#MS_DatabaseMasterKey##	101	256	AES_256

Para crear el certificado usaremos:

```
CREATE CERTIFICATE profelcert AUTHORIZATION profe1
WITH subject = 'Abcd1234.', EXPIRY_DATE = '2025-12-31';
GO
CREATE CERTIFICATE profe2cert AUTHORIZATION profe2
WITH subject = 'Abcd1234.', EXPIRY_DATE = '2025-12-31';
GO
```

Comprobamos que los hemos creado:

```
SELECT name CertName,
       certificate_id CertID,
       pvt_key_encryption_type_desc EncryptType,
       issuer_name Issuer
  FROM sys.certificates;
GO
103  -- Comprobamos
104  SELECT name CertName,
105    certificate_id CertID,
106    pvt_key_encryption_type_desc EncryptType,
107    issuer_name Issuer
108  FROM sys.certificates;
109  GO
```

Results Messages

CertName	CertID	EncryptType	Issuer
profe1cert	256	ENCRYPTED_BY_MASTER_KEY	Abcd1234.
profe2cert	258	ENCRYPTED_BY_MASTER_KEY	Abcd1234.

Por último, ya solo nos queda crear la clave simétrica:

```

CREATE SYMMETRIC KEY SK_01
    WITH ALGORITHM = AES_256
    ENCRYPTION BY CERTIFICATE profelcert;
GO

CREATE SYMMETRIC KEY SK_02
    WITH ALGORITHM = AES_256
    ENCRYPTION BY CERTIFICATE profe2cert;
GO

122  -- Comprobamos
123  SELECT name KeyName,
124      symmetric_key_id KeyID,
125      key_length KeyLength,
126      algorithm_desc KeyAlgorithm
127  FROM sys.symmetric_keys;
128  GO
129

```

Results

	KeyName	KeyID	KeyLength	KeyAlgorithm
1	##MS_DatabaseMasterKey##	101	256	AES_256
2	SK_01	258	256	AES_256
3	SK_02	259	256	AES_256

Ahora vamos a realizar las pruebas, primero abriremos la clave simétrica para usarla

```

OPEN SYMMETRIC KEY SK_01
    DECRYPTION BY CERTIFICATE profelcert
GO

163  -- Abrimos la clave simétrica
164  OPEN SYMMETRIC KEY SK_01
165      DECRYPTION BY CERTIFICATE profelcert
166  GO
167
168  SELECT * FROM sys.openkeys
170

```

Results

	database_id	database_name	key_id	key_name	key_guid	opened_date	status
1	12	autoescuelaP	258	SK_01	DAE5F900-3D76-4A3B-8C4E-F5087958AECA	2025-03-18 19:01:49.567	1

Ahora insertamos datos en la tabla “alumnos” encriptando teléfono y dirección con profel:

```

INSERT INTO [autoescuelaP].[dbo].[alumnos] (
    [DNI_cliente],
    [N_cliente],
    [Apell_cliente],
    [Apel2_cliente],
    [Dir_cliente],
    [Email_cliente],
    [Telf_cliente],
    [localidad_cod_localidad]
)
VALUES
(
    '12345678A',
    'Carlos',
    'Garcia',
    'López',
    EncryptByKey(Key_GUID('SK_01'), 'Calle Gran Vía 123, Madrid'),
    'carlos.garcia@mail.com',
    EncryptByKey(Key_GUID('SK_01'), '637345678'),
    28001
),
(

```

```

'87654321B',
'Laura',
'Martínez',
'Sánchez',
EncryptByKey(Key_GUID('SK_01'), 'Avenida Diagonal 456, Barcelona'),
'laura.martinez@mail.com',
EncryptByKey(Key_GUID('SK_01'), '633456789'),
08001
),
(
'11223344C',
'Miguel',
'Fernández',
'Rodríguez',
EncryptByKey(Key_GUID('SK_01'), 'Plaza del Ayuntamiento 7, Valencia'),
'miguel.fernandez@mail.com',
EncryptByKey(Key_GUID('SK_01'), '663123456'),
46001
);

```

**cerramos la clave:**

```
CLOSE SYMMETRIC KEY SK_01;
GO
```

Repetimos el proceso impersonando en profe2 e insertando otros 3 alumnos:

**Abrimos la clave:**

```
OPEN SYMMETRIC KEY SK_02
    DECRYPTION BY CERTIFICATE profe2cert
GO
```

**Añadimos datos:**

```

INSERT INTO [autoescuelaP].[dbo].[alumnos] (
    [DNI_cliente],
    [N_cliente],
    [Apell_cliente],
    [Apel2_cliente],
    [Dir_cliente],
    [Email_cliente],
    [Telf_cliente],
    [localidad_cod_localidad]
)
VALUES
(
    '55443322L',
    'Ana',
    'Ruiz',
    'Gómez',
    EncryptByKey(Key_GUID('SK_02'), 'Calle Sierpes 45, Sevilla'),
    'ana.ruiz@hotmail.com',
    EncryptByKey(Key_GUID('SK_02'), '654987321'),
    41001
),
(
    '98765432M',
    'Javier',
    'Hernández',
    'Díaz',
    EncryptByKey(Key_GUID('SK_02'), 'Gran Vía 22, Bilbao'),
    'javier.hernandez@gmail.com',
    EncryptByKey(Key_GUID('SK_02'), '688112233'),
    48001
),
(
    '11223344N',
    'Sofía',
    'Jiménez',
    'Moreno',
    EncryptByKey(Key_GUID('SK_02'), 'Paseo Independencia 10, Zaragoza'),
    'sofia.jimenez@yahoo.com',

```

```
EncryptByKey(Key_GUID('SK_02'), '676543219'),  
50001  
);
```

Cerramos clave:

```
CLOSE SYMMETRIC KEY SK_02;  
GO
```

Ahora realizamos una consulta normal y observamos que los campos aparecen encriptados:

297	298	299	300				
	SELECT * FROM alumnos;						
	GO						
100 %							
	DNI_cliente	N_cliente	Apel1_cliente	Apel2_cliente	Dir_cliente	Email_cliente	Telf_cliente
1	11223344C	Miguel	Fernández	Rodríguez	0x00F9E5DA763D3B4A8C4EF5087958AECA020000009F8E321...	miguel.fernandez@mail.com	0x00F9E5DA763D3B4A8C4EF5087958AECA02000...
2	11223344N	Sofía	Jiménez	Moreno	0x001DC63B57E2644B86DFFFA7B273FF4002000004F60092...	sofia.jimenez@yahoo.com	0x001DC63B57E2644B86DFFFA7B273FF4002000...
3	12345678A	Carlos	García	López	0x00F9E5DA763D3B4A8C4EF5087958AECA0200000062B7A8...	carlos.garcia@mail.com	0x00F9E5DA763D3B4A8C4EF5087958AECA02000...
4	55443322L	Ana	Ruiz	Gómez	0x001DC63B57E2644B86DFFFA7B273FF400200000036DF35C...	ana.ruiz@hotmail.com	0x001DC63B57E2644B86DFFFA7B273FF4002000...
5	87654321B	Laura	Martínez	Sánchez	0x00F9E5DA763D3B4A8C4EF5087958AECA020000007C0A9E...	laura.martinez@mail.com	0x00F9E5DA763D3B4A8C4EF5087958AECA02000...
6	98765432M	Javier	Hernández	Díaz	0x001DC63B57E2644B86DFFFA7B273FF4002000000BE07F23...	javier.hernandez@gmail.com	0x001DC63B57E2644B86DFFFA7B273FF4002000...

Vamos a realizar una consulta de los datos desde profel y como profe2 desencriptando los datos:

```
308 -- Ejecutado como profe1:  
309 OPEN SYMMETRIC KEY SK_01 DECRYPTION BY CERTIFICATE profe1cert;  
310 SELECT  
311     DNI_cliente,  
312     CONVERT(VARCHAR, DecryptByKey(Dir_cliente)) AS Direccion,  
313     CONVERT(VARCHAR, DecryptByKey(Telf_cliente)) AS Telefono  
314 FROM alumnos;  
315 CLOSE SYMMETRIC KEY SK_01;
```

```
322  
323 -- Ejecutado como profe2:  
324 OPEN SYMMETRIC KEY SK_02 DECRYPTION BY CERTIFICATE profe2cert;  
325 SELECT  
326     DNI_cliente,  
327     CONVERT(VARCHAR, DecryptByKey(Dir_cliente)) AS Direccion,  
328     CONVERT(VARCHAR, DecryptByKey(Telf_cliente)) AS Telefono  
329 FROM alumnos;  
330 CLOSE SYMMETRIC KEY SK_02;
```

Como podemos ver, los campos que el profesor que consulta no ha introducido, no es capaz de desencriptarlos y aparecen como “NULL”.

**\*\*Importante recordar que para poder borrar el certificado, debemos borrar antes la clave, sino no nos dejará por estar trabajando el certificado con la clave.**

### Ejemplo 2 (encriptación columnas)

Vamos a crear un sistema para registrar pagos de alumnos, los datos de la tarjeta de crédito se almacenarán encriptados y cada profesor solo podrá ver los datos de los pagos que registró

#### Creamos la tabla de pagos

```
CREATE TABLE [autoescuelaP].[dbo].[pagos] (
    [ID_pago] INT IDENTITY(1,1) PRIMARY KEY,
    [DNI_alumno] VARCHAR(9) NOT NULL,
    [Monto] DECIMAL(10,2) NOT NULL,
    [Fecha_pago] DATE NOT NULL,
    [Tarjeta_credito] VARBINARY(MAX),
    [Titular_tarjeta] VARBINARY(MAX)
);
GO
```

#### Otorgamos permisos:

```
GRANT SELECT, INSERT ON [pagos] TO profel, profe2;
GO
```

#### Creamos el procedimiento para añadir los pagos:

```
CREATE PROCEDURE SP_RegistrarPagoEncriptado
    @DNI_alumno VARCHAR(9),
    @Monto DECIMAL(10,2),
    @Fecha_pago DATE,
    @Tarjeta_credito VARCHAR(19),
    @Titular_tarjeta VARCHAR(100)
WITH EXECUTE AS CALLER -- Esto hace que se ejecute con los permisos del que lo llama.
AS
BEGIN
    DECLARE @ClaveSimetrica VARCHAR(10);

    -- Usa la clave según el usuario
    SET @ClaveSimetrica =
        CASE USER_NAME()
            WHEN 'profel' THEN 'SK_01'
            WHEN 'profe2' THEN 'SK_02'
        END;

    -- Abrimos la clave necesaria
    IF @ClaveSimetrica = 'SK_01'
    BEGIN
        OPEN SYMMETRIC KEY SK_01 DECRYPTION BY CERTIFICATE profelcert;
    END
    ELSE IF @ClaveSimetrica = 'SK_02'
    BEGIN
        OPEN SYMMETRIC KEY SK_02 DECRYPTION BY CERTIFICATE profe2cert;
    END

    -- Insertamos los datos con tarjeta y titular encriptados
    INSERT INTO [autoescuelaP].[dbo].[pagos] (
        [DNI_alumno],
        [Monto],
        [Fecha_pago],
        [Tarjeta_credito],
        [Titular_tarjeta]
    )
    VALUES (
        @DNI_alumno,
        @Monto,
        @Fecha_pago,
```

```

        EncryptByKey(Key_GUID(@ClaveSimetrica), @Tarjeta_credito),
        EncryptByKey(Key_GUID(@ClaveSimetrica), @Titular_tarjeta)
    );

    -- Cerramos la clave
    IF @ClaveSimetrica = 'SK_01'
    BEGIN
        CLOSE SYMMETRIC KEY SK_01;
    END
    ELSE IF @ClaveSimetrica = 'SK_02'
    BEGIN
        CLOSE SYMMETRIC KEY SK_02;
    END
END;
** Debemos otorgar permisos a los usuarios para ejecutar el procedimiento almacenado con GRANT EXECUTE ON [dbo].[SP_RegistrarPagoEncriptado] TO profel, profe2

```

Ahora insertamos pagos como profel:

```

EXEC AS USER = 'profel';
GO

EXEC SP_RegistrarPagoEncriptado
    @DNI_alumno = '12345678A',
    @Monto = 250.50,
    @Fecha_pago = '2025-03-20',
    @Tarjeta_credito = '4111-1111-1111-1111',
    @Titular_tarjeta = 'Carlos Garcia Lopez';
GO

```

Insertamos los pagos como profe2:

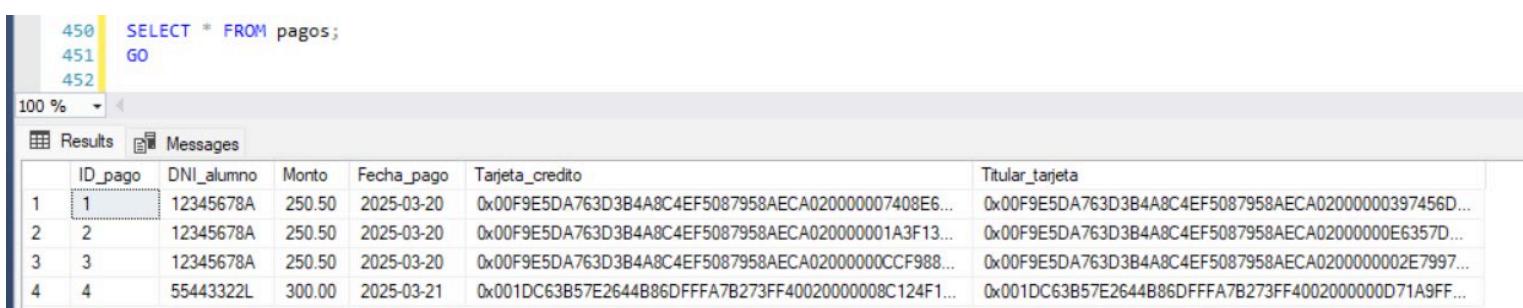
```

EXEC AS USER = 'profe2';
GO

EXEC SP_RegistrarPagoEncriptado
    @DNI_alumno = '55443322L',
    @Monto = 300.00,
    @Fecha_pago = '2025-03-21',
    @Tarjeta_credito = '5500-0000-0000-0004',
    @Titular_tarjeta = 'Ana Ruiz Gomez';
GO

```

Comprobamos:



The screenshot shows the SSMS interface with the following details:

- Query Editor:** Contains the following T-SQL code:
 

```
450  SELECT * FROM pagos;
451  GO
452
```
- Status Bar:** Shows "100 %".
- Results Tab:** Contains a table with the following data:

ID_pago	DNI_alumno	Monto	Fecha_pago	Tarjeta_credito	Titular_tarjeta
1	12345678A	250.50	2025-03-20	0x00F9E5DA763D3B4A8C4EF5087958AECA020000007408E6...	0x00F9E5DA763D3B4A8C4EF5087958AECA02000000397456D...
2	12345678A	250.50	2025-03-20	0x00F9E5DA763D3B4A8C4EF5087958AECA020000001A3F13...	0x00F9E5DA763D3B4A8C4EF5087958AECA02000000E6357D...
3	12345678A	250.50	2025-03-20	0x00F9E5DA763D3B4A8C4EF5087958AECA02000000CCF988...	0x00F9E5DA763D3B4A8C4EF5087958AECA020000002E7997...
4	55443322L	300.00	2025-03-21	0x001DC63B57E2644B86DFFA7B273FF40020000008C124F1...	0x001DC63B57E2644B86DFFA7B273FF4002000000D71A9FF...

```

453 EXEC AS USER = 'profe1';
454 OPEN SYMMETRIC KEY SK_01 DECRYPTION BY CERTIFICATE profe1cert;
455
456 SELECT
457     CONVERT(VARCHAR, DecryptByKey(Tarjeta_credito)) AS Tarjeta,
458     CONVERT(VARCHAR, DecryptByKey(Titular_tarjeta)) AS Titular
459 FROM pagos;
460
461 CLOSE SYMMETRIC KEY SK_01;
462 REVERT;
463 GO

```

100 %

Results Messages

	Tarjeta	Titular
1	4111-1111-1111-1111	Carlos García López
2	4111-1111-1111-1111	Carlos García López
3	4111-1111-1111-1111	Carlos García López
4	NULL	NULL

```

465 EXEC AS USER = 'profe2';
466 OPEN SYMMETRIC KEY SK_02 DECRYPTION BY CERTIFICATE profe2cert;
467
468 SELECT
469     CONVERT(VARCHAR, DecryptByKey(Tarjeta_credito)) AS Tarjeta,
470     CONVERT(VARCHAR, DecryptByKey(Titular_tarjeta)) AS Titular
471 FROM pagos;
472
473 CLOSE SYMMETRIC KEY SK_02;

```

100 %

Results Messages

	Tarjeta	Titular
1	NULL	NULL
2	NULL	NULL
3	NULL	NULL
4	5500-0000-0000-0004	Ana Ruiz Gómez

### Encriptar usando frase

Para encriptar con una frase en vez de con una clave, tan solo tenemos que usar *EncryptByPassPhrase*, en este caso vamos a guardar la frase en una variable para luego aplicarla.

```

DECLARE @FraseSecreta VARCHAR(100) = 'Voy aprobar bases de datos';
UPDATE [profesor]
    SET profesor_Dir_profe = EncryptByPassPhrase(@FraseSecreta, 'Calle Gran Vía 78,
Madrid')
    WHERE profesor_DNI_profe = '09876543Q';

```

```

479
480 -- Encriptar con frase
481 SELECT * FROM profesor;
482 GO
483 -- Declaramos una variable y almacenamos la frase
484 DECLARE @FraseSecreta VARCHAR(100) = 'Voy aprobar bases de datos';
485
486 -- añadimos datos usando la frase
487 UPDATE [profesor]
488     SET profesor_Dir_profe = EncryptByPassPhrase(@FraseSecreta, 'Calle Gran Vía 78, Madrid')
489     WHERE profesor_DNI_profe = '09876543Q';
490
491 UPDATE [autoescuelaP].[dbo].[profesor]
492     SET profesor_Dir_profe = EncryptByPassPhrase(@FraseSecreta, 'Avenida Diagonal 123, Barcelona')
493     WHERE profesor_DNI_profe = '10987654R';
494
495 SELECT * FROM profesor;
496 GO

```

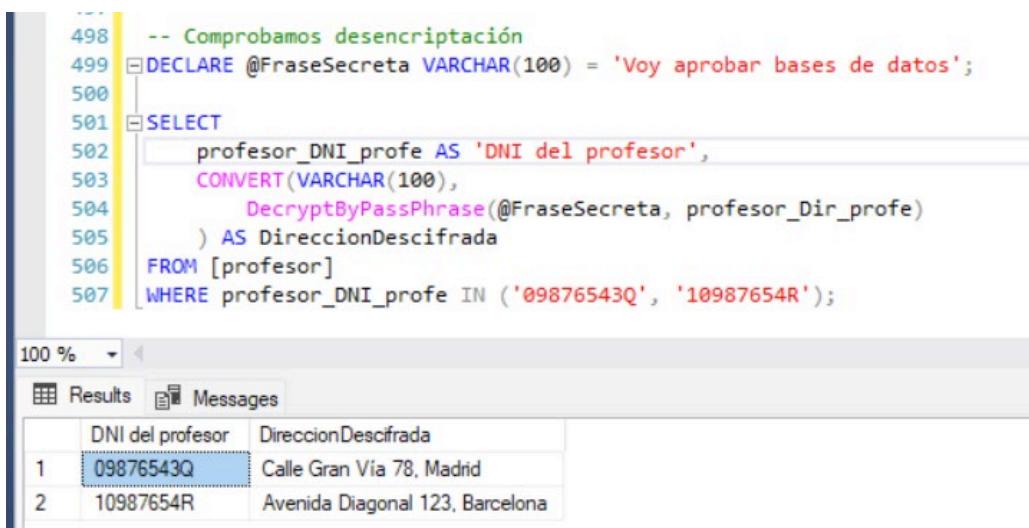
100 %

Results Messages

	profesor_DNI_profe	profesor_N_profe	profesor_Apel1_profe	profesor_Apel2_profe	profesor_Email_profe	profesor_Telf_profe	profesor_Dir_profe	localida...
1	09876543Q	Carmen	Moreno	Alonso	c.moreno@autoescuela.com	609876543	0x02000000C476EAB8E423B9D3C0023CECC8DC8C2BF128BD...	10
2	10987654R	Alberto	Jiménez	Rodríguez	a.jimenez@autoescuela.com	610987654	0x02000000A6CA6BEAFF176F1A9191E73F368849FDF821489...	9
3	21098765S	Isabel	Pérez	Martín	i.perez@autoescuela.com	621098765	NULL	8
4	32109876T	Miguel	Díaz	Sánchez	m.diaz@autoescuela.com	632109876	NULL	7
5	43210987U	Elena	González	Ruiz	e.gonzalez@autoescuela.com	643210987	NULL	6
6	54321098V	Jorge	Fernández	López	j.fernandez@autoescuela.com	654321098	NULL	5
7	65432109W	Ana	Martínez	García	a.martinez@autoescuela.com	665432109	NULL	4

Activar Windows

Comprobación:



```

498 -- Comprobamos desencriptación
499 DECLARE @FraseSecreta VARCHAR(100) = 'Voy aprobar bases de datos';
500
501 SELECT
502     profesor_DNI_profe AS 'DNI del profesor',
503     CONVERT(VARCHAR(100),
504             DecryptByPassPhrase(@FraseSecreta, profesor_Dir_profe)
505         ) AS DireccionDescifrada
506     FROM [profesor]
507 WHERE profesor_DNI_profe IN ('09876543Q', '10987654R');

```

	DNI del profesor	DireccionDescifrada
1	09876543Q	Calle Gran Vía 78, Madrid
2	10987654R	Avenida Diagonal 123, Barcelona

Encriptación de Backups

\*\*Debemos tener en cuenta que para hacer un backup encriptado debemos hacer todo el proceso desde master y el certificado debe generarse en master también, de lo contrario nos dará un error diciendo que no encuentra el certificado.

Primero crearemos la master key si no la tenemos creada:

```
CREATE MASTER KEY ENCRYPTION BY PASSWORD = 'Abcd1234.';
GO
```

Luego creamos el certificado:

```
CREATE CERTIFICATE [CertificadoBackup]
WITH SUBJECT = 'Certificado encriptar backups';
GO
```

Ahora debemos crear un backup del certificado, además de que lo recomienda mircosoft también es aconsejable puesto que si perdemos el certificado no podremos recuperar el backup que encriptemos.

```
BACKUP CERTIFICATE [CertificadoBackup]
    TO FILE = 'C:\Backup\Certificados\CertificadoBackup.cert'
    WITH PRIVATE KEY (
        FILE =
        'C:\Backup\Certificados\CertificadoBackup.key',
        ENCRYPTION BY PASSWORD =
        'Abcd1234.'
    );
GO
```

\*\*Debemos tener creado el directorio donde le mandamos guardar el backup.

```

515 -- Creamos certificado
516 CREATE CERTIFICATE [CertificadoBackup]
517 | WITH SUBJECT = 'Certificado encriptar backups';
518 GO
519
520 -- Crear backup del certificado (por si se pierde)
521 BACKUP CERTIFICATE [CertificadoBackup]
522     TO FILE = 'C:\Backup\Certificados\CertificadoBackup.cert'
523     WITH PRIVATE KEY (
524         FILE = 'C:\Backup\Certificados\CertificadoBackup.key',
525         ENCRYPTION BY PASSWORD = 'Abcd1234.'
526     );
527 GO
528
529
530
531
532
533

```

100 %

Messages

Commands completed successfully.

Completion time: 2025-03-

Certificados

	Nombre
📁 Música	
📁 Vídeos	
📁 BBDD (\VBo)	
📁 2 PROGRAMACI	
📁 Qwerys	
📁 Certificados	
	└── CertificadoBackup.key
	└── CertificadoBackup.cert

Ahora creamos el backup de la base de datos encriptado:

```

BACKUP DATABASE [AutoescuelaP]
    TO DISK = 'C:\Backup\AutoescuelaP.bak'
    WITH COMPRESSION, ENCRYPTION(ALGORITHM = AES_256,
    SERVER CERTIFICATE = [CertificadoBackup]);
GO

```

```

518 -- Creamos certificado
519 CREATE CERTIFICATE [CertificadoBackup]
520 | WITH SUBJECT = 'Certificado encriptar backups';
521 GO
522
523 -- Crear backup del certificado (por si se pierde)
524 BACKUP CERTIFICATE [CertificadoBackup]
525     TO FILE = 'C:\Backup\Certificados\CertificadoBackup.cert'
526     WITH PRIVATE KEY (
527         FILE = 'C:\Backup\Certificados\CertificadoBackup.key',
528         ENCRYPTION BY PASSWORD = 'Abcd1234.'
529     );
530 GO
531
532 -- Realizamos backup encriptado
533 BACKUP DATABASE [AutoescuelaP]
534     TO DISK = 'C:\Backup\AutoescuelaP.bak'
535     WITH COMPRESSION, ENCRYPTION(ALGORITHM = AES_256,
536     SERVER CERTIFICATE = [CertificadoBackup]);
537 GO

```

100 %

Messages

Processed 16 pages for database 'AutoescuelaP', file 'Lugo' on file 1.  
 Processed 16 pages for database 'AutoescuelaP', file 'Orense' on file 1.  
 Processed 16 pages for database 'AutoescuelaP', file 'Pontevedra' on file 1.  
 Processed 16 pages for database 'AutoescuelaP', file 'Madrid' on file 1.  
 Processed 59 pages for database 'AutoescuelaP', file 'FOTOGRAFIAS' on file 1.  
 Processed 1 pages for database 'AutoescuelaP', file 'autoescuela\_log' on file 1.  
 BACKUP DATABASE successfully processed 1475 pages in 0.825 seconds (13.962 MB/sec).

Ahora abrimos otra máquina virtual, en mi caso he usado una con Windows10 y SSMS20.

Me he conectado con un usuario con rol Sysadmin e identificación SQL Server (De otra manera no me dejaba recuperar el certificado).

### Creamos clave master

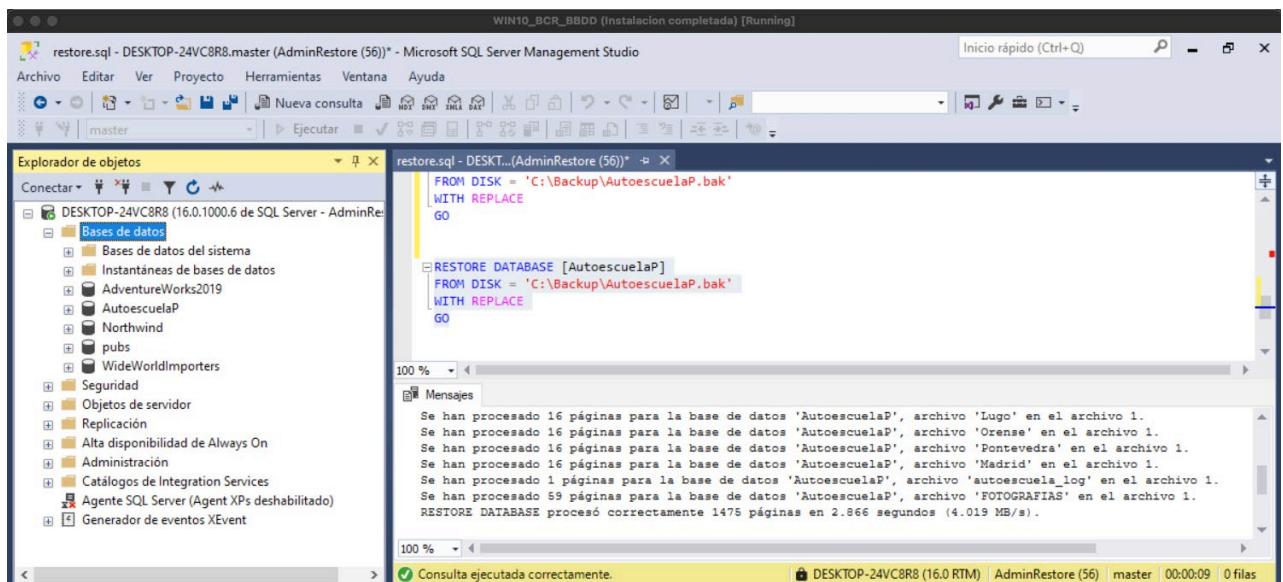
```
CREATE MASTER KEY ENCRYPTION BY PASSWORD = 'Abcd1234.';
GO
```

### Restauramos el certificado:

```
CREATE CERTIFICATE [CertificadoBackup]
FROM FILE = 'C:\Certificados\CertificadoBackup.cert'
WITH PRIVATE KEY (
    FILE = 'C:\Certificados\CertificadoBackup.key',
    DECRYPTION BY PASSWORD = 'Abcd1234.' -- Contraseña del backup del certificado
);
GO
```

### Restauramos el backup de la base de datos:

```
RESTORE DATABASE [AutoescuelaP]
FROM DISK = 'C:\Backup\AutoescuelaP.bak'
WITH REPLACE
GO
```



\*\*Debemos mover también los archivos que use la base de datos (en este caso carpeta DATA) y debemos tener en cuenta que en la nueva máquina tendremos que darle permisos al login para que tenga control total sobre la carpeta donde se va a reescribir.

### Encriptación de TDE (Transparent data encryption)

Cifra los archivos de la base de datos (.mdf, .ndf, .ldf) y los backups (.bak).

### Creamos la master key:

```
CREATE MASTER KEY ENCRYPTION BY PASSWORD = 'Abcd1234.';
GO
```

### Creamos el certificado:

```
CREATE CERTIFICATE [CertificadoTDE]
WITH SUBJECT = 'Certificado encriptar TDE';
GO
```

### Creamos backup del certificado:

```
BACKUP CERTIFICATE [CertificadoTDE]
TO FILE = 'C:\Backup\Certificados\CertificadoTDE.cert'
```

```

WITH PRIVATE KEY (
    'C:\Backup\Certificados\KeyTDE.key',
    FILE =
    'Abcd1234.',
    ) ;
GO

```

Ahora pasamos a usar nuestra base de datos, creamos la clave de encriptación:

```

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

```

Y la activamos:

```

ALTER DATABASE [AutoescuelaP] SET ENCRYPTION ON;
GO

```

Ahora comprobamos el estado:

The screenshot shows a SQL query window with the following code:

```

585 SELECT DB_name(database_id) AS 'Database', encryption_state
586 FROM sys.dm_database_encryption_keys;
587 GO

```

The results pane displays a table with two rows:

Database	encryption_state
tempdb	3
autoescuelaP	3

\*\* 3 significa que TDE se encuentra activado.

Ahora hacemos un backup completo y un backup de log (el de log no es imprescindible)

```

-- Hacemos backup completo
BACKUP DATABASE AutoescuelaP
TO DISK = 'C:\Backup\RecoveryfullWithTDE.bak';
GO

-- Hacemos backup de log
BACKUP LOG AutoescuelaP
TO DISK = 'C:\Backup\RecoveryWithTDE_log.bak';
GO

```

En este caso, vamos a probar a restaurar la base de datos en la otra máquina virtual mediante ATTACH, para ello ejecutamos en nuestra máquina principal para detener los procesos y separar los archivos de la base de datos de la instancia:

```

USE master;
ALTER DATABASE AutoescuelaP SET SINGLE_USER WITH ROLLBACK IMMEDIATE;
EXEC sp_detach_db 'AutoescuelaP';
GO

```

Ahora vamos a la nueva máquina virtual e intentamos restaurar la base de datos realizando un ATTACH pero vemos que nos da error porque falta el certificado:

The screenshot shows a SQL query window with the following code:

```

CREATE DATABASE AutoescuelaP
ON (FILENAME = 'C:\DATA\AutoescuelaP.mdf'),
(Filename = 'C:\DATA\AutoescuelaP_log.ldf')
FOR ATTACH;
GO

```

The results pane displays the following error message:

```

Mens. 33111, Nivel 16, Estado 3, Línea 55
No se encuentra el servidor certificado con la huella digital '0x9151BE704A34C361A08B1B232A4D0ECD65858372'.
Mens. 1802, Nivel 16, Estado 7, Línea 55
Error de CREATE DATABASE. No se pueden crear algunos de los archivos de la lista. Consulte los errores relacionados.

Hora de finalización: 2025-03-19T10:20:55.8398489+01:00

```

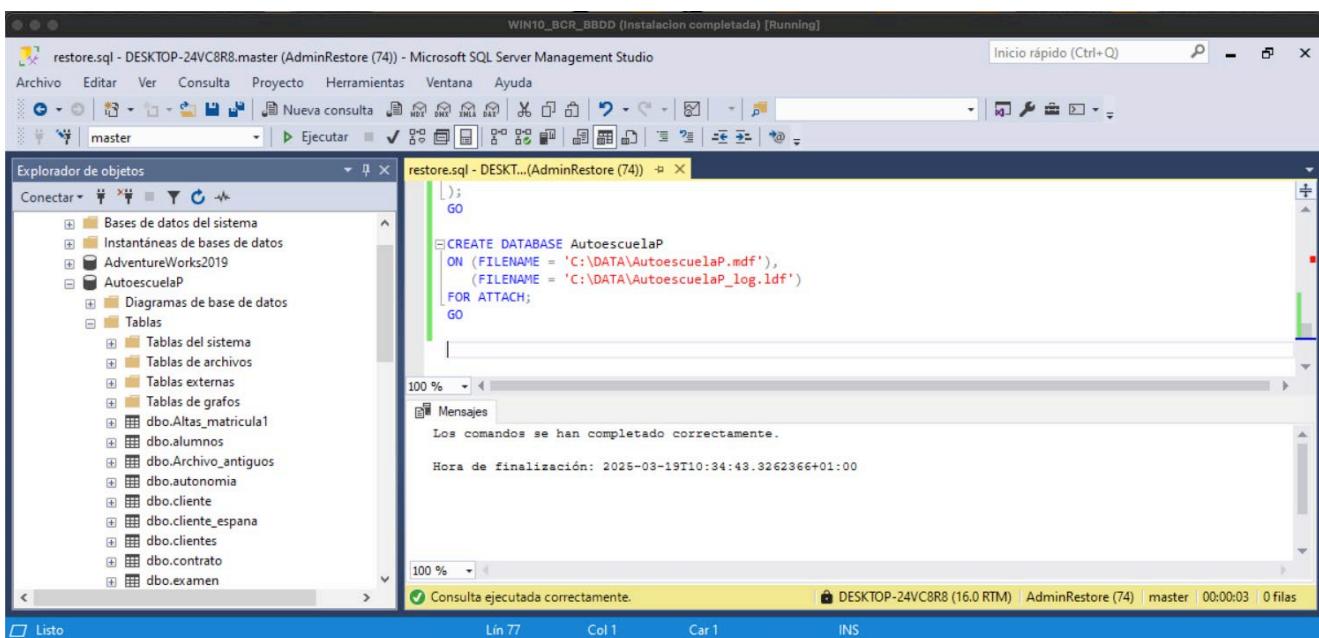
Restauramos el certificado:

```
CREATE CERTIFICATE [CertificadoTDE]
FROM FILE = 'C:\DATA\CertificadoTDE.cert'
WITH PRIVATE KEY (
    FILE = 'C:\DATA\KeyTDE.key',
    DECRYPTION BY PASSWORD = 'Abcd1234.' -- Contraseña del backup del certificado
);
GO
```

Ahora volvemos a intentar relizar el ATTACH:

```
CREATE DATABASE AutoescuelaP
ON (FILENAME = 'C:\DATA\AutoescuelaP.mdf'),
(FILENAME = 'C:\DATA\AutoescuelaP_log.ldf')
FOR ATTACH;
GO
```

La base de datos se crea sin errores:



## Hashing

El hashing es un proceso criptográfico que convierte una contraseña en una cadena de caracteres de longitud fija. Esto se hace usando funciones hash como SHA-2 (SHA-256, SHA-512) o MD5.

El salting es una técnica adicional al hashing. Consiste en agregar una cadena aleatoria (sal) a la contraseña antes de generar el hash. Esto previene ataques como Rainbow Tables.

No puedes recuperar una contraseña almacenada en forma de **hash**, porque el **hashing es un proceso unidireccional** (no reversible). Esto significa que, una vez que una contraseña se convierte en un hash, no se puede convertir de nuevo en la contraseña original.

Aunque el hashing se suele usar para contraseñas, en este caso voy a aplicarlo a mi base de datos para ocultar los números de cuenta de los trabajadores cada vez que se inserte una nómina en la base de datos.

Para ello, vamos a seguir el ejemplo dado en clase y vamos a generar un trigger que salte cada vez que se intente insertar un registro en la tabla nominas, de esta manera el trigger

cogerá los valores que se quieren insertar, seleccionará el número de cuenta, le aplicará el hash, luego se reforzará la seguridad aplicando salting y por último registrará el dato en la tabla.

Generamos el trigger:

```
CREATE OR ALTER TRIGGER insertar_nomina_hash
ON Nomina
INSTEAD OF INSERT
AS
DECLARE
    @salt VARCHAR(20), @hash VARCHAR(64),
    @cuenta VARCHAR(30), @hashedcuenta VARCHAR(80),
    @Id_Nomina VARCHAR(12), @Department VARCHAR(12),
    @Nivel VARCHAR(12), @H_extra VARCHAR(12),
    @contrato_ID_contrato INT;
    SELECT @Id_Nomina = ins.Id_Nomina,
           @Department = ins.Department,
           @Nivel = ins.Nivel,
           @H_extra = ins.H_extra,
           @cuenta = ins.N_Cuenta,
           @contrato_ID_contrato = ins.contrato_ID_contrato
    FROM INSERTED ins;
BEGIN
    SET @cuenta = (SELECT Id_Nomina FROM inserted)
    SELECT @salt = CONVERT(VARCHAR(20), CRYPT_GEN_RANDOM(8), 2)
    SET @hash = CONVERT(VARCHAR(64), HashBytes('SHA2_256', (@salt +
@cuenta)), 2)
    SET @hashedcuenta = @salt + @hash;
    INSERT INTO Nomina ([Id_Nomina], [Department], [Nivel], [H_extra],
    [N_Cuenta], [contrato_ID_contrato])
    VALUES (@Id_Nomina, @Department, @Nivel, @H_extra, @hashedcuenta,
    @contrato_ID_contrato);
END;
GO
```

Ahora vamos a intentar insertar una nómina en la tabla:

```
INSERT INTO [dbo].[Nomina]
([Id_Nomina], [Department], [Nivel], [H_extra], [N_Cuenta],
[contrato_ID_contrato])
VALUES
('NOM011', 'IT', 'Senior', '5', 'ES1234567890123456789012', 3);
GO
```

Comprobamos y vemos que el número de cuenta aparece encriptado:



The screenshot shows the SQL Server Management Studio interface. On the left, there's a vertical toolbar with icons for new query, open file, save, and other database operations. Below it, the status bar shows '100 %'. The main area has two tabs: 'Results' (selected) and 'Messages'. In the 'Results' tab, the output of the query 'SELECT \* FROM Nomina;' is displayed. The results table has columns: Id\_Nomina, Department, Nivel, H\_extra, N\_Cuenta, and contrato\_ID\_contrato. The N\_Cuenta column contains the original value ('ES1234567890123456789012') and the contrato\_ID\_contrato column contains the value '3'. The other columns show standard data like department names and levels. The 'Messages' tab is empty.

	Id_Nomina	Department	Nivel	H_extra	N_Cuenta	contrato_ID_contrato
5	NOM005	Enseñanza	Junior	6	ES6621000418401234567895	5
6	NOM006	Enseñanza	Medio	7	ES1720852066201234567896	6
7	NOM007	Enseñanza	Junior	4	ES0182038454701234567897	7
8	NOM008	Enseñanza	Senior	15	ES7100302053091234567898	8
9	NOM009	Enseñanza	Medio	9	ES1000492352081234567899	9
10	NOM010	Enseñanza	Junior	5	ES0049295427011234567800	10
11	NOM011	IT	Senior	5	331026B03D889DCFA25A2AD7EC54C00E8A927E6D1EF8B0...	3

Aunque el hashing no se puede revertir, si que podemos aplicar el hash a una contraseña y saber si coinciden el hash nuevo con el antiguo y eso nos sirve para comprobar si la

contraseña introducida coincide con la guardada, ya que el hash da siempre el mismo resultado si se le aplica a lo mismo y el salt es una cadena de caracteres añadida.

Vamos a comprobarlo, aplicándolo a la cuenta a la que le hemos aplicado el salt y el hash en la tabla Nómina, de esta manera al ingresar la nomina podemos no saber que cuenta tenemos almacenada pero si podemos saber si en la que vamos hacer el ingreso es la correcta.

Generamos la función:

```
CREATE OR ALTER FUNCTION cuentavalida (
    @Id_Nomina varchar(20), @nuevacuenta varchar(30)
)
RETURNS bit
AS

BEGIN
    DECLARE
        @isValid bit, @cuentalmacenada varchar(80), @nuevacuentahash varchar(80),
        @salt varchar(16), @hash varchar(64)
    SET @cuentalmacenada = (SELECT [N_Cuenta]
                                FROM Nomina
                                WHERE
                                Id_Nomina = @Id_Nomina)
        IF (@cuentalmacenada IS NULL) RETURN 0
        SET @salt = SUBSTRING(@cuentalmacenada, 1, 16)
        SET @hash = CONVERT(VARCHAR(64), HashBytes('SHA2_256', (@salt +
@nuevacuenta)),2)
        SET @nuevacuentahash = @salt + @hash;

        IF (@cuentalmacenada != @nuevacuentahash)
            SET @isValid = 0
        ELSE
            SET @isValid = 1
        RETURN @isValid
END;
GO
```

No consigo detectar el error, aunque la función no me da error, nunca me detecta el valor correcto:

The screenshot shows a SQL query window with the following content:

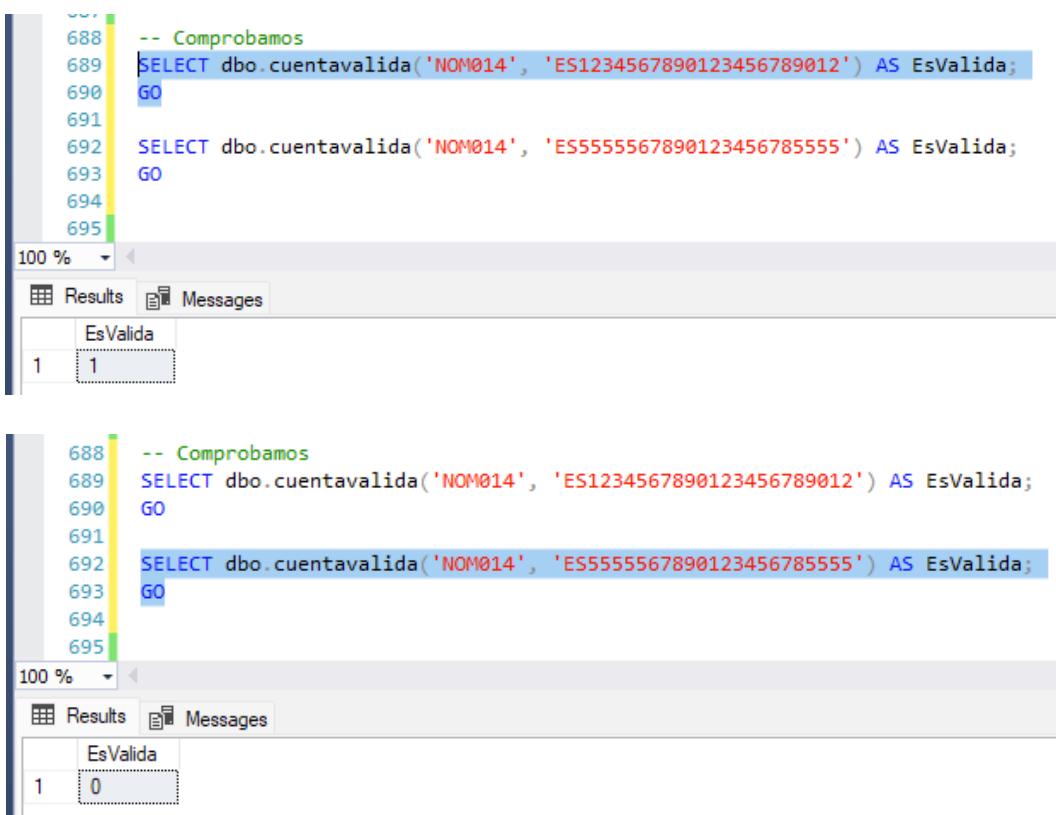
```
691 -- Comprobamos
692 SELECT dbo.cuentavalida('NOM012', 'ES1234567890123456789012') AS EsValida;
693 GO
694
```

The results pane shows a single row with the column name "EsValida" and the value "0".

EsValida
0

Detectado el error:

El error estaba en el trigger ya que en vez de estar tomando el nº de cuenta para aplicar el hash, estaba utilizando el nº de nómina. Al cambiar la variable ya funcionó todo correctamente.



```

688 -- Comprobamos
689 SELECT dbo.cuentavalida('NOM014', 'ES1234567890123456789012') AS EsValida;
690 GO
691
692 SELECT dbo.cuentavalida('NOM014', 'ES5555567890123456785555') AS EsValida;
693 GO
694
695
100 %
Results Messages
EsValida
1 1

```

```

688 -- Comprobamos
689 SELECT dbo.cuentavalida('NOM014', 'ES1234567890123456789012') AS EsValida;
690 GO
691
692 SELECT dbo.cuentavalida('NOM014', 'ES5555567890123456785555') AS EsValida;
693 GO
694
695
100 %
Results Messages
EsValida
1 0

```

## DDM Dynamic Data Masking

Dynamic Data Masking (DDM), que es una característica de seguridad diseñada para limitar la exposición de datos sensibles a usuarios sin privilegios.

Características principales en SQL Server :

Protección a nivel de columna: Puedes aplicar máscaras a columnas específicas que contienen información sensible

Múltiples funciones de enmascaramiento:

- Máscara predeterminada: Muestra XXXX para tipos de datos de cadena, 0 para tipos numéricos y 01.01.1900 para fecha/hora
- Máscara de correo electrónico: Expone solo la primera letra y el dominio (ej., a\*\*\*\*\*@ejemplo.com)
- Máscara parcial: Muestra los primeros y últimos caracteres (ej., ABCD1234 → AB\*\*\*\*34)
- Máscara de cadena personalizada: Permite patrones de reemplazo personalizados

Acceso basado en privilegios: Los usuarios con los permisos adecuados (p.ej., permiso UNMASK) pueden ver los datos originales

Creamos una tabla con campos enmascarados:

```
DROP TABLE IF EXISTS aprobados
GO
CREATE TABLE aprobados(
    ID INT IDENTITY PRIMARY KEY,
    Nombre NVARCHAR(15) MASKED WITH (FUNCTION = 'default()') NULL,
    Apellidos NVARCHAR(30) MASKED WITH (FUNCTION = 'default()') NULL,
    Email NVARCHAR(18) MASKED WITH (FUNCTION = 'email()') NULL,
    N_cuenta NVARCHAR (25) MASKED WITH (FUNCTION = 'partial(0,"XXXX-XXXX-XXXX-",4)') NULL
);
GO
```

Ahora insertamos valores:

```
INSERT INTO aprobados (Nombre, Apellidos, Email, N_Cuenta)
VALUES ('Borja', 'Costa Rojo', 'cjgalego@gmail.com', '2100-2938-7765-9876-2345');
GO
```

Si realizamos una consulta con un usuario con permisos, se verán todos los datos pero si realizamos una consulta con un usuario sin permisos, solo con permiso de consulta, se muestra la información enmascarada de la siguiente manera:

```
767 GRANT SELECT ON aprobados TO profel;
768 GO
769
770 EXECUTE AS USER = 'profel';
771 SELECT * FROM aprobados;
772 GO
773
774 REVERT
775 GO
```

	ID	Nombre	Apellidos	Email	N_cuenta
1	1	xxxx	xxxx	cXXX@XXXX.com	XXXX-XXXX-XXXX-2345

Podemos hacer que un usuario vea los campos concediendo permisos UNMASK:

```
GRANT UNMASK ON aprobados TO profel;
GO
777 -- Probamos a dar permiso UNMASK al usuario profel
778 GRANT UNMASK ON aprobados TO profel;
779 GO
780 --Comprobamos
781 EXECUTE AS USER = 'profel';
782 SELECT * FROM aprobados;
783 GO
```

	ID	Nombre	Apellidos	Email	N_cuenta
1	1	Borja	Costa Rojo	cjgalego@gmail.com	2100-2938-7765-9876-2345

Ahora vamos a probar a desenmascarar solo un campo, concedemos permisos a profe2 y desenmascaramos el correo:

```
GRANT SELECT ON aprobados TO profe2;
GO
ALTER TABLE aprobados
    ALTER COLUMN email DROP MASKED;
GO
```

The screenshot shows a SQL query window with the following code:

```

794 EXECUTE AS USER = 'profes2';
795 GO
796 SELECT * FROM aprobados;
797 GO
798
799
  
```

The results grid below shows one row of data:

ID	Nombre	Apellidos	Email	N_cuenta
1	xxxx	xxxx	cjgalego@gmail.com	XXXX-XXXX-XXXX-2345

## Secuencias

Las secuencias son objetos de base de datos que generan una **secuencia numérica ordenada**, independiente de las tablas. Fueron introducidas en SQL Server 2012 y ofrecen mayor flexibilidad que las columnas de identidad. La principal diferencia con el IDENTITY es que la secuencia es un objeto independiente de la propia base de datos y podemos trabajar con la secuencia en diferentes tablas.

Vamos a suponer que queremos que a medida que registremos clientes en nuestra autoescuela, se vayan distribuyendo en aulas, en nuestra autoescuela disponemos de 3 aulas y en cada aula caben 5 alumnos. Usando una secuencia que se reinicie cada 15 alumnos (cuando se completen todas las aulas) podemos distribuir a los alumnos y calcularemos el aula que le corresponde dividiendo entre las 3 aulas. Vamos a probarlo:

```

-- Creamos la secuencia
CREATE SEQUENCE dbo.ControlAulas
START WITH 1
INCREMENT BY 1
MINVALUE 1
MAXVALUE 15
CYCLE;

-- Actualizamos el aula automáticamente usando un valor calculado
ALTER TABLE dbo.clientes
ADD CONSTRAINT DF_Aula DEFAULT
((NEXT VALUE FOR dbo.ControlAulas - 1)/5 % 3 + 1) FOR aula;
GO
  
```

Ahora introducimos 20 clientes y vemos si funciona:

```

809 -- Creamos una secuencia que se reinicie cada 15 (5 alumnos x 3 aulas)
810 CREATE SEQUENCE dbo.ControlAulas
811 START WITH 1
812 INCREMENT BY 1
813 MINVALUE 1
814 MAXVALUE 15
815 CYCLE;
816
817 -- Actualizamos el aula automáticamente usando un valor calculado
818 ALTER TABLE dbo.clientes
819 ADD CONSTRAINT DF_Aula DEFAULT
820 ((NEXT VALUE FOR dbo.ControlAulas - 1)/5 % 3 + 1) FOR aula;
821 GO
822
823 -- Comprobamos
824
825
826 INSERT INTO clientes (DNI_cliente, N_cliente, Apel1_cliente, Apel2_cliente, Dir_cliente, Email_cliente, Telf_cliente, localidad_cod_localidad)...
827 GO

```

Results Messages

DNI_cliente	N_cliente	Apel1_cliente	Apel2_cliente	Dir_cliente	Email_cliente	Telf_cliente	localidad_cod_localidad	aula	
7	45879632R	Sofía	Rodríguez	NULL	Plaza de María Pita 9	sofia.rodriguez@coruna.es	655998877	1	1
8	32659874W	David	Sánchez	Castro	Calle de San Andrés 22	david.sanchez@coruna.es	699332211	1	2
9	11223344A	Elena	Fernández	Vázquez	Avenida de Finisterre 100	elena.femandez@coruna.es	688776655	1	2
10	99887766B	Javier	Romero	NULL	Rúa Panaderas 15	javier.romero@coruna.es	612345678	1	2
11	66554433C	Lucía	García	Iglesias	Calle de la Estrella 4	lucia.garcia@coruna.es	687654321	1	2
12	22334455D	Pedro	Torres	Molina	Praza de Pontevedra 1	pedro.tomes@coruna.es	600112233	1	2
13	77889900E	Marta	Navarro	Ortega	Calle de San Juan 30	marta.navarro@coruna.es	699887766	1	3
14	44556677F	Daniel	Herrera	NULL	Avenida de Oza 250	daniel.herrera@coruna.es	633445566	1	3
15	33445566G	Paula	Dominguez	Castro	Rúa de la Franja 8	paula.dominguez@coruna...	677889900	1	3
16	12345678Z	Adrián	Vázquez	Lorenzo	Calle de la Barrera 12	adrian.vazquez@coruna.es	688990011	1	3
17	87654321X	Raquel	Castro	Santos	Avenida de los Cantones 5	raquel.castro@coruna.es	622334455	1	3
18	19283746S	Sergio	Ortega	Díaz	Rúa de la Galera 17	sergio.ortega@coruna.es	655443322	1	1
19	56473829V	Carmen	Molina	NULL	Calle de la Torre 55	carmen.molina@coruna.es	699554433	1	1
20	65748392N	Alejandro	Iglesias	Gómez	Praza de España 2	alejandro.iglesias@coruna...	611223344	1	1

\*\*Si tuviésemos los clientes distribuidos en varias tablas, podríamos usar igualmente la secuencia puesto que funciona en todas las tablas de la base de datos.

Activar Windows

Podríamos por ejemplo usar esta secuencia y este supuesto para que cada 5 clientes, la secuencia diese un salto, de esta forma dejaríamos siempre un hueco vacío en cada aula para posibles incorporaciones de urgencia. Bastaría que cuando sequence valiese 5 se ejecutase un NEXT VALUE FOR [dbo].[ControlAulas].

## ROW LEVEL SECURITY (RLS)

**Row-Level Security (RLS)** en SQL Server es una funcionalidad de seguridad que permite controlar el acceso a **filas específicas** de una tabla en función de las características del usuario que ejecuta la consulta. Su objetivo es garantizar que los usuarios solo puedan ver o modificar los datos relevantes para su contexto, sin necesidad de modificar la lógica de la aplicación.

**Predicados de filtro:** Limitan las filas que se pueden leer (SELECT)

**Predicados de bloqueo:** Limitan las filas donde se pueden realizar operaciones de modificación (INSERT, UPDATE, DELETE)

Vamos a intentar implementar un Row Level Security para que cada profesor de práctica solo tenga acceso a las prácticas de sus alumnos y no al resto de prácticas de los demás alumnos

Para ello comenzamos creando la función de seguridad:

```

CREATE FUNCTION dbo.fn_seguridad_profesor (@DNI_profe AS SYSNAME)
RETURNS TABLE
WITH SCHEMABINDING
AS
RETURN (
    SELECT 1 AS acceso
    WHERE @DNI_profe = USER_NAME()
    OR IS_ROLEMEMBER('db_owner') = 1
);
GO

```

Ahora creamos la política de seguridad:

```

CREATE SECURITY POLICY PoliticaSeguridadPractica
ADD FILTER PREDICATE dbo.fn_seguridad_profesor(profe_practicas_profesor_DNI_profe)
ON dbo.practica
WITH (STATE = ON);
GO

```

Ahora creamos un usuario para un profesor y hacemos una consulta impersonandolo:

```

876 -- Comprobamos
877 -- Creamos usuario de profesor
878 CREATE USER [09876543Q] WITHOUT LOGIN;
879 GRANT SELECT ON dbo.practica TO [09876543Q];
880 GO
881
882 -- Test acceso profesor
883 EXECUTE AS USER = '09876543Q';
884 SELECT * FROM practica; -- Solo verá 3 registros (los de este profesor)
885 REVERT;

```

The screenshot shows the SQL code above being run in SSMS. Below the code, the 'Results' tab displays a table with four rows of data from the 'practica' table, all belonging to the professor with DNI 09876543Q.

ID_practica	F_practica	Estado_practica	cliente_DNI_cliente	vehiculo_vehiculo_ID	profe_practicas_profesor_DNI_profe
1	5	Pendiente	01234567J	5	09876543Q
2	11	Aprobada	01234567J	1	09876543Q
3	12	En curso	10111213E	2	09876543Q

Como podemos ver, solo se muestran las prácticas realizadas por el propio profesor. Sin embargo, si hacemos la misma consulta con un usuario con todos los permisos, observamos que se muestran todos los datos de todos los clientes:

```

887 SELECT * FROM practica;
888 GO
889
890
891

```

The screenshot shows the SQL code above being run in SSMS. Below the code, the 'Results' tab displays a table with 14 rows of data from the 'practica' table, representing all clients.

ID_practica	F_practica	Estado_practica	cliente_DNI_cliente	vehiculo_vehiculo_ID	profe_practicas_profesor_DNI_profe
1	1	Completada	67890123F	1	87654321Y
2	2	Pendiente	78901234G	2	65432109W
3	3	En curso	89012345H	3	43210987U
4	4	Completada	90123456I	4	21098765S
5	5	Pendiente	01234567J	5	09876543Q
6	11	Aprobada	01234567J	1	09876543Q
7	12	En curso	10111213E	2	09876543Q
8	13	Reprobada	11121314R	3	09876543Q
9	14	Aprobada	11223344J	4	21098765S
10	15	En curso	12131415E	5	21098765S
11	16	Reprobada	12345618A	1	21098765S
12	17	Aprobada	12345678A	2	43210987U
13	18	En curso	13141516W	3	43210987U
14	19	Reprobada	14151617Q	4	43210987U

Para entenderlo mejor, se podría decir que la función de seguridad es un listado de reglas sobre quien puede o no puede acceder a ciertos datos y la política de seguridad es la que nos dice a que aplicamos la función de seguridad y en que casos.

\*\*Debemos tener en cuenta que si solo usamos FILTER PREDICATE, los usuarios que no puedan ver el contenido si que podrán editarlo, eso quiere decir que si un usuario conoce algún dato de la tabla podría modificarlo o borrarlo. Para evitar este tipo de fallos siempre se debe usar FILTER PREDICATE combinado con BLOCK PREDICATE.

Otro ejemplo de RLS

Vamos a suponer que tenemos una tabla trabajadores con sus departamentos y sus usuarios, queremos que cada usuario solo tenga permisos para consultar y editar datos de su propio departamento:

```
CREATE TABLE trabajadores (
    id INT IDENTITY(1,1) PRIMARY KEY,
    nombre VARCHAR(50) NOT NULL,
    apellidos VARCHAR(100) NOT NULL,
    telefono VARCHAR(15) NOT NULL,
    sueldo DECIMAL(10,2) NOT NULL,
    puesto VARCHAR(50) NOT NULL,
    usuario VARCHAR(50) NOT NULL,
    departamento VARCHAR(50) CHECK (departamento IN ('Recursos humanos',
    'Logística', 'Administración')));
GO
```

Creamos la función de seguridad:

```
CREATE FUNCTION dbo.fn_FiltroPorDepartamento(@departamento VARCHAR(50))
RETURNS TABLE
WITH SCHEMABINDING
AS
RETURN (
    SELECT 1 AS acceso
    FROM dbo.trabajadores
    WHERE departamento = @departamento AND usuario = USER_NAME()
);
GO
```

Creamos la política de seguridad:

```
CREATE SECURITY POLICY Trabajadores_Policy
ADD FILTER PREDICATE dbo.fn_FiltroPorDepartamento(departamento) ON
dbo.trabajadores,
ADD BLOCK PREDICATE dbo.fn_FiltroPorDepartamento(departamento) ON dbo.trabajadores
WITH (STATE = ON);
GO
```

Y ahora realizamos pruebas, el usuario de recursos humanos puede ver y editar datos de trabajadores de su departamento, incluso añadir trabajadores si son de su propio departamento pero no podrá ver ni editar nada de otros departamentos:

The screenshot shows a SQL Server Management Studio interface. In the top-left pane, there are numbered lines of code from 973 to 977. Lines 973 and 974 show the execution of a user-defined function. Line 975 shows a GO command. Lines 976 and 977 show a SELECT statement that retrieves all columns from the 'trabajadores' table. Below this is a 'Results' tab showing the output of the query. The results grid has columns: id, nombre, apellidos, telefono, sueldo, puesto, departamento, and usuario. The data is as follows:

	id	nombre	apellidos	telefono	sueldo	puesto	departamento	usuario
1	1	Marta	González López	681234567	2450.00	Analista de RRHH	Recursos humanos	mgonzalez
2	2	Carlos	Martínez Silva	689876543	2750.00	Reclutador técnico	Recursos humanos	cmartinez
3	3	Ana	Díaz Femández	666112233	2950.00	Jefa de personal	Recursos humanos	adiaz
4	10	Iria	Méndez Sande	600112233	2650.00	Especialista en nóminas	Recursos humanos	imendez
5	14	Carla	Mosquera Pardo	688990011	2850.00	Responsable de formación	Recursos humanos	cmosquera
6	17	Juan	Pérez García	600123456	2500.00	Analista	Recursos humanos	jperez

Ahora probamos a insertar un trabajador de otro departamento y nos da error:

```

979  INSERT INTO trabajadores (nombre, apellidos, telefono, sueldo, puesto, departamento, usuario)
980  VALUES ('Juan', 'Pérez García', '600123456', 2500.00, 'Analista', 'Logistica', 'jperez');
981  GO
982
100 %

```

Messages

Msg 33504, Level 16, State 1, Line 978  
The attempted operation failed because the target object 'AutoescuelaP.dbo.trabajadores' has a block predicate that conflicts with this operation.  
The statement has been terminated.

Completion time: 2025-03-20T10:58:19.1621952+01:00

Sin embargo si la inserción es de su propio departamento si nos deja:

```

983  INSERT INTO trabajadores (nombre, apellidos, telefono, sueldo, puesto, departamento, usuario)
984  VALUES ('Juan', 'Pérez García', '600123456', 2500.00, 'Analista', 'Recursos humanos', 'jperez');
985  GO
100 %

```

Messages

(1 row affected)

Completion time: 2025-03-20T10:59:55.6354235+01:00

## Always Encrypted

Encriptar la circulación de datos entre un cliente y el servidor.

### Column Encryption Key (CEK)

La **Column Encryption Key (CEK)** es una clave utilizada para cifrar y descifrar los valores de las columnas de datos. Esta clave es específica de cada columna que deseas cifrar.

### Column Master Key (CMK)

La **Column Master Key (CMK)** es una clave maestra que se utiliza para cifrar las Column Encryption Keys (CEK). Es la clave de más alto nivel y se asegura de proteger las CEK de manera segura.

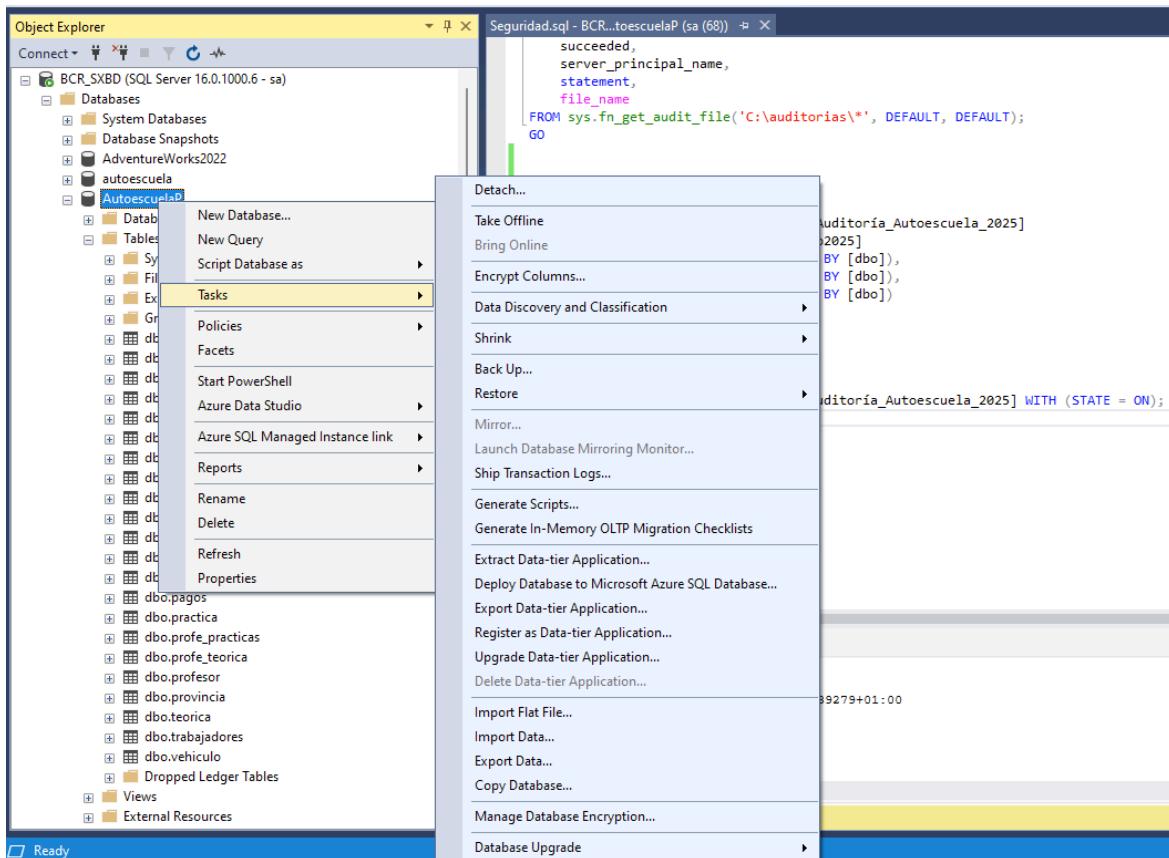
### Cifrado Determinístico (Deterministic Encryption)

El **cifrado determinístico** genera el mismo valor cifrado para una entrada de datos idéntica cada vez que se cifra. Esto significa que si tienes dos filas con el mismo valor en una columna, su valor cifrado será el mismo, sin importar cuántas veces se cifre.

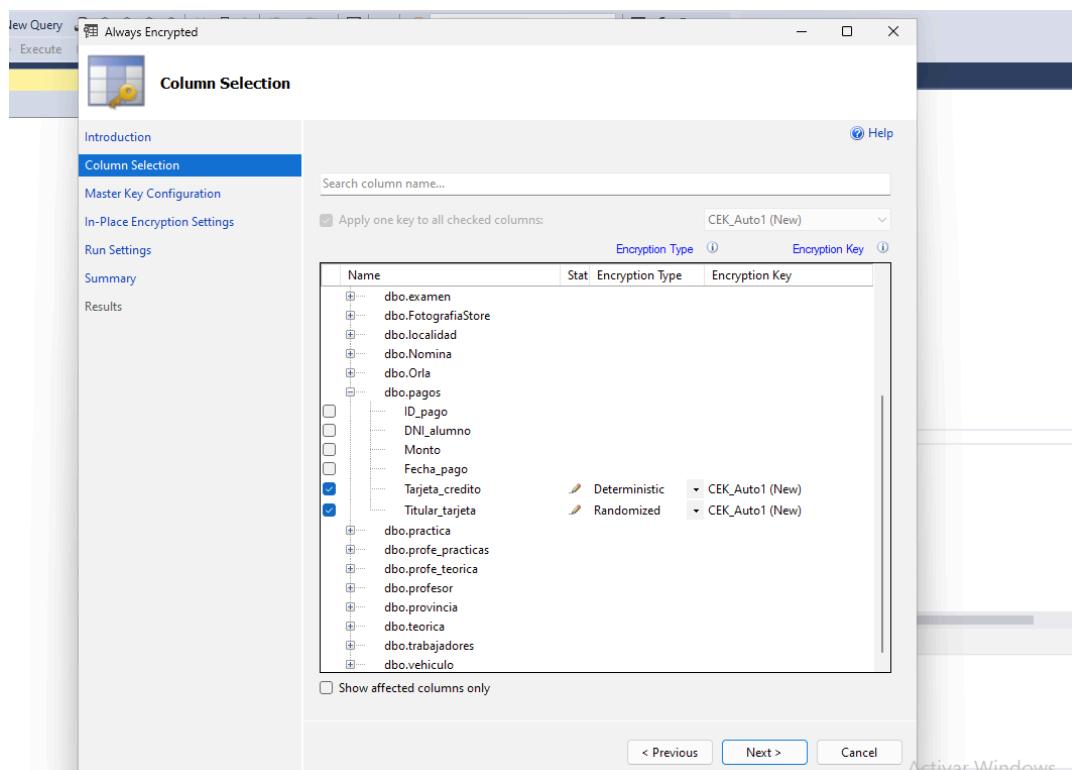
### Cifrado Aleatorio (Randomized Encryption)

El **cifrado aleatorio** genera un valor cifrado diferente cada vez que se cifra el mismo dato, incluso si los valores de entrada son iguales. Esto significa que, aunque los datos originales sean idénticos, su representación cifrada será diferente en cada operación de cifrado.

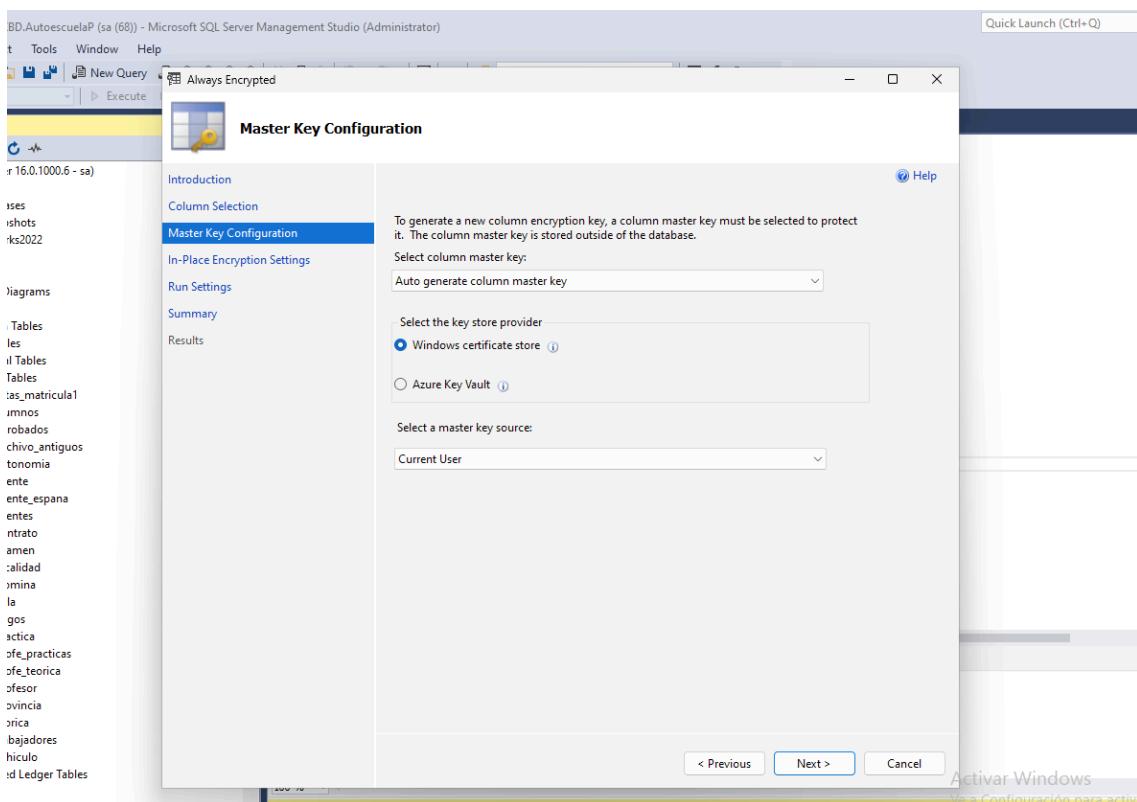
Vamos a la barra lateral y pinchamos sobre nuestra base de datos con el botón derecho, luego Task y luego “Encrypt Columns...”.



Ahora seleccionamos las columnas que queremos encriptar, en este caso vamos a encriptar los datos de las tarjetas de crédito usadas en los pagos, podemos escoger entre cifrado deterministico o random:



La siguiente pantalla la dejamos tal y como está, para que se almacene en windows (podríamos almacenar en Azure también).



Una vez termine de realizar las operaciones, realizamos la consulta y comprobamos que los datos están encriptados

```
SELECT * FROM pagos;
GO
```

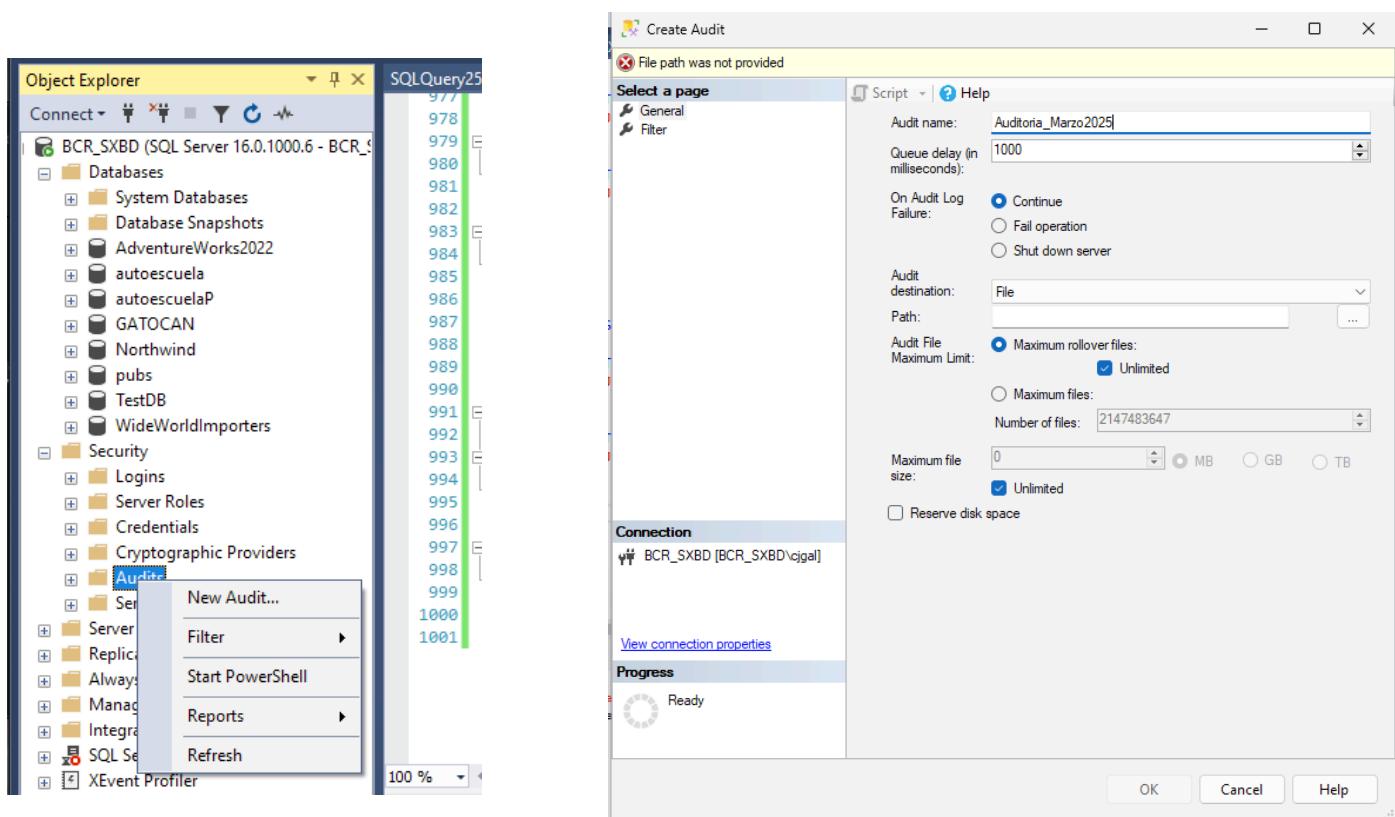
ID_pago	DNI_alumno	Monto	Fecha_pago	Tarjeta_credito	Titular_tarjeta
1	12345678A	250.50	2025-03-20	0x0177A53E011AEDC68701A9A66D59DFA8A69FAD8D180B18...	0x01E11D1FA974FBE76EB7C5424B66A9B7F0452723EB09356...
2	12345678A	250.50	2025-03-20	0x01761DD9A1124664ED4C8D877E9862DB51C1D8564B4563...	0x0116F5D481A423387F152B494966509C6E6A9E85C7706904...
3	12345678A	250.50	2025-03-20	0x01DE043B6EBA6699B294BAC7977F8F28364E4C6B8DBF5A0...	0x0165050C8271F2534D36194EAD2A9DE4F44CD565F3662E4...
4	55443322L	300.00	2025-03-21	0x01D56D3111A40DBA5902B2EB70C6002EB6EE1D4277C683...	0x01FA5D680EA4CCF0E90086590C4062CE8311FD1064DD0F...

## 2.- Auditorías

Las auditorías en SQL Server permiten registrar eventos y actividades dentro del servidor de base de datos. Son útiles para monitorear seguridad, cumplimiento normativo y cambios en los datos.

### Auditoría de servidor

Desde el interfaz gráfico:



- **Application log:** La captura se envía al visor de sucesos Applications
- **Security log:** La captura va a visor de sucesos Security
- **File:** La captura va a un fichero

Desde el script

Siempre deben realizarse desde la base de datos master.

Primero realizamos la auditoría al appLog:

```
USE MASTER
GO

CREATE SERVER AUDIT [Auditoria_marzo2025]
    TO application_log
WITH
    (queue_delay = 1000,
    on_failure = fail_operation
    )
GO
```

The screenshot shows the Object Explorer on the left with the connection to 'BCR\_SXBD'. In the 'Audits' node under 'Security', 'App\_Auditoria\_marzo2025' is selected. The main pane displays the T-SQL script for creating the audit:

```

1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1010
1011
1012
1013
1014 -- Auditorías
1015 PRINT USER
1016 GO
1017
1018 USE MASTER
1019 GO
1020
1021 CREATE SERVER AUDIT [App_Auditoria_marzo2025]
1022 TO application_log
1023 WITH
1024 (queue_delay = 1000,
1025 on_failure = fail_operation
1026 )
1027 GO
1028
1029

```

The 'Messages' pane at the bottom shows the command completed successfully.

Ahora la auditoría al Security log:

```

CREATE SERVER AUDIT [Security_Auditoria_marzo2025]
    TO security_log
WITH
    (queue_delay = 1000,
    on_failure = fail_operation
    )
GO

```

The screenshot shows the Object Explorer on the left with the connection to 'BCR\_SXBD'. In the 'Audits' node under 'Security', both 'App\_Auditoria\_marzo2025' and 'Security\_Auditoria\_marzo2020' are selected. The main pane displays the T-SQL script for creating the audits:

```

1008
1009
1010
1011
1012
1013
1014 -- Auditorías
1015 PRINT USER
1016 GO
1017
1018 USE MASTER
1019 GO
1020
1021 CREATE SERVER AUDIT [App_Auditoria_marzo2025]
1022 TO application_log
1023 WITH
1024 (queue_delay = 1000,
1025 on_failure = fail_operation
1026 )
1027 GO
1028
1029
1030 CREATE SERVER AUDIT [Security_Auditoria_marzo2025]
1031 TO security_log
1032 WITH
1033 (queue_delay = 1000,
1034 on_failure = fail_operation
1035 )
1036 GO
1037

```

The 'Messages' pane at the bottom shows the commands completed successfully.

Ahora vamos a realizar la auditoría a archivo:

Primero creamos la carpeta donde se guardará:

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

```
EXEC sp_configure 'xp_cmdshell', 1;
RECONFIGURE;

EXEC sp_configure 'show advanced options', 1;
RECONFIGURE;

EXEC xp_cmdshell 'mkdir C:\auditorias';
```

Ahora generamos la auditoría:

```
CREATE SERVER AUDIT [File_Auditoria_marzo20225]
    TO FILE
(filepath = 'c:\auditorias\' ,
maxsize = 0mb,
max_rollover_files = 2147483647,
reserve_disk_space = off
)
WITH(
    queue_delay = 1000,
    on_failure = continue
)
GO
```

```
1035      )
1036      GO
1037
1038      -- Creamos carpeta para guardar la auditoría
1039      EXEC sp_configure 'show advanced options', 1;
1040      RECONFIGURE;
1041
1042      EXEC sp_configure 'xp_cmdshell', 1;
1043      RECONFIGURE;
1044
1045      EXEC sp_configure 'show advanced options', 1;
1046      RECONFIGURE;
1047
1048      EXEC xp_cmdshell 'mkdir C:\auditorias';
1049
1050      -- Creamos auditoría a archivo
1051      CREATE SERVER AUDIT [File_Auditoria_marzo20225]
1052          TO FILE
1053              (filepath = 'c:\auditorias\' ,
1054                  maxsize = 0mb,
1055                  max_rollover_files = 2147483647,
1056                  reserve_disk_space = off
1057
1058      WITH(
1059          queue_delay = 1000,
1060          on_failure = continue
1061
1062      GO
1063
```

Ahora tendríamos las tres auditorías creadas pero no estarían activas, para activarlas podemos hacerlo mediante el entorno gráfico, pulsando con el botón derecho sobre la auditoría o bien mediante el script:

```
ALTER SERVER AUDIT File_Auditoria_marzo20225 WITH (STATE = ON)
```

GO

Desaparece la X roja de la auditoría en la barra lateral.

Ahora nos queda darle las indicaciones de lo que queremos que audite, vamos a probar a hacerlo desde el script ya que desde el entorno gráfico es sencillo mediante botón derecho sobre la auditoría. Vamos a pedirle que registre inicios de sesión fallidos, inicios de sesión exitosos y cambios de contraseña.

```
CREATE SERVER AUDIT SPECIFICATION [Specificaciones_Audit_File]
FOR SERVER AUDIT [File_Auditoria_marzo20225]
    ADD (FAILED_LOGIN_GROUP),
    ADD (SUCCESSFUL_LOGIN_GROUP),
    ADD (USER_CHANGE_PASSWORD_GROUP)
```

GO

Ahora haremos varias operaciones como cambiar de usuario, cambiar contraseñas...etc y comprobamos que se registrasen los cambios en la auditoría:

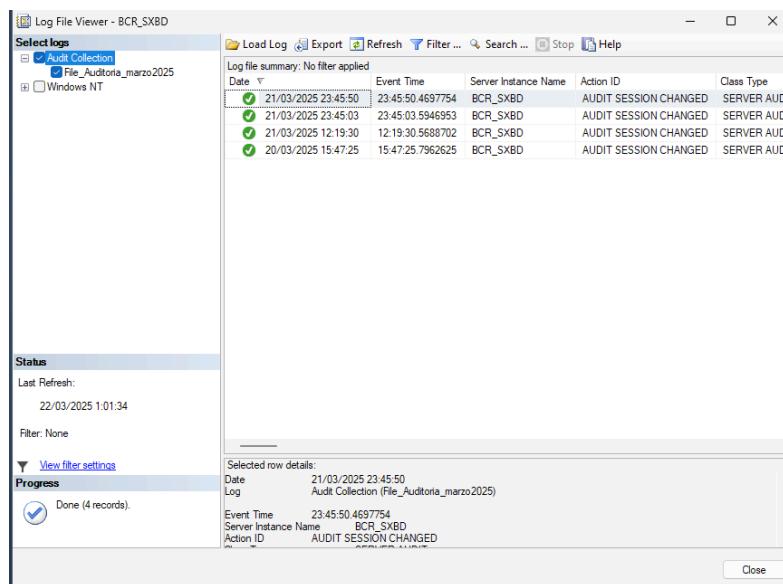
The screenshot shows the SSMS interface with the following details:

- Object Explorer:** Shows the database structure including 'BCR\_SXBD' and its objects like 'Databases', 'Security', and 'Audits'.
- Query Window:** Contains T-SQL code for creating a server audit specification and adding audit groups:
 

```
CREATE SERVER AUDIT SPECIFICATION [Specificaciones_Audit_File]
FOR SERVER AUDIT [File_Auditoria_marzo20225]
    ADD (FAILED_LOGIN_GROUP),
    ADD (SUCCESSFUL_LOGIN_GROUP),
    ADD (USER_CHANGE_PASSWORD_GROUP)
```
- Log File Viewer:** A separate window titled 'Captura de pantalla 2025-03-20 a las 15.16.38' showing audit logs for 'File\_Auditoria\_marzo2025'. It displays columns such as event\_time, action\_id, succeeded, server\_principal\_name, statement, and file\_name. The log shows various audit events, including successful logins and password changes.

\*\*No sabemos el motivo pero a pesar de que parece que todo está funcionando correctamente ya que al consultar mediante el script se muestran los registros, vemos que a través del entorno gráfico nos da un error y no se nos muestran. He mirado foros y consultado a distintas IA pero no he encontrado solución.

Gracias al apunte de un compañero, hemos descubierto la actualización que había para SSMS y al instalarla se ha resuelto el problema y hemos podido ver las auditorías desde la propia interfaz gráfica de SSMS.



## Auditoría de base de datos

Viene a ser prácticamente lo mismo que la auditoría de servidor pero a nivel base de datos, se usa el registro creado a nivel servidor y se implementa una especificación a nivel base de datos. En este caso hemos probado a decirle que audite operaciones de lectura, escritura y borrado.

```
CREATE DATABASE AUDIT SPECIFICATION [Auditoria_Autoescuela_2025]
FOR SERVER AUDIT [File_auditoria_marzo2025]
ADD (SELECT ON OBJECT::[trabajadores] BY [dbo]),
ADD (INSERT ON OBJECT::[trabajadores] BY [dbo]),
ADD (DELETE ON OBJECT::[trabajadores] BY [dbo])
GO
```

```
SELECT * FROM trabajadores;
GO
```

**\*\*Debemos acordarnos siempre de activar las especificaciones**

```
ALTER DATABASE AUDIT SPECIFICATION [Auditoria_Autoescuela_2025] WITH (STATE = ON);
GO
```

The screenshot shows the SSMS Object Explorer on the left with a tree view of the database structure. In the center, a query editor window is open with the following T-SQL code:

```
-- Auditoría de base de datos
CREATE DATABASE AUDIT SPECIFICATION [Auditoria_Autoescuela_2025]
FOR SERVER AUDIT [File_auditoria_marzo2025]
ADD (SELECT ON OBJECT::[trabajadores] BY [dbo]),
ADD (INSERT ON OBJECT::[trabajadores] BY [dbo]),
ADD (DELETE ON OBJECT::[trabajadores] BY [dbo])
GO
```

Below the code, another query is shown:

```
SELECT * FROM trabajadores;
GO
```

The results pane at the bottom shows the audit log entries for the SELECT operation:

event_time	action_id	succeeded	server_principal_name	statement	file_name
2025-03-20 15:26:39.4946145	LGIS	1	sa	-- network protocol: LPC set quoted_identifier ...	C:\auditorias\File_Aud
2025-03-20 15:26:41.2015146	LGIS	1	sa	-- network protocol: LPC set quoted_identifier ...	C:\auditorias\File_Aud
2025-03-20 15:26:42.7252441	LGIS	1	sa	-- network protocol: LPC set quoted_identifier ...	C:\auditorias\File_Aud
2025-03-20 15:26:43.1030348	LGIS	1	sa	-- network protocol: LPC set quoted_identifier ...	C:\auditorias\File_Aud
2025-03-20 15:26:49.0927443	SL	1	sa	SELECT * FROM trabajadores	C:\auditorias\File_Aud

A message at the bottom of the results pane says "Query executed successfully."

## 3.- LEDGER

LEDGER en SQL Server es una extensión de las bases de datos tradicionales que proporciona un registro inmutable de los datos. Permite a los usuarios verificar que los datos no han sido alterados ni manipulados desde su creación.

### Tabla de Historial (History Table)

Cada tabla LEDGER genera automáticamente una tabla de historial, que guarda los registros anteriores cuando se realizan actualizaciones o eliminaciones.

### Tabla de Ledger (Ledger Table)

Es una tabla que almacena datos de forma inmutable. Existen dos tipos:

### Updatable Ledger Table

Permite operaciones **INSERT, UPDATE y DELETE**, pero mantiene un historial inmutable de los cambios.

Creamos una tabla con ledger activo:

```
CREATE TABLE [dbo].[pagos] (
    [ID_pago] [int] IDENTITY(1,1) NOT NULL,
    [DNI_alumno] [varchar](9) NOT NULL,
    [Monto] [decimal](10, 2) NOT NULL,
    [Fecha_pago] [date] NOT NULL,
    [Tarjeta_credito] [varchar](25) NOT NULL,
    [Titular_tarjeta] [varchar](60) NOT NULL
)
WITH
(
    SYSTEM_VERSIONING = ON (HISTORY_TABLE = dbo.PagosHistory),
    LEDGER = ON
);
GO
```

The screenshot shows the SSMS Object Explorer on the left, displaying various database objects like tables, views, and stored procedures. The central pane shows the T-SQL script for creating a ledger table (`dbo.pagos`) and its corresponding history table (`dbo.PagosHistory`). The history table is created with the same schema as the ledger table but includes system columns for versioning and ledger management. The ledger table is created with `SYSTEM_VERSIONING = ON` and `LEDGER = ON`. The bottom pane shows the results of the execution, displaying a table of payment records with columns: ID\_pago, DNI\_alumno, Monto, Fecha\_pago, Tarjeta\_credito, and Titular\_tarjeta. The results show 6 rows of sample data.

ID_pago	DNI_alumno	Monto	Fecha_pago	Tarjeta_credito	Titular_tarjeta
1	12345678A	50.00	2024-01-10	4111111111111111	Juan Pérez
2	23456789B	75.50	2024-01-15	5500000000000004	Maria López
3	34567890C	120.00	2024-01-20	3400000000000009	Carlos Sánchez
4	45678901D	89.99	2024-01-25	3000000000000004	Laura García
5	56789012E	35.25	2024-02-05	6011000000000004	Andrés Fernández
6	67890123F	99.99	2024-02-10	3501113333000000	Paula Gómez

Query executed successfully.

Comprobamos los registros del ledger:

```

SELECT [ledger_start_transaction_id],
       [ledger_end_transaction_id],
       [ledger_start_sequence_number],
       [ledger_end_sequence_number]
  FROM pagos
 GO
  
```

	ledger_start_transaction_id	ledger_end_transaction_id	ledger_start_sequence_number	ledger_end_sequence_number
1	4664	NULL	0	NULL
2	4664	NULL	1	NULL
3	4664	NULL	2	NULL
4	4664	NULL	3	NULL
5	4664	NULL	4	NULL
6	4664	NULL	5	NULL

Query executed successfully.

Podemos obtener los registros del ledger a través de la vista:

ID_pago	DNI_alumno	Monto	Fecha_pago	Tarjeta_credito	Titular_tarjeta	ledger_transaction_id	ledger_sequence_number	ledger_operation_type	ledger_operation_type_desc
1	12345678A	50.00	2024-01-10	4111 1111 1111 1111	Juan Pérez	4664	0	1	INSERT
2	23456789B	75.50	2024-01-15	5500 0000 0000 0004	Maria López	4664	1	1	INSERT
3	34567890C	120.00	2024-01-20	3400 0000 0000 0009	Carlos Sánchez	4664	2	1	INSERT
4	45678901D	89.99	2024-01-25	3023 0000 0000 0004	Laura García	4664	3	1	INSERT
5	56789012E	35.25	2024-02-05	6011 0000 0000 0004	Andrés Fernández	4664	4	1	INSERT
6	67890123F	99.99	2024-02-10	3530 1113 3330 0000	Paula Gómez	4664	5	1	INSERT
7	78901234G	200.00	2024-02-15	6304 0000 0000 0000	David Martínez	4664	6	1	INSERT
8	89012345H	45.75	2024-02-20	4111 1111 1111 1111	Sofía Ramírez	4664	7	1	INSERT
9	90123456I	150.00	2024-02-25	5500 0000 0000 0004	Luis Ortega	4664	8	1	INSERT
10	10123456J	80.00	2024-03-01	3405 0000 0000 0009	Elena Hernández	4664	9	1	INSERT
11	11111111K	60.50	2024-03-05	3000 0000 0000 0674	Fernando Ruiz	4664	10	1	INSERT
12	12222222L	110.25	2024-03-10	6011 0000 0000 0004	Ana Domínguez	4664	11	1	INSERT
13	13333333M	95.00	2024-03-15	3530 1113 3330 0000	Pedro Salazar	4664	12	1	INSERT
14	14444444N	130.75	2024-03-20	6304 0000 0000 0000	Clara Mendoza	4664	13	1	INSERT
15	15555555O	175.00	2024-03-25	4111 1111 1111 1111	Hugo Morales	4664	14	1	INSERT
16	16666666P	220.99	2024-04-01	5500 0000 0000 0004	Marta Gil	4664	15	1	INSERT
17	77777777Q	48.75	2024-04-05	3400 0000 0000 4009	Sergio Castrón	4664	16	1	INSERT

```

SELECT * FROM sys.database_ledger_transactions
GO
  
```

transaction_id	block_id	transaction_ordinal	commit_time	principal_name	table_hashes
1	4624	0	2025-03-21 00:10:50.2733333	sa	0x5D05B621469727282DCC7EF... 0x5D05B621BEFAF4A5039DB3F6EE69B239D6B7A7024AF934F...
2	4634	0	2025-03-21 00:11:50.8700000	sa	0x5D05B621023F3C39CA7933F0C44F54E8A9790CABE960E0E1...
3	4637	0	2025-03-21 00:12:09.9700000	sa	0xA6C2220DB045135338D42274835BDAE536D19C58CC1F87...
4	4639	0	2025-03-21 00:13:16.2133333	sa	0x5D05B6213127C94096EFBD204326FBA76C4129092C96987...
5	4642	0	2025-03-21 00:16:02.5700000	sa	0x5D05B6215R7C4A8599BC1F2C6CF3F0D2A8A85ADC4D1FA...
6	4643	0	2025-03-21 00:16:21.0033333	sa	0x5D05B6215R7C4A8599BC1F2C6CF3F0D2A8A85ADC4D1FA...

Query executed successfully.

Ahora realizamos varias operaciones en la tabla:

```

INSERT INTO [dbo].[pagos] ([DNI_alumno], [Monto], [Fecha_pago], [Tarjeta_credito],
[Titular_tarjeta])
VALUES
('12345678Z', 60.00, '2024-01-15', '5555 5555 1111 1111', 'Juana Mayol');

GO

UPDATE pagos
    SET Tarjeta_credito = '5555 5555 1111 5555'
    WHERE DNI_alumno = '12345678Z';
GO

DELETE pagos
    WHERE DNI_alumno = '12345678Z';
GO

DELETE pagos
    WHERE DNI_alumno = '23456789B';
GO

```

Consultamos la vista y comprobamos que aparecen las operaciones realizadas:

	ID_pago	DNI_alumno	Monto	Fecha_pago	Tarjeta_credito	Titular_tarjeta	ledger_transaction_id	ledger_sequence_number	ledger_operation_type	ledger_operation_type_desc
12	13	3333333M	95.00	2024-03-15	3530 1113 3330 0000	Pedro Salazar	4664	12	1	INSERT
13	14	4444444N	130.75	2024-03-20	6304 0000 0000 0000	Clara Mendoza	4664	13	1	INSERT
14	15	55555555O	175.00	2024-03-25	4111 1111 1111 1111	Hugo Morales	4664	14	1	INSERT
15	16	66666666P	220.99	2024-04-01	5500 0000 0000 0004	Marta Gil	4664	15	1	INSERT
16	17	77777777Q	48.75	2024-04-05	3400 0000 0000 4009	Sergio Castro	4664	16	1	INSERT
17	18	8888888R	90.25	2024-04-10	3055 0000 0000 0004	Isabel Vega	4664	17	1	INSERT
18	19	99999999S	125.00	2024-04-15	6011 0000 0000 0004	Javier Núñez	4664	18	1	INSERT
19	20	00000000T	55.50	2024-04-20	3530 1113 3330 0000	Beatriz Tomes	4664	19	1	INSERT
20	21	12345678Z	60.00	2024-01-15	5555 5555 1111 1111	Juana Mayol	4672	0	1	INSERT
21	21	12345678Z	60.00	2024-01-15	5555 5555 1111 5555	Juana Mayol	4674	0	1	INSERT
22	2	23456789B	75.50	2024-01-15	5500 0000 0000 0004	Maria López	4664	1	1	INSERT
23	21	12345678Z	60.00	2024-01-15	5555 5555 1111 1111	Juana Mayol	4674	1	2	DELETE
24	21	12345678Z	60.00	2024-01-15	5555 5555 1111 5555	Juana Mayol	4679	0	2	DELETE
25	2	23456789B	75.50	2024-01-15	5500 0000 0000 0004	Maria López	4690	0	2	DELETE

Activar Windows

## Append-only Ledger Table

Solo permite **INSERT**, evitando cambios o eliminaciones de datos.

Creamos la tabla examen con APPEND-ONLY:

```

CREATE TABLE [dbo].[examen] (
    [ID_examen] [int] NOT NULL,
    [F_examen] [date] NULL,
    [Tipo_examen] [nvarchar] (10) NULL,
    [Tipo_carnet] [nvarchar] (10) NULL,
    [cliente_DNI_cliente] [varchar] (9) NOT NULL,
    [vehiculo_vehiculo_ID] [numeric] (28, 0) NOT NULL
)
WITH
    (LEDGER = ON (APPEND_ONLY = ON));
GO

```

The screenshot shows the SSMS interface. On the left, the Object Explorer pane displays a tree view of database objects including tables like dbo.aprobados, dbo.examen, and views like dbo.vista\_profesores. On the right, a query window titled 'SQLQuery10.sql - B...oescuelaP (sa (67))' contains several T-SQL statements. The first two statements are DELETE commands:

```

GO
DELETE pagos
WHERE DNI_alumno = '12345678Z';
GO

DELETE pagos
WHERE DNI_alumno = '23456789B';
GO

```

The third statement is a SELECT query:

```

SELECT [ledger_start_transaction_id],
       [ledger_end_transaction_id],
       [ledger_start_sequence_number],
       [ledger_end_sequence_number]
  FROM pagos
GO

```

The fourth statement is a CREATE TABLE command:

```

CREATE TABLE [dbo].[examen](
    [ID_examen] [int] NOT NULL,
    [F_examen] [date] NULL,
    [Tipo_examen] [nvarchar](10) NULL,
    [Tipo_carnet] [nvarchar](10) NULL,
    [cliente_DNI_cliente] [varchar](9) NOT NULL,
    [vehiculo_vehiculo_ID] [numeric](28, 0) NOT NULL
)
WITH
    (LEDGER = ON (APPEND_ONLY = ON));
GO

```

The 'Messages' pane at the bottom shows the command completed successfully.

Ahora vemos como podemos insertar registros en la tabla pero no nos deja borrar ni actualizar registros:

The screenshot shows a query window with the following T-SQL code:

```

-- Insertar un registro en la tabla examen
INSERT INTO [dbo].[examen]
([ID_examen], [F_examen], [Tipo_examen], [Tipo_carnet], [cliente_DNI_cliente], [vehiculo_vehiculo_ID])
VALUES
(1, '2025-03-21', 'Teórico', 'B', '12345678A', 1001);
GO

```

The 'Messages' pane at the bottom shows '(1 row affected)' and the completion time.

The status bar at the bottom right indicates 'Activar' and 'BCR\_SXBD (16.0 RTM)'.

The screenshot shows two separate sessions in SQL Server Management Studio. Both sessions attempt to modify a table named 'examen'.

**Session 1:**

```
-- Eliminar un registro de la tabla examen
DELETE FROM [dbo].[examen]
WHERE ID_examen = 1;
GO
```

Messages pane output:

Msg 37359, Level 16, State 3, Line 1221  
Updates are not allowed for the append only Ledger table 'dbo.examen'.  
Completion time: 2025-03-21T01:32:44.8427227+01:00

**Session 2:**

```
-- Actualizar un registro en la tabla examen
UPDATE [dbo].[examen]
SET
    F_examen = '2025-04-01',
    Tipo_examen = 'Práctico',
    Tipo_carnet = 'A2'
WHERE ID_examen = 1;
GO
```

Messages pane output:

Msg 37359, Level 16, State 1, Line 1227  
Updates are not allowed for the append only Ledger table 'dbo.examen'.  
Completion time: 2025-03-21T01:34:47.0082689+01:00

Both sessions result in a yellow 'Query completed with errors.' message at the bottom of their respective panes.

## Prueba detección de modificaciones

### sys.database\_ledger\_blocks

Es una vista del sistema en SQL Server que almacena el historial de los bloques de auditoría (Ledger Blocks) en una base de datos con Ledger activado. Cada bloque representa un conjunto de transacciones registradas en las tablas ledger para garantizar la inmutabilidad y la integridad de los datos.

```
SELECT * FROM sys.database_ledger_blocks
Go
```

block_id	transactions_root_hash	block_size	previous_block_hash
1	0x31014E3533E52AA42C69C30B475D5E972421825110FB18A8BF55A2C07C3D6BD0	14	NULL

### sp\_generate\_database\_ledger\_digest

El Ledger Digest es un resumen criptográfico (hash) de los datos almacenados en las tablas ledger de SQL Server. Se usa para garantizar la integridad y la inmutabilidad de los datos en una base de datos con Ledger activado.

Generamos el ledger digest :

```
EXECUTE sp_generate_database_ledger_digest
go
```

Nos da como resultado:

```
{"database_name":"AutoescuelaP","block_id":0,"hash":"0xCE4736A4A458A617ACADE243D1BF
35539A32FA8853A62F5F3CEC62FCA4A9089A","last_transaction_commit_time":"2025-03-
21T01:32:13.8433333","digest_time":"2025-03-21T12:25:27.7360992"}
```

### SET ALLOW\_SNAPSHOT\_ISOLATION ON

La Snapshot Isolation es un nivel de aislamiento en SQL Server que **evita bloqueos de lectura** y permite que las consultas lean datos consistentes sin ser afectadas por transacciones concurrentes.

```
ALTER DATABASE AutoescuelaP
    SET ALLOW_SNAPSHOT_ISOLATION ON;
GO
```

### sp\_verify\_database\_ledger

Verifica la integridad del historial de transacciones en la tabla dentro de la base de datos, usando los valores de ledger digest proporcionados.

```
EXECUTE sp_verify_database_ledger N'
[{"database_name":"AutoescuelaP","block_id":0,"hash":"0xCE4736A4A458A617ACADE243D1BF
35539A32FA8853A62F5F3CEC62FCA4A9089A","last_transaction_commit_time":"2025-03-
21T01:32:13.8433333","digest_time":"2025-03-21T12:25:27.7360992"}]
';
GO
```

### sys.fn\_PhysLocFormatter(%%physloc%%)

Devuelve la dirección de almacenamiento de cada fila con el formato: (File:Page:Slot)

- File: Número del archivo de datos donde se almacena la fila.
- Page: Número de la página dentro del archivo.
- Slot: Posición de la fila dentro de la página.

```
SELECT sys.fn_PhysLocFormatter(%%physloc%%) as [Physical RID], *
FROM dbo.pagos
Go
```

```
SELECT sys.fn_PhysLocFormatter(%%physloc%%) as [Physical RID], *
FROM dbo.pagos
go
```

The screenshot shows the SSMS interface with a query window containing the above T-SQL code. Below the code is a results grid showing 10 rows of data from the dbo.pagos table. The columns are: Physical RID, ID\_pago, DNI\_alumno, Monto, Fecha\_pago, Tarjeta\_credito, and Titular\_tarjeta. The Physical RID column contains values like '(1:1344:0)', '(1:1344:2)', etc. The results grid looks like this:

	Physical RID	ID_pago	DNI_alumno	Monto	Fecha_pago	Tarjeta_credito	Titular_tarjeta
1	(1:1344:0)	1	12345678A	50.00	2024-01-10	4111 1111 1111 1111	Juan Pérez
2	(1:1344:2)	3	34567890C	120.00	2024-01-20	3400 0000 0000 0009	Carlos Sánchez
3	(1:1344:3)	4	45678901D	89.99	2024-01-25	3023 0000 0000 0004	Laura García
4	(1:1344:4)	5	56789012E	35.25	2024-02-05	6011 0000 0000 0004	Andrés Fernández
5	(1:1344:5)	6	67890123F	99.99	2024-02-10	3530 1113 3330 0000	Paula Gómez
6	(1:1344:6)	7	78901234G	200.00	2024-02-15	6304 0000 0000 0000	David Martínez
7	(1:1344:7)	8	89012345H	45.75	2024-02-20	4111 1111 1111 1111	Sofía Ramírez
8	(1:1344:8)	9	90123456I	150.00	2024-02-25	5500 0000 0000 0004	Luis Ortega
9	(1:1344:9)	10	01234567J	80.00	2024-03-01	3405 0000 0000 0009	Elena Herrera

At the bottom of the results grid, there is a yellow bar with a checkmark icon and the text "Query executed successfully."

Podemos solicitar que nos devuelva una posición de una fila en concreto filtrando por el ID, en este caso vamos a pedir que nos muestre en qué dirección se encuentra almacenado el registro 5, que es el que queremos modificar para ver si el ledger detecta la intrusión, por ejemplo:

```
SELECT sys.fn_PhysLocFormatter(%%physloc%%) as [Physical RID], *
FROM dbo.pagos
WHERE ID_pago = 5;
GO

-- Consultamos la posición de un dato en concreto
SELECT sys.fn_PhysLocFormatter(%%physloc%%) as [Physical RID], *
FROM dbo.pagos
WHERE ID_pago = 5;
GO
```

The screenshot shows the SSMS interface with a results grid displaying the physical location of the row where ID\_pago = 5. The grid has columns: Physical RID, ID\_pago, DNI\_alumno, Monto, Fecha\_pago, Tarjeta\_credito, and Titular\_tarjeta. The Physical RID column shows '(1:1344:4)' for the row where ID\_pago is 5. The results grid looks like this:

	Physical RID	ID_pago	DNI_alumno	Monto	Fecha_pago	Tarjeta_credito	Titular_tarjeta
1	(1:1344:4)	5	56789012E	35.25	2024-02-05	6011 0000 0000 0004	Andrés Fernández

## DBCC TRACEON(3604)

Activa una traza específica (Trace Flag 3604), que redirige la salida de ciertos comandos de depuración a la ventana de resultados en SSMS en lugar de almacenarla internamente. Cuando ejecutas comandos de diagnóstico como DBCC PAGE o DBCC IND, SQL Server no muestra los resultados por defecto. Activar TRACEON(3604) permite que se impriman en la consola de SSMS.

Vamos a pedir que se nos muestre lo que contiene la página del registro que queremos modificar:

```
DBCC TRACEON (3604)
GO
DBCC PAGE(AutoescuelaP, 1, 1344, 3) -- Muestra la estructura de la página 2500
GO
```

```
-- Consultamos la página donde está el dato grabado
DBCC TRACEON(3604)
GO
DBCC PAGE(AutoescuelaP, 1, 1344, 3) -- Muestra la estructura de la página 2500
GO

100 % < >
Messages
DBCC execution completed. If DBCC printed error messages, contact your system administrator.

PAGE: (1:1344)

BUFFER:

BUF @0x0000025D4DAD5B80

bpage = 0x0000025BE1D08000      bPrmpage = 0x00000000000000000000
bsort_r_prevbP = 0x00000000000000000000 bhash = 0x00000000000000000000
bpart = 0                      bstat = 0xb
berrcode = 0                   bUse1 = 8125
blog = 0x15ac                 bsampleCount = 0
resPoolId = 0                  bcputicks = 0
bDirtyPendingCount = 0          bDirtyContext = 0x0000025BE1A7C250
bdbid = 12                     bpru = 0x0000025BD40A8040
bsort_r_nextbP = 0x00000000000000000000
bpageno = (1:1344)
breferences = 0
bstat2 = 0x0
bIoCount = 0
bReadMicroSec = 943
bDbPageBroker = 0x00000000000000000000

100 % < >

-- Consultamos la página donde está el dato grabado
DBCC TRACEON(3604)
GO
DBCC PAGE(AutoescuelaP, 1, 1344, 3) -- Muestra la estructura de la página 2500
GO

100 % < >
Messages
Slot 4 Column 1 Offset 0x4 Length 4 Length (physical) 4
ID_pago = 5
Slot 4 Column 2 Offset 0x40 Length 9 Length (physical) 9
DNI_alumno = 56789012E
Slot 4 Column 3 Offset 0x8 Length 9 Length (physical) 9
Monto = 35.25
Slot 4 Column 4 Offset 0x11 Length 3 Length (physical) 3
Fecha_pago = 2024-02-05
Slot 4 Column 5 Offset 0x49 Length 19 Length (physical) 19
Tarjeta_credito = 6011 0000 0000 0004
Slot 4 Column 6 Offset 0x5c Length 16 Length (physical) 16
Titular_tarjeta = Andrés Fernández
```

Ahora vamos a convertir a VARBINARY los números de las tarjetas, tanto la buena como la pirateada:

```
SELECT CONVERT (VARBINARY(33), '6011 0000 0000 0004') -- Conversión de tarjeta buena
GO
-- 0x3630313120303030203030302030303034
```

```
SELECT CONVERT (VARBINARY(33), '6666 6666 6666 6666') -- Conversión de tarjeta
pirateada
GO
-- 0x36363636203636363620363636362036363636
```

Ahora viene la parte en la que “corrompemos” los datos de la base de datos y que pretendemos detectar con el ledger. Para ello, primero vamos a desactivar la verificación de páginas:

```
ALTER DATABASE AutoescuelaP SET PAGE_VERIFY NONE
GO
```

Cambiamos la base de datos a modo monousuario:

```
ALTER DATABASE AutoescuelaP SET SINGLE_USER
GO
```

Ahora reescribimos lo que queremos en la página:

```
DBCC WRITEPAGE
('AutoescuelaP',1,544,104,33,0x3636363620363636203636362036363620363636,0)
GO
```

Ahora volvemos a poner la base de datos en multiusuario y activamos el control de página:

```
ALTER DATABASE AutoescuelaP SET MULTI_USER
GO
ALTER DATABASE AutoescuelaP SET PAGE_VERIFY CHECKSUM
GO
```

Al realizar una consulta nos damos cuenta que hemos cometido un error al introducir el comando para modificar la tarjeta:

```
-- Consultamos la tabla
SELECT * FROM pagos
GO
```

An error occurred while executing batch. Error message is: Arithmetic Overflow.

Completion time: 2025-03-21T16:06:15.8940764+01:00

El error estaba en que habíamos introducido mal el offset de la sentencia de modificación, ahora si vemos la tarjeta modificada:

ID_pago	DNI_alumno	Monto	Fecha_pago	Tarjeta_credito	Titular_tarjeta
1	12345678A	50.00	2024-01-10	4111 1111 1111 1111	Juan Pérez
2	34567890C	120.00	2024-01-20	3400 0000 0000 0009	Carlos Sánchez
3	45678901D	89.99	2024-01-25	3023 0000 0000 0004	Laura García
4	56789012E	35.25	2024-02-05	6666 6666 6666 6666	Andrés Fernández
5	67890123F	99.99	2024-02-10	3530 1113 3330 0000	Paula Gómez
6	78901234G	200.00	2024-02-15	6304 0000 0000 0000	David Martínez
7	89012345H	45.75	2024-02-20	4111 1111 1111 1111	Sofía Ramírez
8	90123456I	150.00	2024-02-25	5500 0000 0000 0004	Luis Ortega
9	01234567J	80.00	2024-03-01	3405 0000 0000 0009	Elena Herrera
10	11111111K	60.50	2024-03-05	3000 0000 0000 0674	Fernando Ruiz

Query executed successfully.

Ahora al comprobar la integridad de la base de datos con ledger, nos dice que el hash asociado no coincide con el hash actual y por tanto la base de datos está corrupta o modificada.

```
EXECUTE sp_verify_database_ledger N'
[{"database_name":"AutoescuelaP","block_id":0,"hash":"0xCE4736A4A458A617ACADE243D1B
F35539A32FA8853A62F5F3CEC62FCA4A9089A","last_transaction_commit_time":"2025-03-
21T01:32:13.8433333","digest_time":"2025-03-21T12:25:27.7360992"}]
';
GO
```

```
EXECUTE sp_verify_database_ledger N'
[{"database_name":"AutoescuelaP","block_id":0,"hash":"0xCE4736A4A458A617ACADE243D1B
F35539A32FA8853A62F5F3CEC62FCA4A9089A","last_transaction_commit_time":"2025-03-
21T01:32:13.8433333","digest_time":"2025-03-21T12:25:27.7360992"}]
';
GO
```

100 %

Messages

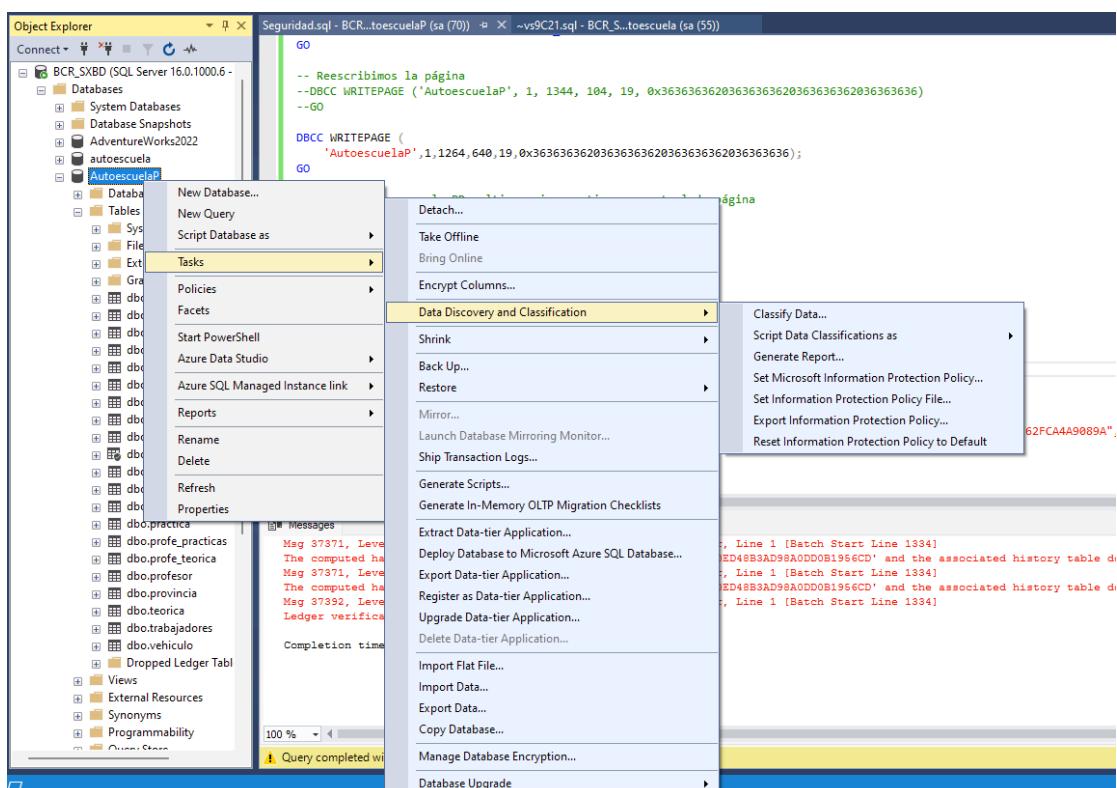
Msg 37371, Level 16, State 1, Procedure sp\_verify\_database\_ledger, Line 1 [Batch Start Line 1334]  
The computed hash from 'MSSQL\_DroppedLedgerTable\_pages\_7D4A531D00ED48B3AD98A0DD0B1956CD' and the associated history table does not match the hash persisted in sys.database\_l1  
Msg 37371, Level 16, State 1, Procedure sp\_verify\_database\_ledger, Line 1 [Batch Start Line 1334]  
The computed hash from 'MSSQL\_DroppedLedgerTable\_pages\_7D4A531D00ED48B3AD98A0DD0B1956CD' and the associated history table does not match the hash persisted in sys.database\_l1  
Msg 37392, Level 16, State 1, Procedure sp\_verify\_database\_ledger, Line 1 [Batch Start Line 1334]  
Ledger verification failed.

Completion time: 2025-03-21T16:28:17.0681879+01:00

## 4.- SQL Data Discovery and Classification in SSMS

### SQL Data Discovery

Es una herramienta en SSMS que sirve para clasificar el tipo de información que se guarda en cada columna según la sensibilidad de la misma. En un primer momento, el propio SSMS nos realiza una primera clasificación automática que luego nosotros podremos modificar o bien añadir nuevas clasificaciones. Pinchamos con el botón derecho sobre nuestra base de datos y luego en “Task->Data Discovery and Classification->Clasify Data”.



The screenshot shows the SSMS Data Classification interface. At the top, it says "Data Classification... 25 19:56 - BCR\_SXBD Data Classification - AutoescuelaP Seguridad.sql - BCR...toescuelaP (sa (70)) -vs9C21.sql - BCR\_S...toescuela (sa (55))". Below this, a message says "The classification changes have been updated successfully." A tooltip indicates "We have found 13 columns with classification recommendations. Click here to hide recommendations." A dropdown menu allows selecting a schema (dbo) and table (pagos), with a "Load Columns" button. A link "Learn more - Getting Started Guide" is also present. The main area displays a table titled "13 columns with classification recommendations (click to minimize)". The table has columns: Schema, Table, Column, Information Type, and Sensitivity Label. The data shows various columns from different tables being classified as "Contact Info" or "Credit Card" with "Confidential" sensitivity. A message at the bottom right says "Activar Windows Ve a Configuración para activar Windows.".

Aunque en un primer momento, el propio SSMS nos hace una clasificación, si pinchamos en la parte superior en “Add classification”, podemos añadir nosotros la clasificación que queramos de cada dato de la base de datos. Se desplegará una columna lateral en donde escogeremos la columna, el tipo de información y la etiqueta de seguridad. Por ejemplo a la cuenta bancaria o tarjeta de crédito podemos ponerle “Financial” y “Confidencial GDPR”.

The screenshot shows the SSMS Data Classification interface. At the top, it says "Data Classification - AutoescuelaP Seguridad.sql - BCR...toescuelaP (sa (70)) -vs9C21.sql - BCR\_S...toescuela (sa (55))". A message says "No classified columns found." A tooltip indicates "We have found 14 columns with classification recommendations. Click here to view them." A dropdown menu allows selecting a schema (dbo) and table (pagos), with a "Load Columns" button. The main area displays a table titled "0 classified columns". On the right, a "Add Classification" dialog box is open, titled "Adding classification for dbo.pagos". It contains fields for "Column" (set to "Tarjeta\_credito"), "Information Type" (set to "Financial"), and "Sensitivity Label" (set to "Confidential - GDPR"). Buttons for "Add" and "Cancel" are at the bottom of the dialog.

Data Classification - AutoescuelaP Seguridad.sql - BCR...toescuelaP (sa (70)) ~vs9C21.sql - BCR\_S...toescuela (sa (55))

[Save](#) [Add Classification](#) [View Report](#) [Clear and Refresh](#)

There are pending classification updates. Please save.

We have found 13 columns with classification recommendations. Click here to view them.

Choose a table to view its classified columns:

Schema: dbo Table: pagos Load Columns

3 classified columns

Column	Information Type	Sensitivity Label
Tarjeta_credito	Financial	Confidential - GDPR
DNI_alumno	Contact Info	Confidential - GDPR
Titular_tarjeta	Contact Info	Confidential - GDPR

Además, si pinchamos en “View report” se nos generará un reporte de los distintos tipos de datos y clasificación.

Data Classification...25 19:56 - BCR\_SXBD Data Classification - AutoescuelaP Seguridad.sql - BCR...toescuelaP (sa (70)) ~vs9C21.sql - BCR\_S...toescuela (sa

[View Report](#)

## SQL Data Classification Report

Microsoft

Server name: BCR\_SXBD Database name: AutoescuelaP Report generated on: 21/03/2025 19:56:39

Classified columns: 6 / 217 Tables containing sensitive data: 2 / 30 Unique information types: 2

Label distribution: 6 Confidential - GDPR

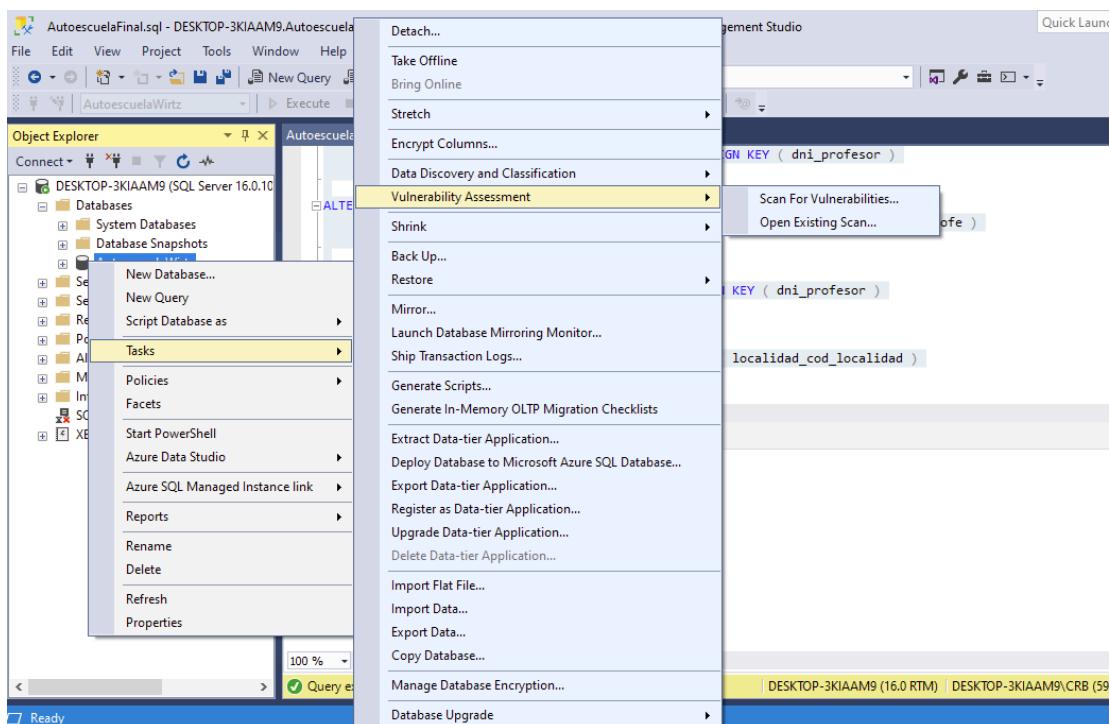
Information Type distribution: 4 Contact Info, 2 Financial

Schema	Table	Column	Information Type	Sensitivity Label
dbo	pagos	DNI_alumno	Contact Info	Confidential - GDPR

## Vulnerability Assessment

Es una evaluación de vulnerabilidades, el propio sistema te escanea tu base de datos y te devuelve los errores y posibles vulnerabilidades que tiene la base de datos y que se deben corregir. Sin embargo, la funcionalidad de 'Vulnerability Assessment' fue eliminada de SSMS a partir de la versión 19.1. Esta decisión se tomó porque Microsoft consolidó esta capacidad en una solución más amplia llamada Microsoft Defender for SQL, que ofrece herramientas más actualizadas y completas para evaluar y proteger la seguridad de las bases de datos.

Hemos instalado SSMS 18 en otra máquina virtual y luego restaurado nuestra base de datos para realizar la prueba.



Como podemos observar, al aplicarlo a nuestra base de datos nos detecta dos vulnerabilidades y una de ellas concretamente nos dice que la base de datos no tiene encriptación TDE y por tanto los datos no están protegidos. Vamos a activar el TDE y luego comprobaremos si se ha solucionado el fallo de seguridad.

Total failing checks	Total passing checks	High Risk	Medium Risk	Low Risk
2	33	0	2	0

ID	Security Check	Category	Risk	Additional Information
VA1143	'dbo' user should not be used for normal service operation	Surface Area Reduction	Medium	
VA1219	Transparent data encryption should be enabled	Data Protection	Medium	

Una vez encriptada la base de datos, comprobamos:

```

ALTER DATABASE AutoescuelaWirtz
SET ENCRYPTION ON;

BACKUP CERTIFICATE TDE_Certificate
TO FILE = 'C:\TDE_Certificate.cer'
WITH PRIVATE KEY (
FILE = 'C:\TDE_PrivateKey.pvk',
ENCRYPTION BY PASSWORD = 'otra_clave_segura');

SELECT database_id, name, is_encrypted
FROM sys.databases
WHERE name = 'AutoescuelaWirtz';

```

The screenshot shows the SSMS interface with the above T-SQL script. Below the script, the results pane displays a table with one row:

database_id	name	is_encrypted
5	AutoescuelaWirtz	1

Ahora ya vemos como el fallo de seguridad está resuelto y ya solo nos marca un fallo.

The screenshot shows the 'Vulnerability Assessment Results' tool interface. It displays the following statistics:

- Total failing checks: 1 (marked with a red circle)
- Total passing checks: 34 (marked with a green circle)
- Risk distribution: High Risk: 0, Medium Risk: 1, Low Risk: 0

Below the statistics, there is a table with one row:

ID	Security Check	Category	Risk	Additional Information
VA1143	'dbo' user should not be used for normal service operation	Surface Area Reduction	Medium	

## 5.- Ataques contra bases de datos

### Inyección SQL

La inyección SQL es un método mediante el cual se inserta en un formulario una consulta SQL en vez de los datos solicitados en el formulario. A través de esta consulta se accede a la base de datos para acceder a su información o bien para inutilizarla. El caso más reciente que conozco es el del Hospital Clínico de Barcelona en 2023 que aunque no se confirmara que fuese mediante inyección SQL, fue lo que se sospechó en aquel momento.

### Ransomware

El ransomware es un tipo de ataque que encripta o secuestra los datos de un centro o empresa y luego piden un rescate por ellos. No es extraño ver vinculada la inyección SQL para acceder a la base de datos y luego el ransomware para encriptar los datos y posteriormente pedir un rescate por ellos. Como ejemplo, podemos volver a citar el caso del Hospital Clínico de Barcelona donde una vez accedieron a los datos, los encriptaron y pidieron rescate por ellos. Por lo que se dijo en el momento, el ataque fue por parte del grupo RansomHouse y se dijo que no se pagó rescate por los datos y fueron devueltos semanas después.

## **Medidas preventivas**

Si fuésemos los encargados de la seguridad de los datos de una empresa, la principal medida para evitar ataques de inyección SQL sería limitar los datos que se pueden introducir en formularios a los caracteres y formatos estrictamente requeridos por el formulario, evitando que en dichos campos se puedan enviar consultas complejas. Además, podemos realizar auditorías para ver quien ha accedido a la base de datos y de que manera o podemos utilizar herramientas de escaneo de vulnerabilidades (por ejemplo las vistas en este tema en el propio SSMS). Respecto a la prevención contra ransomware, lo principal sería tener copias de seguridad de la información para, en caso de robo o bloqueo de la misma, que podamos seguir funcionando con normalidad y no paralizar la actividad de la empresa. También se deberían bloquear la ejecución de scripts y sobretodo, en lo que mas se suele incidir es en la concienciación de los trabajadores ya que la mayor parte de los ataques suelen ser por errores o despistes del personal que trabaja en la empresa.