



TAREA3: Modelado -Adm. SQL SERVER

TAREA3: Modelado -Adm. SQL SERVER | Notion

1.- Análisis previo

<https://galvanized-text-b5c.notion.site/TAREA3-Modelado-Adm-SQL-SERVER-11abadfb262d80e68bbec8ebf72b0be5?pvs=4>



TAREA3: Modelado -Adm. SQL ...

Creado con Notion

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

1.- Análisis previo

- [1.1.- Especificaciones y requerimientos](#)
- [1.2.- Supuestos semánticos](#)

2.- Modelo lógico

3.- Modelo relacional

4.- Modelo físico

5.- Script

[Backup de nuestra base de datos](#)

[Instalación base de datos de ejemplo](#)

- [Restore Backup desde SSMS](#)
- [Restore desde script](#)
- [Restore desde Powershell](#)
- [**Restore con attach**](#)

6.- Bases de datos contenidas

7.- Planes de mantenimiento

8. BLOB (Binary Large Object)

- [Opciones para BLOBs \(Binary Large Objects\) en SQL Server](#)
- [NTFS \(Carpetas del sistema operativo\)](#)
- [FILESTREAM](#)
- [Otro ejemplo FILESTREAM](#)
- [FILETABLES](#)

[POWERBI](#)

9.- Particiones

- [SPLIT](#)
- [MERGE](#)
- [SWITCH](#)
- [TRUNCATE](#)
- [REPASO CREAR PARTICIÓN](#)
- [Otro ejemplo Particiones \(integer\)](#)

[10.- Tablas temporales versión sistema](#)

[AS OF](#)
[FROM](#)
[BETWEEN](#)
[CONTAINED IN](#)
[Limitaciones](#)

[11.- Tablas en memoria](#)

1.- Análisis previo

La autoescuela Fernando Wirtz nos pide que implementemos una base de datos para gestionar desde los contratos y nóminas de sus trabajadores, sus vehículos, sus clientes y todas las clases teóricas y prácticas que estos realizan.

Las condiciones que se nos piden es que se pueda distinguir entre clases teóricas o prácticas, que se pueda ver que vehículo ha usado cada profesor y cada alumno para cada práctica y que puedan acceder a todos los datos de los trabajadores, desde la gestión de nóminas hasta las clases que imparten. Además, quieren que se reflejen también los exámenes que hace cada alumno.

1.1.- Especificaciones y requerimientos

De los trabajadores interesa conocer todos sus datos personales y además sus datos laborales (Nóminas, contrato, horarios, salario, antigüedad...). También debemos conocer las prácticas que realizan y con que vehículo las hacen ya que no tienen un vehículo asignado y va adjudicándose uno para cada práctica que imparten, además debemos tener en cuenta si son profesores de teoría o de práctica.

Tendremos que reflejar también las clases teóricas y prácticas que imparte cada uno.

De los alumnos/clientes debemos de reflejar también sus datos personales, las prácticas que realizan y con que profesor y vehículo las realizan. También los exámenes a los que se presentan.

Debemos llevar un control de los vehículos, tanto de sus datos y fecha de adquisición como de las prácticas que hacen y que profesores y alumnos los usan.

1.2.- Supuestos semánticos

Un profesor puede impartir clases teóricas o clases prácticas pero no los dos tipos de clases. Cada profesor tendrá un solo contrato (va a interesar solo el contrato en vigor) pero podrá tener cero, una o varias nóminas.

Un alumno podrá recibir clases teóricas y prácticas a la vez, ya que puede estar pendiente de examen teórico e ir empezando a hacer prácticas. Además un alumno podrá presentarse a varios exámenes pero cada examen solo tendrá un alumno (aunque en el aula o en el vehículo del examen práctico concurren más alumnos) puesto que el examen es algo personal.

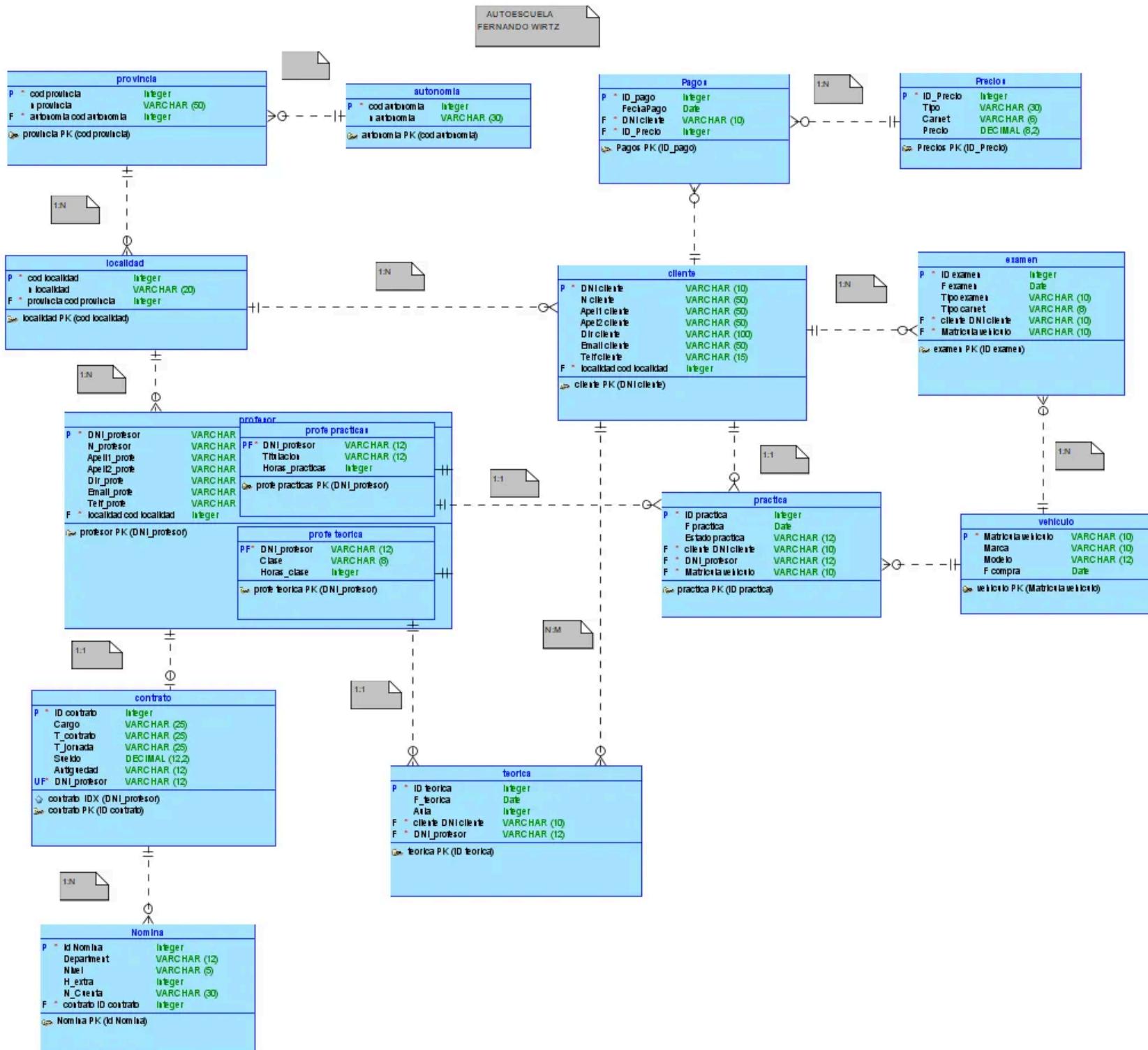
Un vehículo podrá realizar varias prácticas pero no podrá realizarse una práctica sin un vehículo, a su vez, en un examen podrá utilizarse un vehículo y un vehículo podrá ser utilizado en un examen.

2.- Modelo lógico

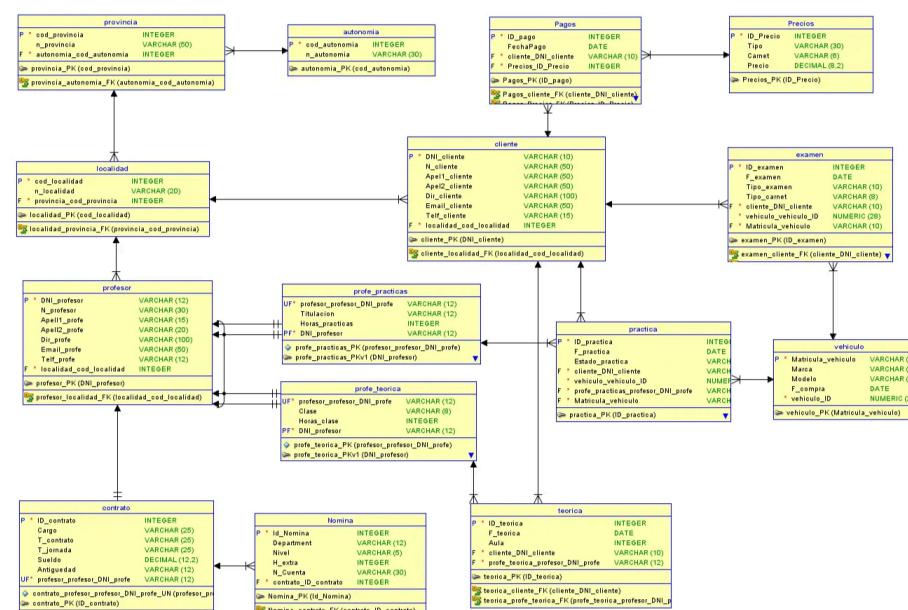
Hemos optado por una especialización con el tipo de profesor, ya que cada profesor imparte clases distintas y un profesor de prácticas no podrá impartir clases teóricas y viceversa, además de esta manera asignamos a cada práctica y teoría un profesor de ese departamento. Sin embargo, pensamos en hacer lo mismo con el tipo de examen (teórico o práctico) pero finalmente no lo hemos hecho, ya que la única diferencia entre un tipo de examen y otro es que lleva vinculado un vehículo y esto lo hemos solucionado estableciendo una relación 1-N donde un examen podrá no tener vehículo asignado pero un vehículo podrá tener asignados varios exámenes o ninguno.

Hemos añadido también los gastos que cada cliente hace en la autoescuela a partir de una tabla que contiene los precios de cada producto.

Al aplicar la normalización, hemos extraído las tablas nómina y contrato de la tabla trabajador y de la misma manera hemos extraído las tablas localidad, provincia y autonomía de la tabla trabajador y cliente para evitar la dependencia de otro atributo de la misma tabla.



3.- Modelo relacional



4.- Modelo físico

Una vez hayamos hecho el modelado lógico en Datamodeler y lo hayamos pasado a relacional, generaremos el archivo .ddl que contendrá el script que podremos trasladar a SQL Server para implementar allí nuestra base de datos. Luego abrimos SQL Server y abrimos el archivo .ddl, antes de ejecutar el código, añadiremos al inicio de este las sentencias para crear la base de datos y activarla:

```
DROP DATABASE IF EXISTS AutoescuelaWirtz;
GO
CREATE DATABASE AutoescuelaWirtz;
GO
Use AutoescuelaWirtz;
GO
```

Ahora ya solo nos queda ejecutar la query y se nos creará la base de datos y las tablas.

De aquí sacamos el modelo físico desde la propia base de datos y el apartado "Database Diagrams", botón derecho y "New Database Diagram", luego seleccionamos todas las tablas y se generará el siguiente esquema físico:

https://prod-files-secure.s3.us-west-2.amazonaws.com/a8d14bb1-8a2d-4184-8380-1227d7b2836e/d0f068d2-9d69-46b2-86c1-ab2c2a7a4645/Modelado_fisico.pdf

5.- Script

```
CREATE DATABASE AutoescuelaWirtz;
GO

USE AutoescuelaWirtz;
GO

CREATE TABLE autonomia (
    cod_autonomia INTEGER NOT NULL,
    n_autonomia  VARCHAR(30)
);

ALTER TABLE autonomia ADD CONSTRAINT autonomia_pk PRIMARY KEY ( cod_autonomia );

CREATE TABLE cliente (
    dni_cliente      VARCHAR(10) NOT NULL,
    n_cliente        VARCHAR(50),
    apel1_cliente   VARCHAR(50),
    apel2_cliente   VARCHAR(50),
    dir_cliente     VARCHAR(100),
    email_cliente   VARCHAR(50),
    telf_cliente    VARCHAR(15),
    localidad_cod_localidad INTEGER NOT NULL
);

ALTER TABLE cliente ADD CONSTRAINT cliente_pk PRIMARY KEY ( dni_cliente );

CREATE TABLE contrato (
    id_contrato      INTEGER NOT NULL,
    cargo            VARCHAR(25),
    t_contrato       VARCHAR(25),
    t_jornada        VARCHAR(25),
    sueldo           DECIMAL(12, 2),
    antiguedad       VARCHAR(12),
    profesor_profesor_dni_profe VARCHAR(12) NOT NULL
);
```

```
);
```

```
ALTER TABLE contrato ADD CONSTRAINT contrato_pk PRIMARY KEY ( id_contrato );
```

```
ALTER TABLE contrato ADD CONSTRAINT contrato_profesor_profesor_dni_profe_un UNIQUE ( profesor_profesor_
```

```
CREATE TABLE examen (
    id_examen      INTEGER NOT NULL,
    f_examen       DATE,
    tipo_examen    VARCHAR(10),
    tipo_carnet    VARCHAR(8),
    cliente_dni_cliente VARCHAR(10) NOT NULL,
    matricula_vehiculo  VARCHAR(10) NOT NULL
);
```

```
ALTER TABLE examen ADD CONSTRAINT examen_pk PRIMARY KEY ( id_examen );
```

```
CREATE TABLE localidad (
    cod_localidad   INTEGER NOT NULL,
    n_localidad     VARCHAR(20),
    provincia_cod_provincia INTEGER NOT NULL
);
```

```
ALTER TABLE localidad ADD CONSTRAINT localidad_pk PRIMARY KEY ( cod_localidad );
```

```
CREATE TABLE nomina (
    id_nomina      INTEGER NOT NULL,
    department     VARCHAR(12),
    nivel          VARCHAR(5),
    h_extra         INTEGER,
    n_cuenta       VARCHAR(30),
    contrato_id_contrato INTEGER NOT NULL
);
```

```
ALTER TABLE nomina ADD CONSTRAINT nomina_pk PRIMARY KEY ( id_nomina );
```

```
CREATE TABLE pagos (
    id_pago        INTEGER NOT NULL,
    fechapago      DATE,
    cliente_dni_cliente VARCHAR(10) NOT NULL,
    precios_id_precio  INTEGER NOT NULL
);
```

```
ALTER TABLE pagos ADD CONSTRAINT pagos_pk PRIMARY KEY ( id_pago );
```

```
CREATE TABLE practica (
    id_practica    INTEGER NOT NULL,
    f_practica     DATE,
    estado_practica  VARCHAR(12),
    cliente_dni_cliente VARCHAR(10) NOT NULL,
    profe_practicas_profesor_dni_profe VARCHAR(12) NOT NULL,
    matricula_vehiculo  VARCHAR(10) NOT NULL
);
```

```
ALTER TABLE practica ADD CONSTRAINT practica_pk PRIMARY KEY ( id_practica );
```

```
CREATE TABLE precios (
    id_precio      INTEGER NOT NULL,
    tipo           VARCHAR(30),
    carnet         VARCHAR(6),
```

```

precio DECIMAL(8, 2)
);

ALTER TABLE precios ADD CONSTRAINT precios_pk PRIMARY KEY ( id_precio );

CREATE TABLE profe_practicas (
    profesor_profesor_dni_profe VARCHAR(12) NOT NULL,
    titulacion      VARCHAR(12),
    horas_practicas   INTEGER,
    dni_profesor     VARCHAR(12) NOT NULL
);

ALTER TABLE profe_practicas ADD CONSTRAINT profe_practicas_pkv1 PRIMARY KEY ( dni_profesor );

ALTER TABLE profe_practicas ADD CONSTRAINT profe_practicas_pk UNIQUE ( profesor_profesor_dni_profe );

CREATE TABLE profe_teorica (
    profesor_profesor_dni_profe VARCHAR(12) NOT NULL,
    clase          VARCHAR(8),
    horas_clase    INTEGER,
    dni_profesor    VARCHAR(12) NOT NULL
);

ALTER TABLE profe_teorica ADD CONSTRAINT profe_teorica_pkv1 PRIMARY KEY ( dni_profesor );

ALTER TABLE profe_teorica ADD CONSTRAINT profe_teorica_pk UNIQUE ( profesor_profesor_dni_profe );

CREATE TABLE profesor (
    dni_profesor    VARCHAR(12) NOT NULL,
    n_profesor      VARCHAR(30),
    apell1_profe    VARCHAR(15),
    apell2_profe    VARCHAR(20),
    dir_profe       VARCHAR(100),
    email_profe     VARCHAR(50),
    telf_profe      VARCHAR(12),
    localidad_cod_localidad INTEGER NOT NULL
);

ALTER TABLE profesor ADD CONSTRAINT profesor_pk PRIMARY KEY ( dni_profesor );

CREATE TABLE provincia (
    cod_provincia    INTEGER NOT NULL,
    n_provincia      VARCHAR(50),
    autonomia_cod_autonomia INTEGER NOT NULL
);

ALTER TABLE provincia ADD CONSTRAINT provincia_pk PRIMARY KEY ( cod_provincia );

CREATE TABLE teorica (
    id_teorica        INTEGER NOT NULL,
    f_teorica         DATE,
    aula              INTEGER,
    cliente_dni_cliente VARCHAR(10) NOT NULL,
    profe_teorica_profesor_dni_profe VARCHAR(12) NOT NULL
);

ALTER TABLE teorica ADD CONSTRAINT teorica_pk PRIMARY KEY ( id_teorica );

CREATE TABLE vehiculo (
    matricula_vehiculo VARCHAR(10) NOT NULL PRIMARY KEY,

```

```

marca      VARCHAR(10),
modelo     VARCHAR(12),
f_compra   DATE,
);

ALTER TABLE cliente
ADD CONSTRAINT cliente_localidad_fk FOREIGN KEY ( localidad_cod_localidad )
REFERENCES localidad ( cod_localidad );

ALTER TABLE contrato
ADD CONSTRAINT contrato_profesor_fk FOREIGN KEY ( profesor_profesor_dni_profe )
REFERENCES profesor ( dni_profesor );

ALTER TABLE examen
ADD CONSTRAINT examen_cliente_fk FOREIGN KEY ( cliente_dni_cliente )
REFERENCES cliente ( dni_cliente );

ALTER TABLE examen
ADD CONSTRAINT examen_vehiculo_fk FOREIGN KEY ( matricula_vehiculo )
REFERENCES vehiculo ( matricula_vehiculo );

ALTER TABLE localidad
ADD CONSTRAINT localidad_provincia_fk FOREIGN KEY ( provincia_cod_provincia )
REFERENCES provincia ( cod_provincia );

ALTER TABLE nomina
ADD CONSTRAINT nomina_contrato_fk FOREIGN KEY ( contrato_id_contrato )
REFERENCES contrato ( id_contrato );

ALTER TABLE pagos
ADD CONSTRAINT pagos_cliente_fk FOREIGN KEY ( cliente_dni_cliente )
REFERENCES cliente ( dni_cliente );

ALTER TABLE pagos
ADD CONSTRAINT pagos_precios_fk FOREIGN KEY ( precios_id_precio )
REFERENCES precios ( id_precio );

ALTER TABLE practica
ADD CONSTRAINT practica_cliente_fk FOREIGN KEY ( cliente_dni_cliente )
REFERENCES cliente ( dni_cliente );

ALTER TABLE practica
ADD CONSTRAINT practica_profe_practicas_fk FOREIGN KEY ( profe_practicas_profesor_dni_profe )
REFERENCES profe_practicas ( profesor_profesor_dni_profe );

ALTER TABLE practica
ADD CONSTRAINT practica_vehiculo_fk FOREIGN KEY ( matricula_vehiculo )
REFERENCES vehiculo ( matricula_vehiculo );

ALTER TABLE profe_practicas
ADD CONSTRAINT profe_practicas_profesor_fk FOREIGN KEY ( profesor_profesor_dni_profe )
REFERENCES profesor ( dni_profesor );

ALTER TABLE profe_practicas
ADD CONSTRAINT profe_practicas_profesor_fkv2 FOREIGN KEY ( dni_profesor )
REFERENCES profesor ( dni_profesor );

ALTER TABLE profe_teorica
ADD CONSTRAINT profe_teorica_profesor_fk FOREIGN KEY ( profesor_profesor_dni_profe )
REFERENCES profesor ( dni_profesor );

```

```
ALTER TABLE profe_teorica
ADD CONSTRAINT profe_teorica_profesor_fkv2 FOREIGN KEY ( dni_profesor )
    REFERENCES profesor ( dni_profesor );
```

```
ALTER TABLE profesor
ADD CONSTRAINT profesor_localidad_fk FOREIGN KEY ( localidad_cod_localidad )
    REFERENCES localidad ( cod_localidad );
```

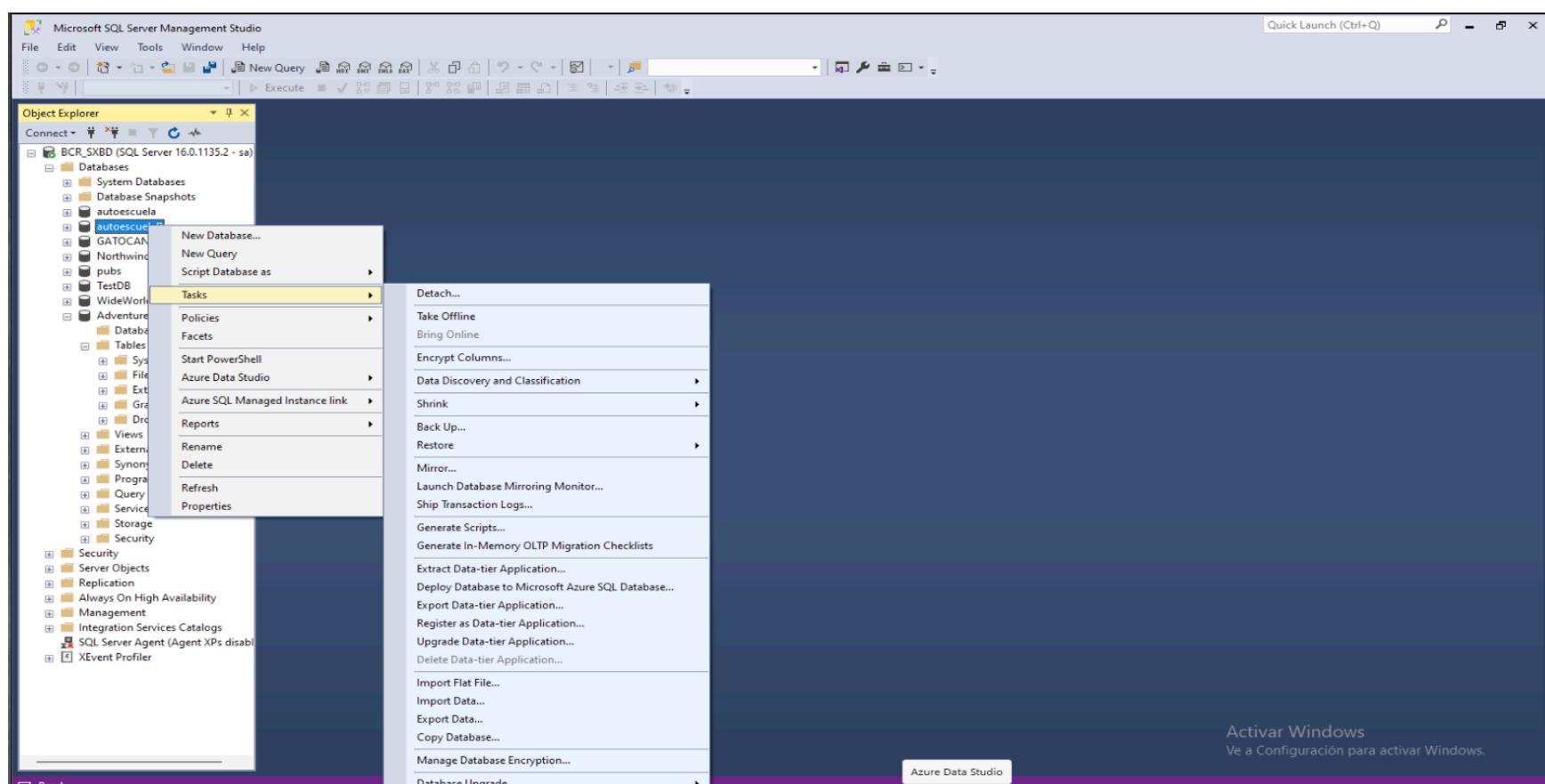
```
ALTER TABLE provincia
ADD CONSTRAINT provincia_autonomia_fk FOREIGN KEY ( autonomia_cod_autonomia )
    REFERENCES autonomia ( cod_autonomia );
```

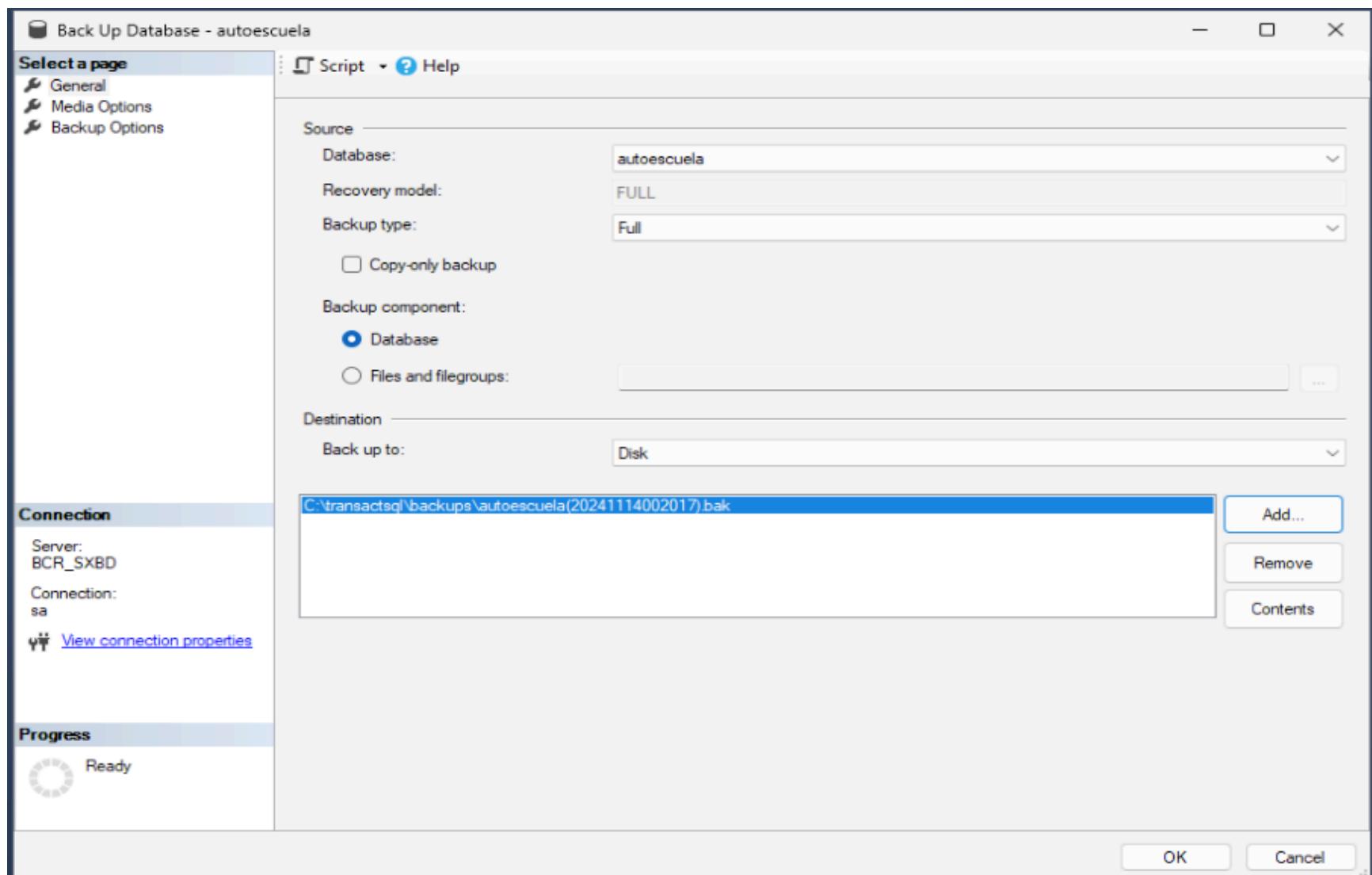
```
ALTER TABLE teorica
ADD CONSTRAINT teorica_cliente_fk FOREIGN KEY ( cliente_dni_cliente )
    REFERENCES cliente ( dni_cliente );
```

```
ALTER TABLE teorica
ADD CONSTRAINT teorica_profe_teorica_fk FOREIGN KEY ( profe_teorica_profesor_dni_profe )
    REFERENCES profe_teorica ( profesor_profesor_dni_profe );
```

Backup de nuestra base de datos

Para finalizar, realizaremos un backup de nuestra base de datos desde SSMS para guardar una copia por si surgiese cualquier problema mientras trabajamos con ella. Para ello, desde SSMS, en el navegador de la barra lateral vamos a “Tasks→Back up...”, aquí seleccionamos el directorio donde queremos que se guarde el backup y presionamos aceptar.



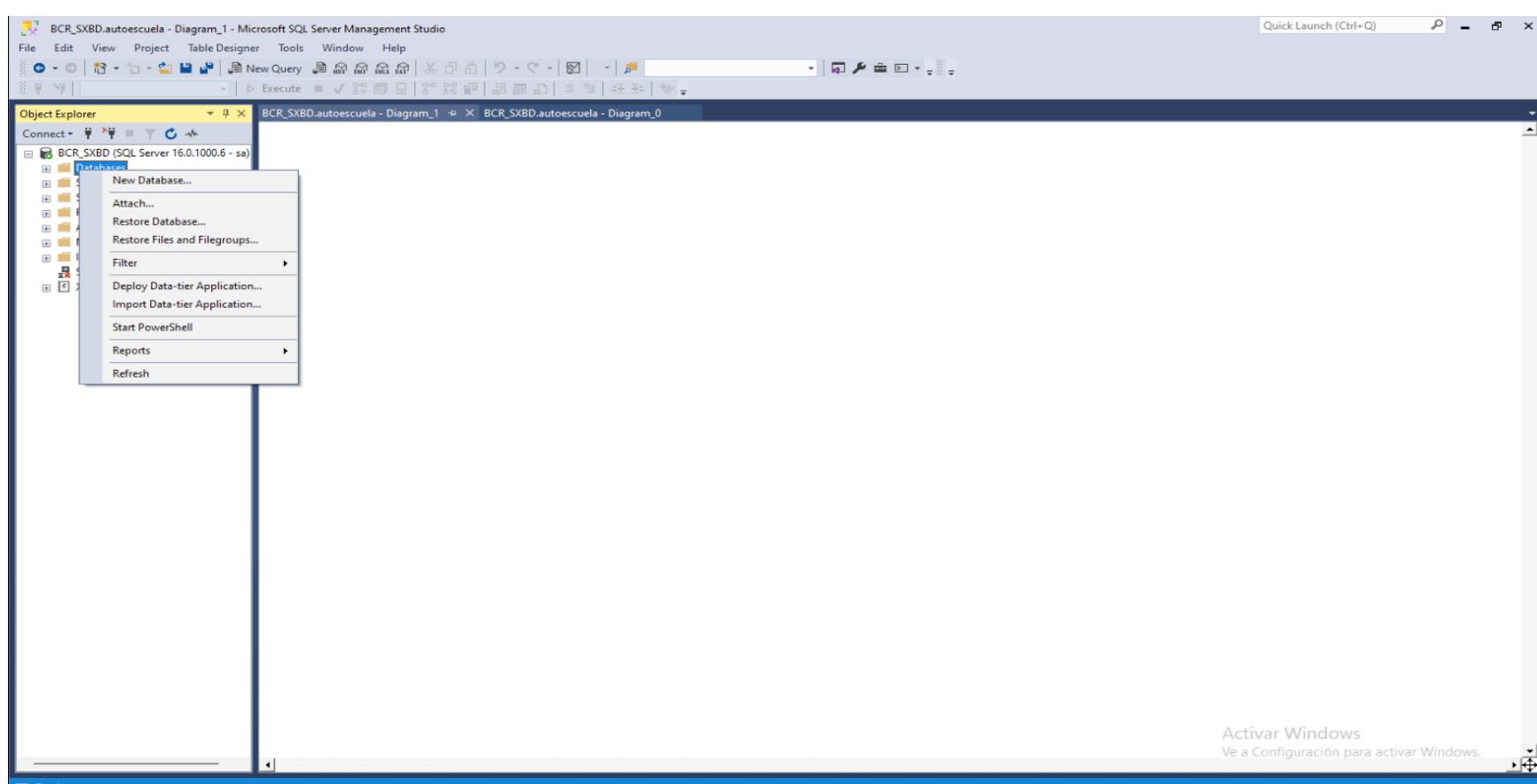


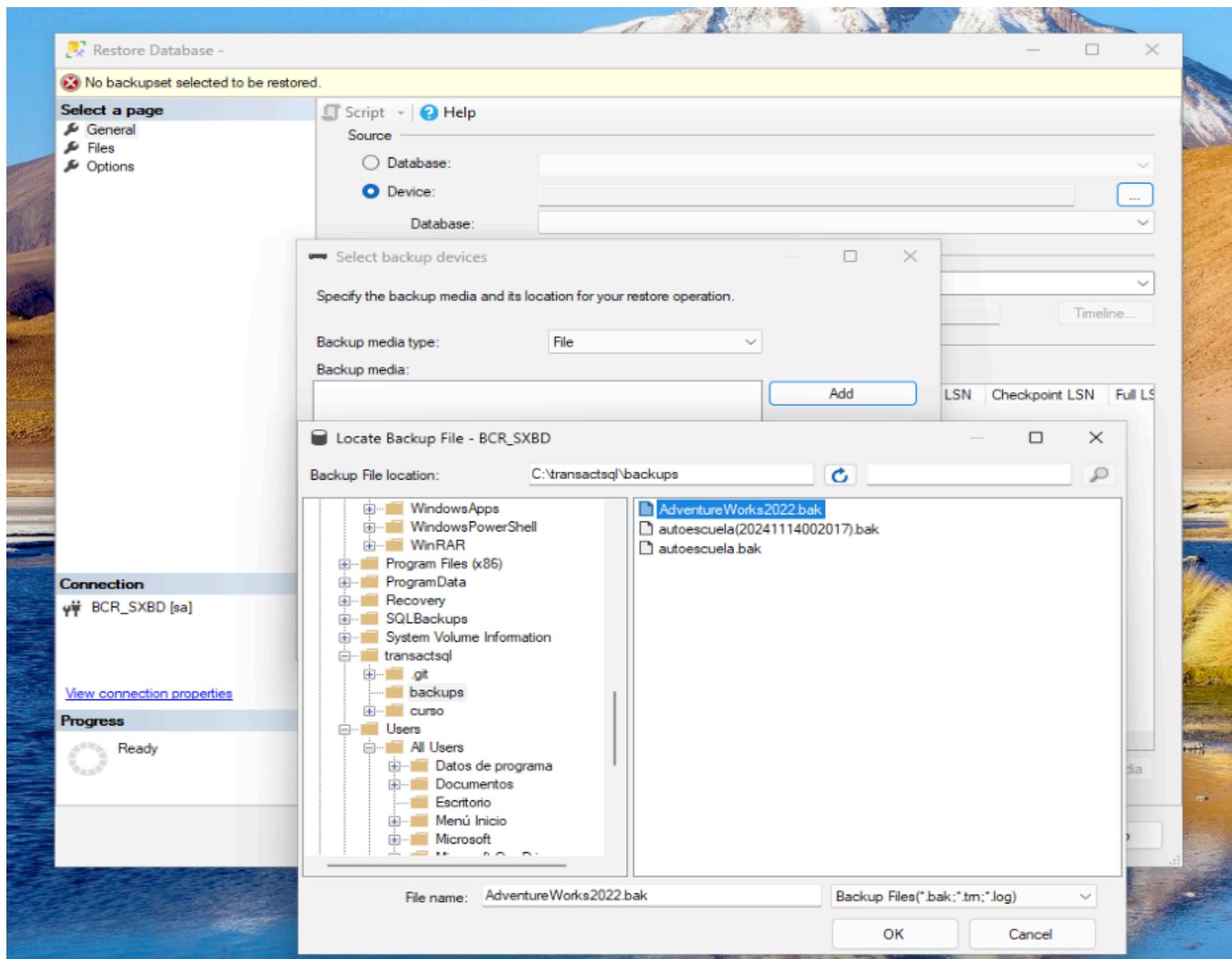
Instalación base de datos de ejemplo

Existen muchas formas de instalar las bases de ejemplo, vamos a explicar varias de ellas.

Restore Backup desde SSMS

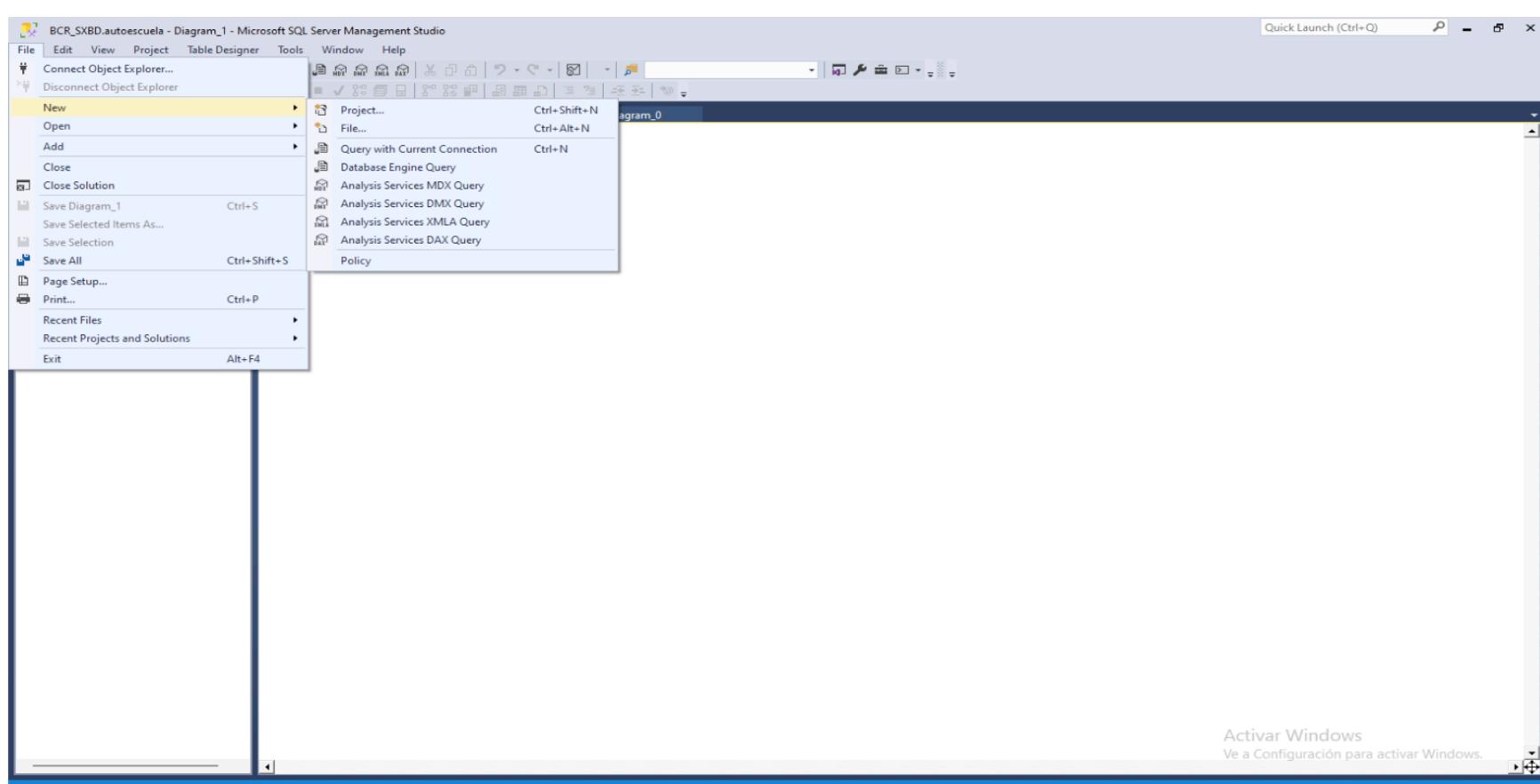
Comenzamos por realizar un backup directamente desde SSMS, una vez hayamos descargado el archivo .bak, vamos a "Databases" en la barra de navegación lateral, presionamos con el botón derecho y seleccionamos "Restore backup", seleccionamos "Device", buscamos nuestro archivo y presionamos en "OK" para restaurar.





Restore desde script

Otra opción es ejecutar el script, del mismo modo que hacemos al pasar de Datamodeler a SSMS. Para ello podemos descargar el script desde la web oficial y una vez lo tengamos, ejecutarlo desde SSMS. Para ello en SSMS presionamos en “File→Open→File...”, seleccionamos el archivo que hemos descargado con la query y una vez abierto lo ejecutamos



```

1  /*
2   * File: instawdb.sql
3   *
4   * Summary: Creates the AdventureWorks sample database. Run this on
5   * any version of SQL Server (2008R2 or later) to get AdventureWorks for your
6   * current version.
7   *
8   * Date: October 26, 2017
9   * Updated: October 26, 2017
10  *
11  * This file is part of the Microsoft SQL Server Code Samples.
12  *
13  * Copyright (C) Microsoft Corporation. All rights reserved.
14  *
15  * This source code is intended only as a supplement to Microsoft
16  * Development Tools and/or on-line documentation. See these other
17  * materials for detailed information regarding Microsoft code samples.
18  *
19  * All data in this database is fictitious.
20  *
21  * THIS CODE AND INFORMATION ARE PROVIDED "AS IS" WITHOUT WARRANTY OF ANY
22  * KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE
23  * IMPLIED WARRANTIES OF MERCHANTABILITY AND/OR FITNESS FOR A
24  * PARTICULAR PURPOSE.
25  */
26  /*
27  * HOW TO RUN THIS SCRIPT:
28  *
29  * 1. Enable full-text search on your SQL Server instance.
30  *
31  * 2. Open the script inside SQL Server Management Studio and enable SQLCMD mode.
32  *
33  * 3. Copy this script and the install files to C:\Samples\Adventureworks, or
34  *    set the following environment variable to your own data path.
35  *
36  * 4. Append the SQL Server version number to database name if you want to
37  *    differentiate it from other installs of Adventureworks.
38  */
39 :setvar SqlSamplesSourceDataPath "C:\Samples\Adventureworks"
40 /**
41 */
42 :setvar DatabaseName "AdventureWorks"

```

Restore desde Powershell

Ya que hemos dado los principales comandos de Powershell, vamos aprovechar para repasar como hacer un restore y así instalar de paso una de las bases de datos de ejemplo. Accedemos a Powershell y ejecutamos:

```
Restore-Sqldatabase -Serverinstance "localhost" -Database Adventureworks -Backupfile "C:\transactsql\backups\Adventureworks.bak"
```

*Debemos acordarnos de hacer un “Refresh” en la barra de navegación para que nos aparezcan las bases de datos que instalaremos.

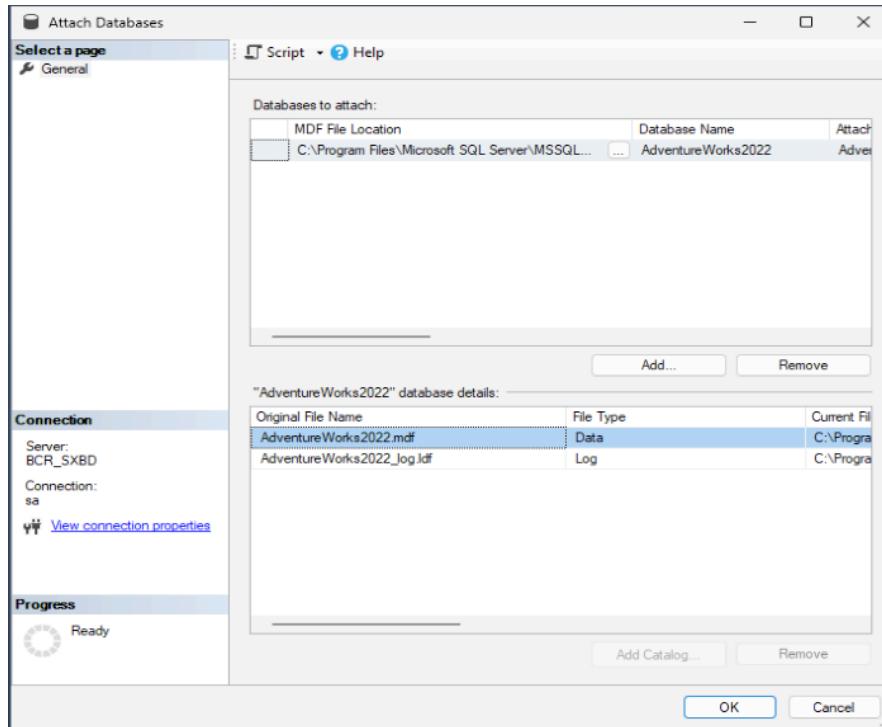
También estaría la opción de hacerlo usando el comando `Invoke-Sqlcmd` lo que nos permite ejecutar instancias de SQL en Powershell directamente:

```
Invoke-Sqlcmd -Serverinstance localhost -TrustServerCertificate -Query "Restore database AdventureWorks From 'C:\transactsql\backups\Adventureworks.bak'"
```

Restore con attach

Para instalar una base de datos usando ATTACH, usaremos los archivos .mdf y .ldf. En el panel de navegación lateral, presionamos con el botón derecho encima de databases y luego en attach. Aquí buscaremos nuestros archivos .mdf y .ldf y presionaremos aceptar para finalizar.

A veces este método puede dar errores, uno de los más comunes es que SSMS no tenga permisos suficientes para acceder a la carpeta donde se encuentran los archivos.



6.- Bases de datos contenidas

Son un tipo especial de bases de datos en sistemas como Microsoft SQL Server, diseñadas para ser lo más independientes posible del servidor que las aloja. Esto significa que su configuración y autenticación están almacenadas completamente dentro de la propia base de datos, en lugar de depender del servidor SQL.

Esto es ideal para bases de datos que deben ser migradas con frecuencia, además no es necesario mantener configuraciones a nivel servidor específicas para una base de datos.

Lo primero que haremos para crear/usuarios bases de datos contenidas es activar las opciones avanzadas y las bases de datos contenidas en SSMS.

```
-- Activamos opciones avanzadas
EXEC sp_configure 'show advanced options' 1
GO

-- Activamos BD contenidas
EXEC sp_configure 'contained database authentication', 1
GO
```

A la hora de crear una base de datos contenida, la única diferencia es que añadiremos los siguientes parámetros:

```
-- Controlamos la existencia
DROP DATABASE IF EXISTS Contenida
GO

-- Creamos la base de datos
CREATE DATABASE Contenida
CONTAINMENT=PARTIAL
GO
```

Recordar que además de crear bases de datos contenidas, también es posible convertir una base de datos que ya tengamos creada a base de datos contenida, para ello simplemente activamos las opciones avanzadas igual que hemos hecho para crear bases de datos contenidas pero en vez de usar "CREATE", ahora usaremos "ALTER" para modificar la que ya tenemos creada.

```
-- Activamos opciones avanzadas
EXEC sp_configure 'show advanced options' 1
GO

-- Activamos BD contenidas
EXEC sp_configure 'contained database authentication', 1
GO

-- Modificamos la base de datos para convertirla en contenida
```

```
ALTER DATABASE [NombreDeLaBaseDeDatos]
SET CONTAINMENT = PARTIAL;
```

Debemos tener en cuenta que debemos migrar también los usuarios y configuraciones de la base de datos, así como tener en cuenta las dependencias externas. Para migrar los usuarios, simplemente los crearemos de nuevo en la base de datos contenida y le asignaremos los permisos.

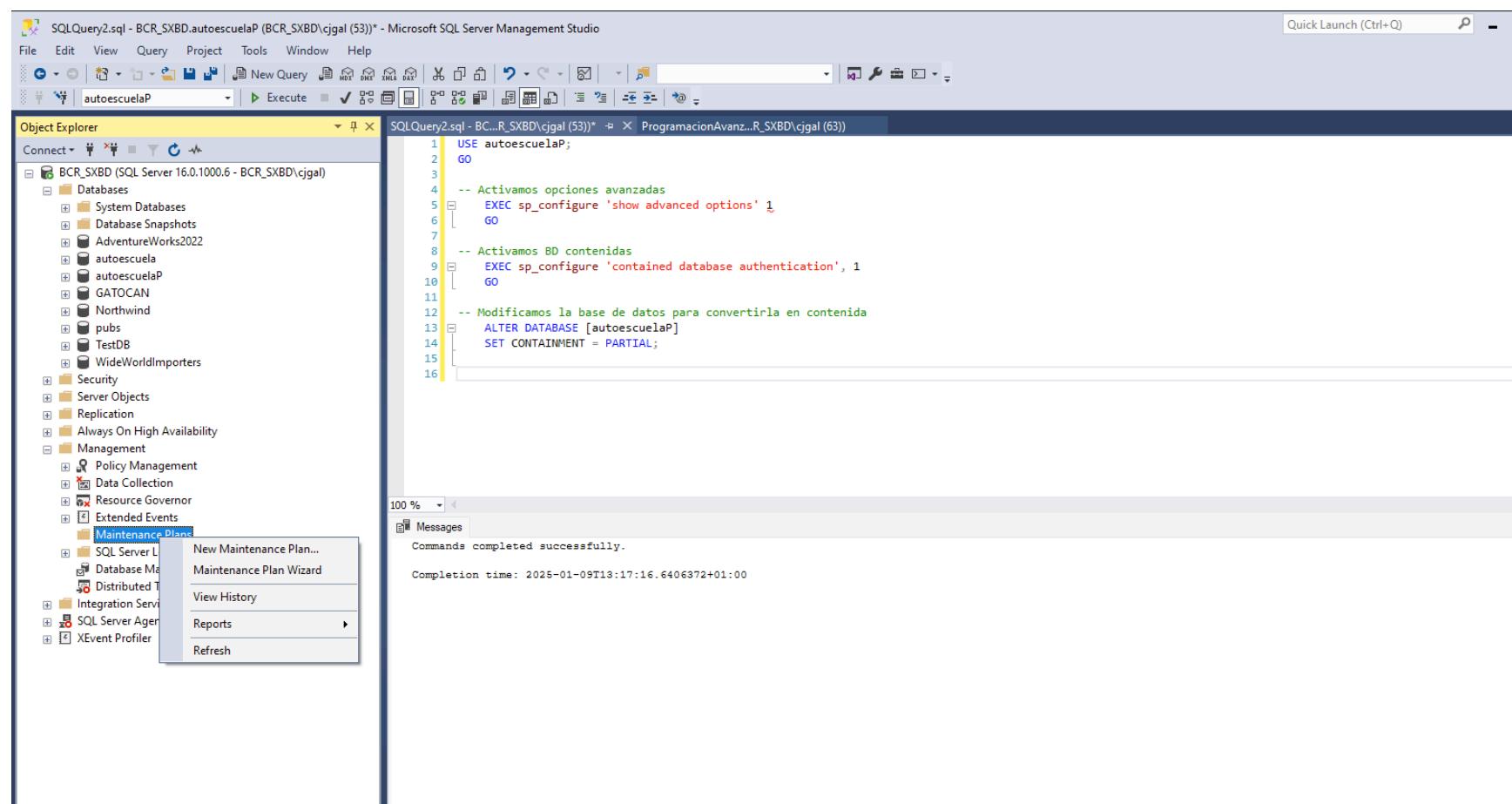
Podemos comprobar si nuestra base de datos figura como contenida con:

```
SELECT name, containment
FROM sys.databases
WHERE name = 'autoescuela';
```

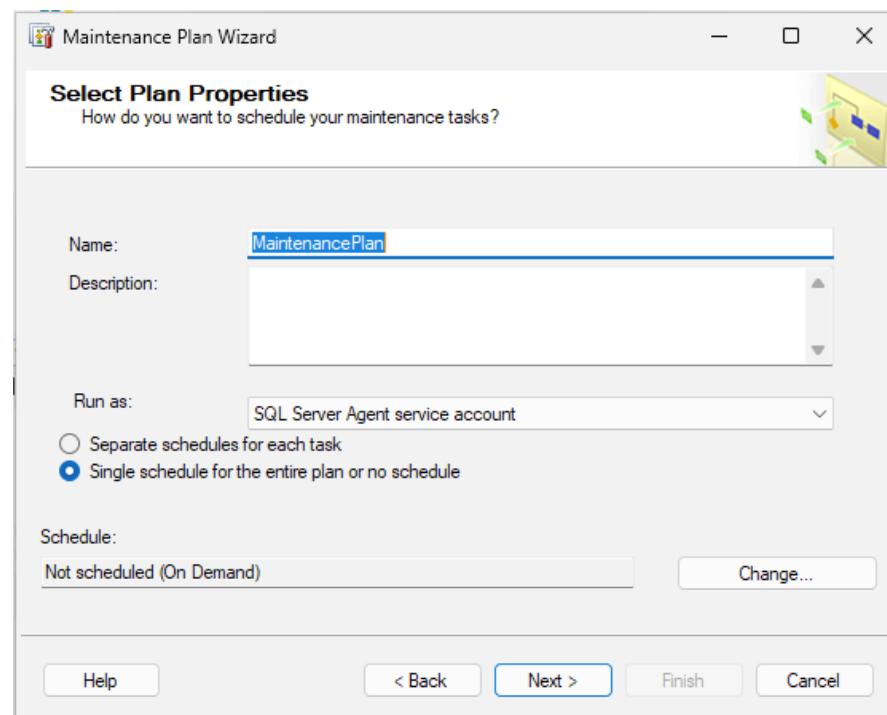
7.- Planes de mantenimiento

El plan de mantenimiento es una herramienta que permite automatizar tareas administrativas rutinarias en una base de datos. Estas tareas están diseñadas para garantizar el correcto funcionamiento, rendimiento y fiabilidad del servidor SQL y sus bases de datos. Antes de nada debemos tener en cuenta que debe estar arrancado el **SQL Agent**.

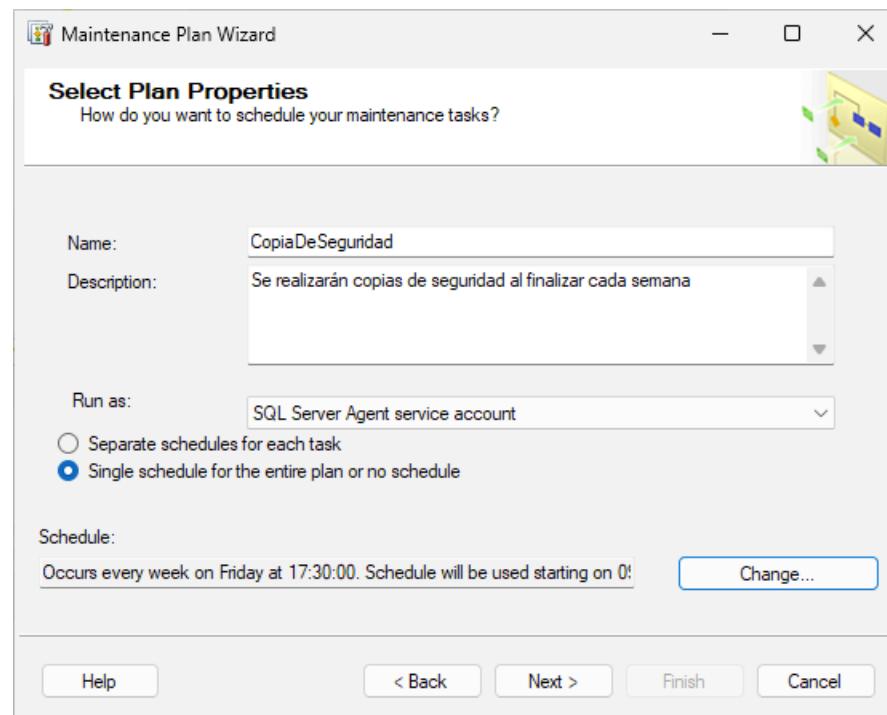
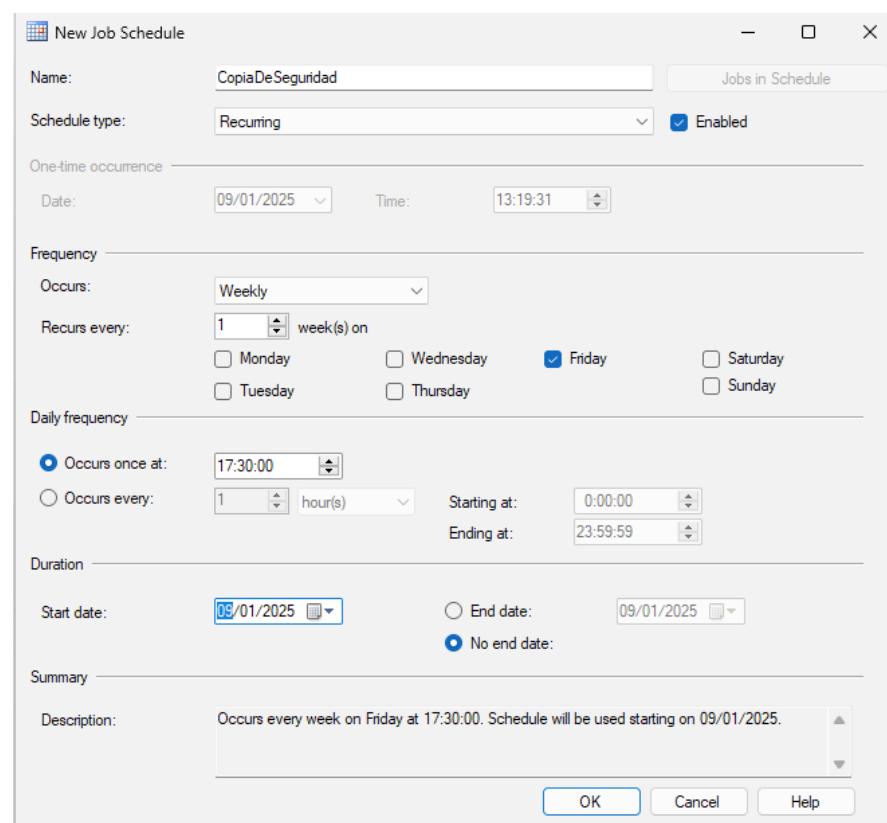
En la barra de navegación lateral vamos a “Management→Maintenance Plans” pulsamos botón derecho y “Maintenance Wizard”, se arrancará directamente el asistente.



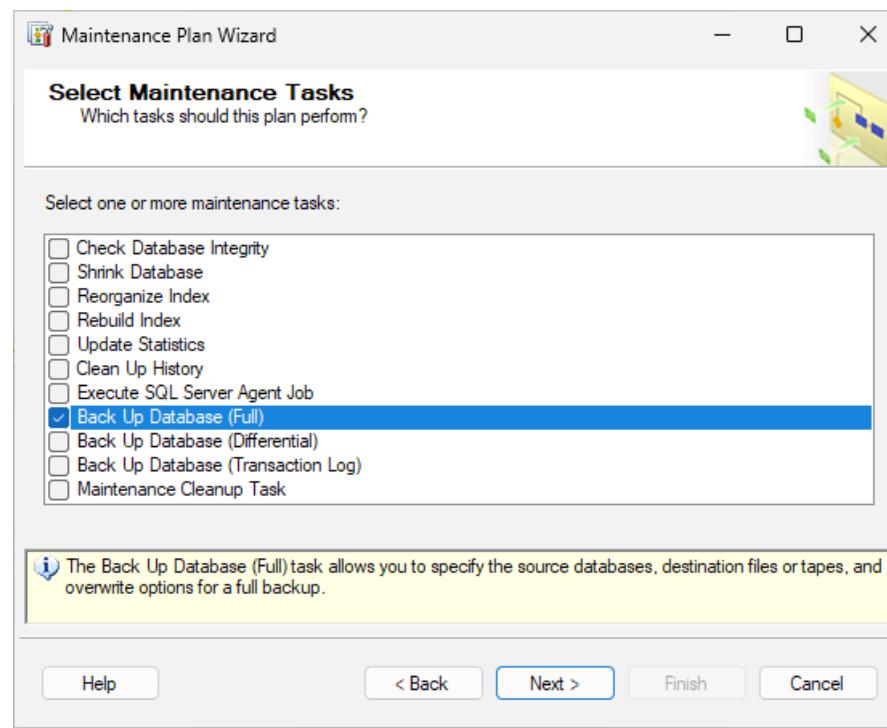
Le daremos un nombre al plan de mantenimiento y aunque se puede crear bajo demanda, en este caso vamos a crear una programación que es el caso más complejo.



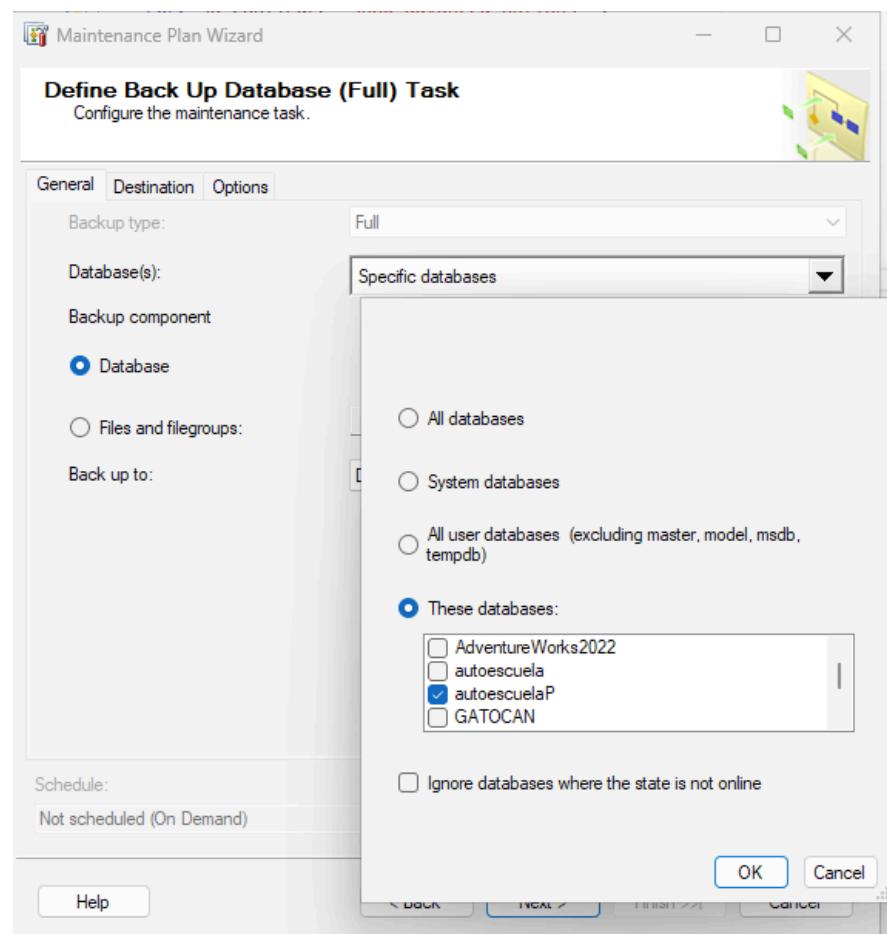
Al pinchar en "change" se nos despliegan las opciones de la programación, seleccionaremos recurrente y lo programaremos para que se ejecute todos los viernes al finalizar la jornada laboral, de esta manera todas las semanas al finalizar el viernes se ejecutará una copia de seguridad completa.



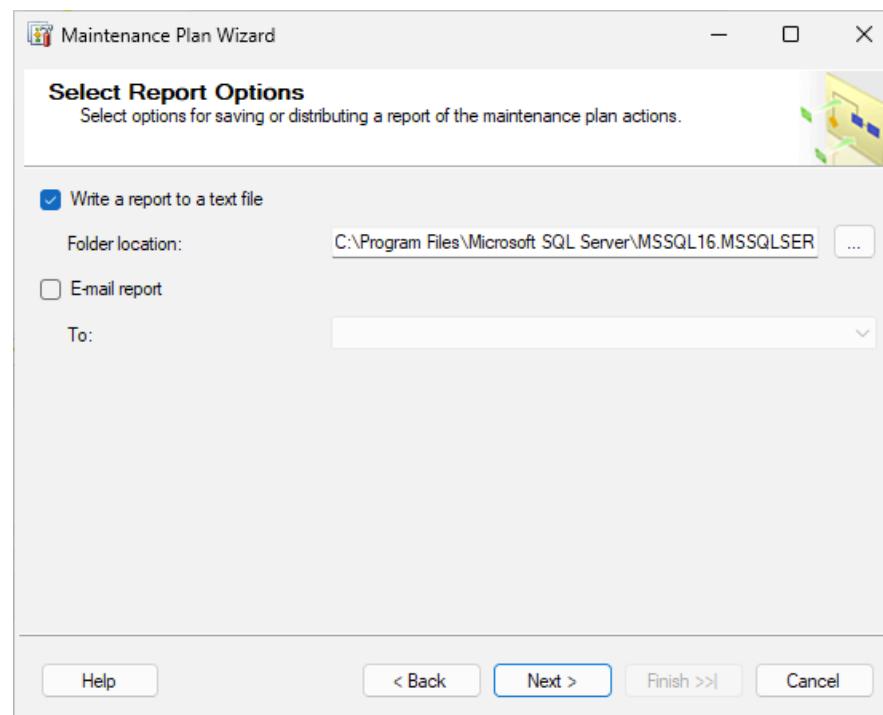
Luego seleccionaremos las tareas a realizar, en este caso será una copia de seguridad completa.



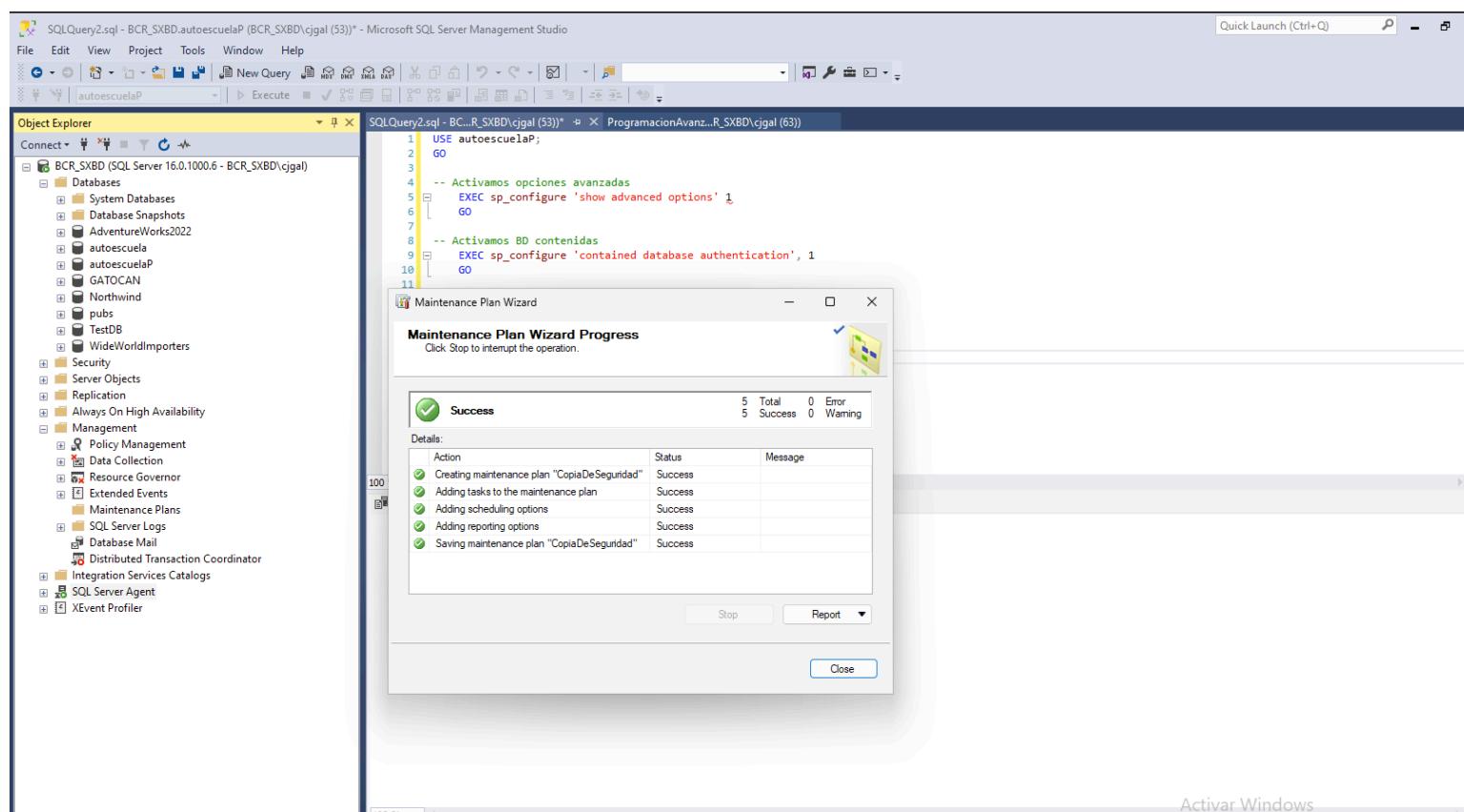
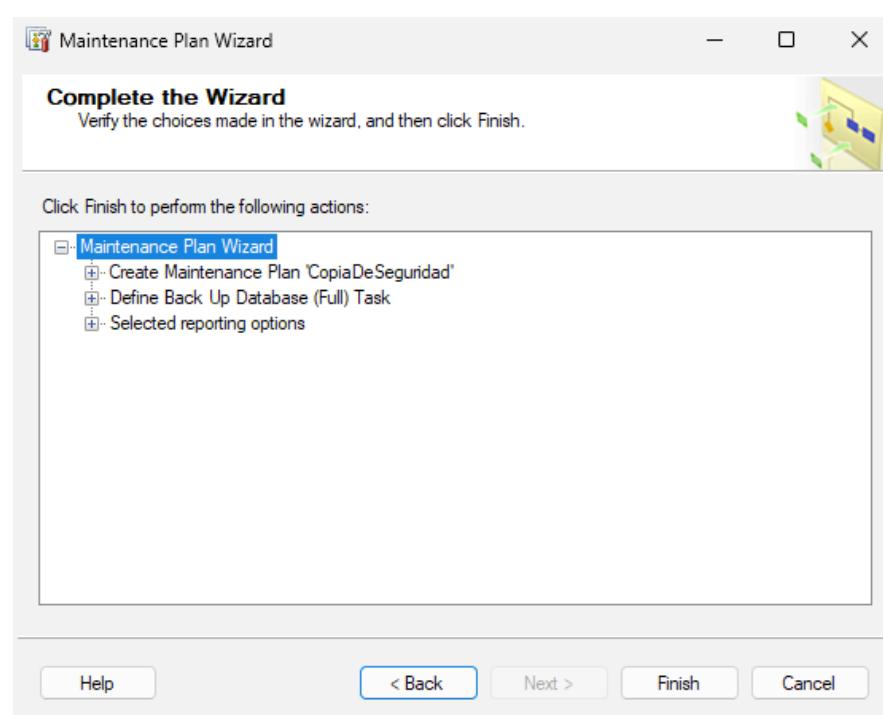
También seleccionaremos las bases de datos a las que queremos que afecte el plan de mantenimiento (en este caso la copia de seguridad). Seleccionamos la base de datos de la autoescuela (la copia que tenemos de la principal para trabajar sobre ella).

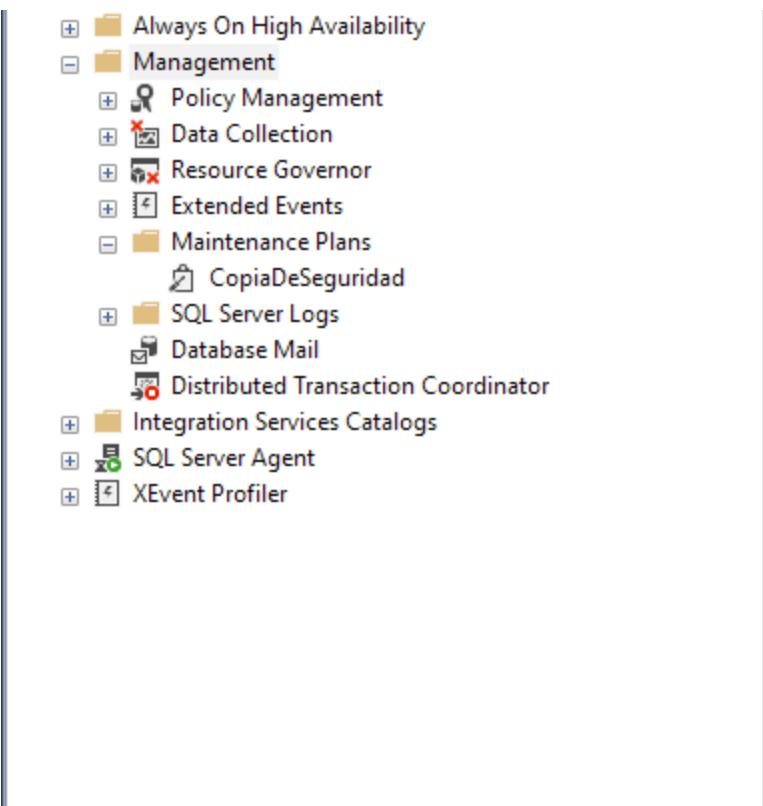


Seleccionamos donde queremos que se guarde el reporte de la tarea.



Ya solo queda finalizar y quedará configurado nuestro plan de mantenimiento.





8. BLOB (Binary Large Object)

Es un tipo de dato que se utiliza en bases de datos para almacenar grandes volúmenes de información binaria, como archivos, imágenes, videos, audio o cualquier otro tipo de datos no estructurados.

Opciones para BLOBS (Binary Large Objects) en SQL Server

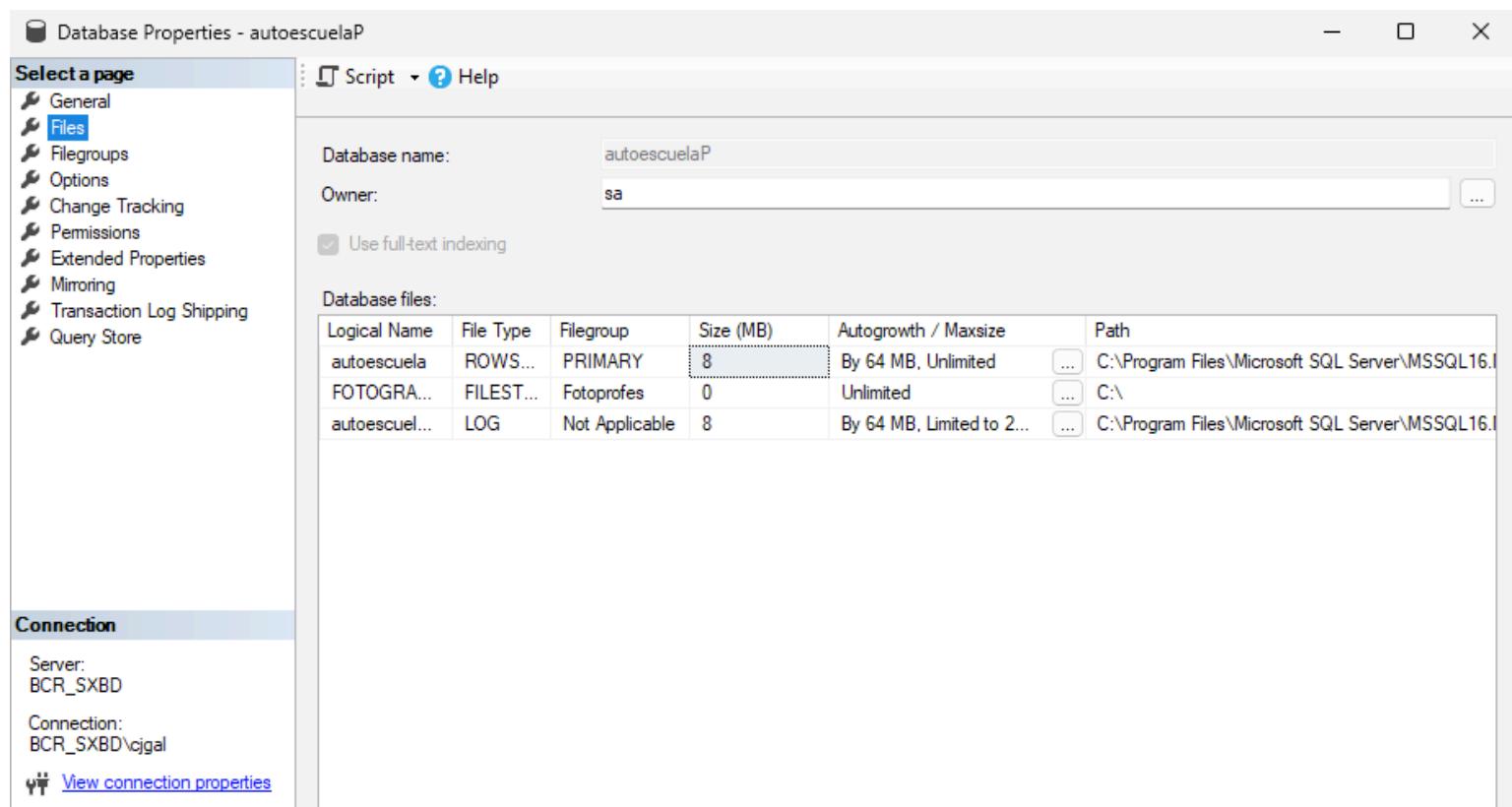
Tecnología	Descripción	Ejemplo	Cuándo usar
NTFS (Sistema de archivos)	Almacenar archivos directamente en el sistema operativo, con rutas guardadas en la base de datos.	Guardar imágenes en <code>C:\Archivos</code> y almacenar la ruta <code>C:\Archivos\foto.jpg</code> en una columna VARCHAR.	- Archivos muy grandes (>2 GB).- Acceso frecuente fuera de SQL (ej: aplicaciones externas).
FILESTREAM	Combina NTFS con SQL Server: los BLOBS se almacenan en archivos del sistema, pero se gestionan transaccionalmente desde la base de datos.	<code>sql CREATE TABLE Documentos (ID UNIQUEIDENTIFIER ROWGUIDCOL, Archivo VARBINARY(MAX) FILESTREAM);</code>	- BLOBS entre 1 MB y 2 GB.- Necesidad de transacciones ACID.- Acceso mixto (SQL + sistema de archivos).
FILETABLE	Extensión de FILESTREAM que permite acceder a los archivos como si fueran una tabla, con jerarquía de carpetas.	<code>sql CREATE FILETABLE DocumentosFileTable WITH (FILETABLE_DIRECTORY = 'Documentos');</code>	- Sistemas legacy que usan carpetas.- Aplicaciones que requieren estructura de archivos jerárquica.

FILEGROUPS

Crear un **filegroup** en SQL Server permite organizar y distribuir los archivos de la base de datos en diferentes ubicaciones físicas, lo que mejora la administración del almacenamiento, optimiza el rendimiento, y facilita el respaldo y la recuperación de datos. Es especialmente útil cuando se trabaja con grandes volúmenes de datos o cuando se habilitan características como **FILESTREAM** o **particionamiento**.

En este caso crearemos un filegroup para organizar los archivos de foto que añadiremos usando FILESTREAM.

```
-- Creamos el filegroup
ALTER DATABASE [autoescuelap]
ADD FILEGROUP [Fotoprofes] CONTAINS FILESTREAM;
GO
-- añadimos el fichero donde vamos a almacenar el filegroup (las imágenes)
ALTER DATABASE [autoescuelap]
ADD FILE (NAME='FOTOGRAFIAS',
FILENAME = 'C:\FOTOGRAFIAS')
TO FILEGROUP [Fotoprofes]
GO
```



NTFS (Carpetas del sistema operativo)

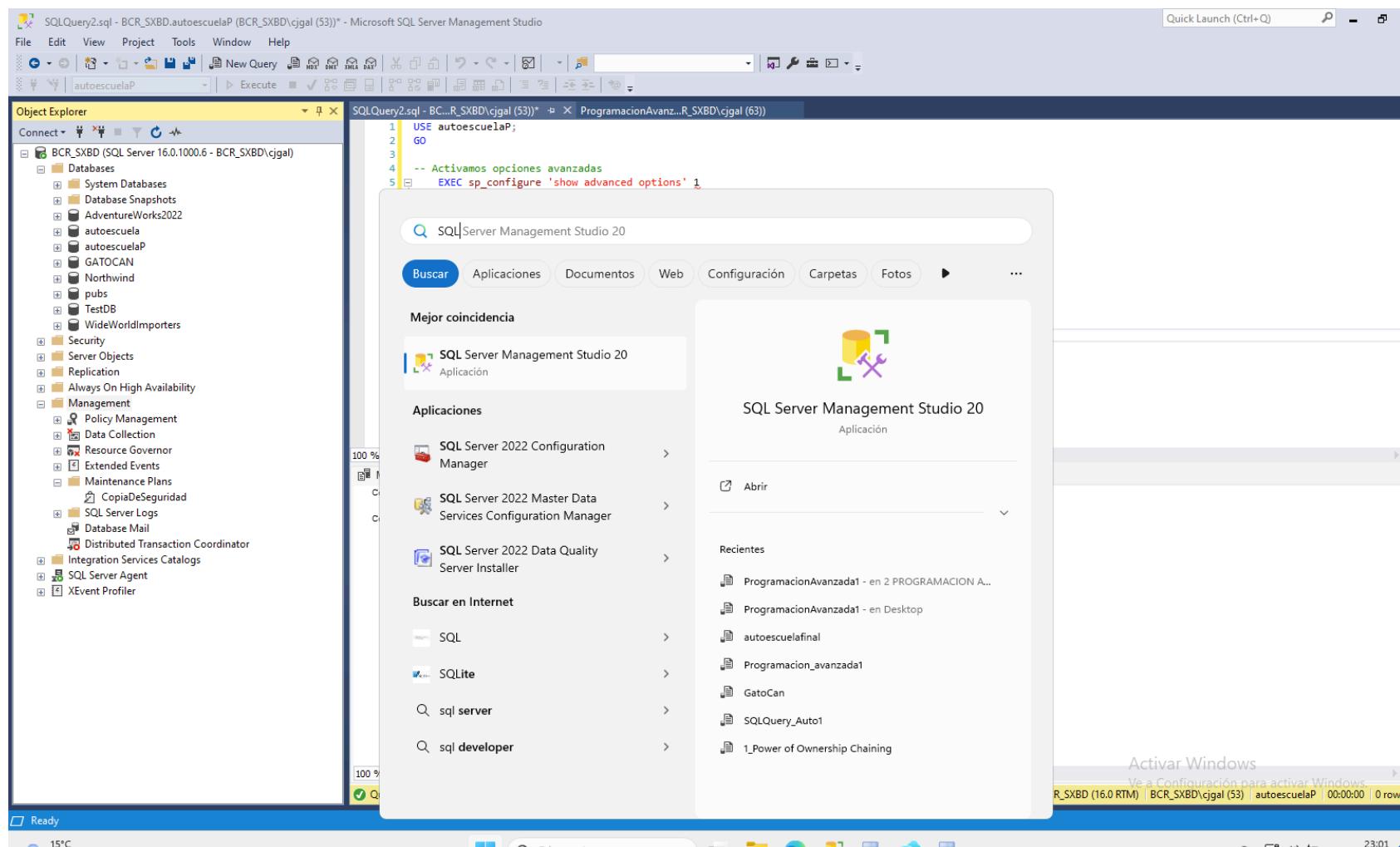
Lo primero que haremos será activar en el archivo de configuración el filestream access.

Tenemos las siguientes opciones: "EXEC sp_configure filestream_access_level" añadiendo las siguientes posibilidades:

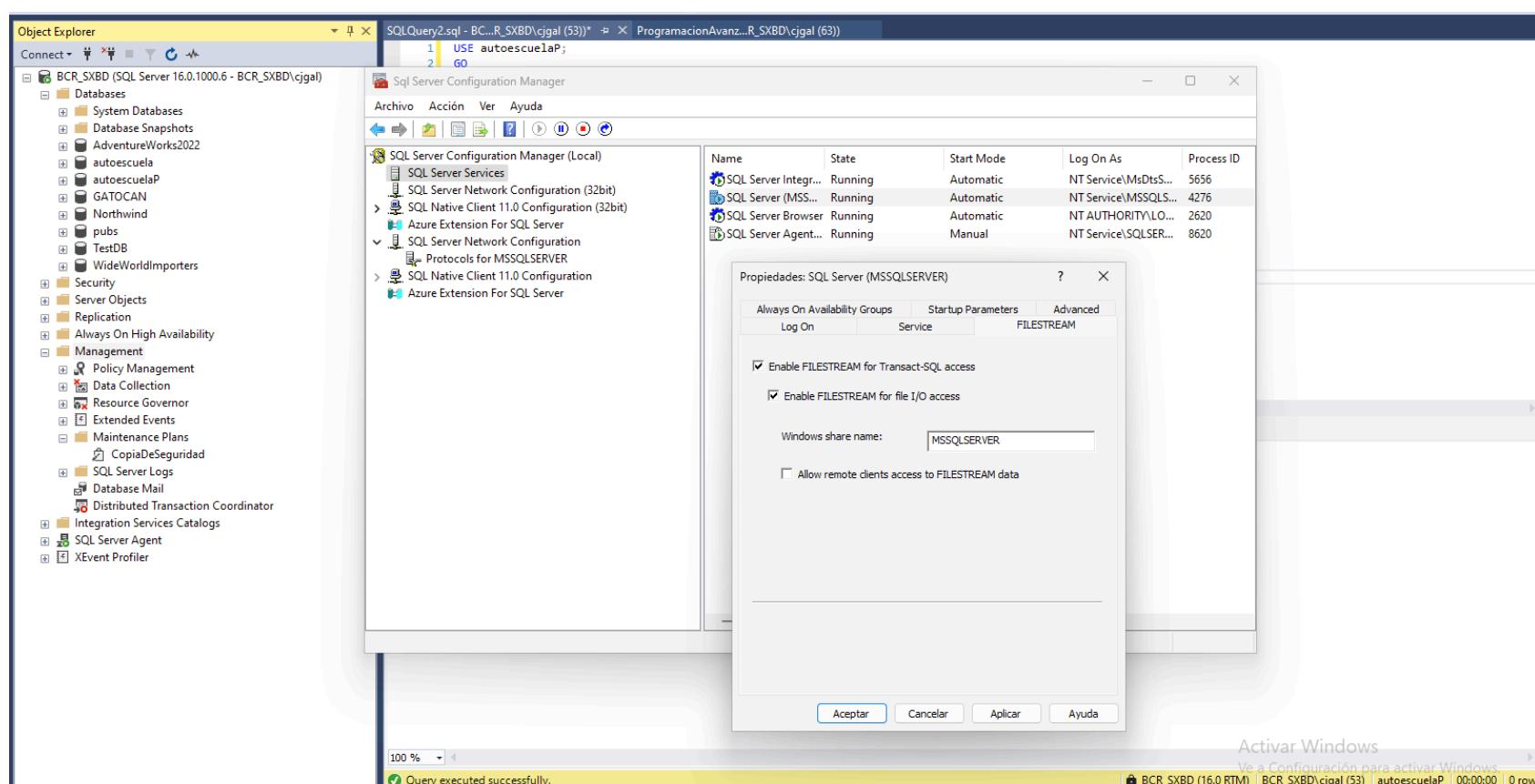
- 0 = Deshabilitado. (valor por defecto).
- 1 = Habilitado solo para acceso de T-SQL.
- 2 = Habilitado para T-SQL y acceso local al sistema de ficheros.
- 3 = Habilitado para T-SQL, acceso local y remoto al sistema de ficheros.
-

```
EXEC sp_configure 'filestream_access_level', 2;
RECONFIGURE;
GO
```

Para configurarlo, aunque podemos hacerlo mediante comandos, es mejor hacerlo desde el SQLServer configuration porque por comandos a veces da fallos. De manera que iremos a cortana y escribiremos SQL server y lo ejecutaremos.



Una vez dentro, buscaremos “SQL Server Services→SQL Server (MSSQLSERVER)”, iremos a la pestaña FILESTREAM y marcaremos las dos casillas de FILESTREAM.



*Reiniciar el servicio para que coja la nueva configuración.

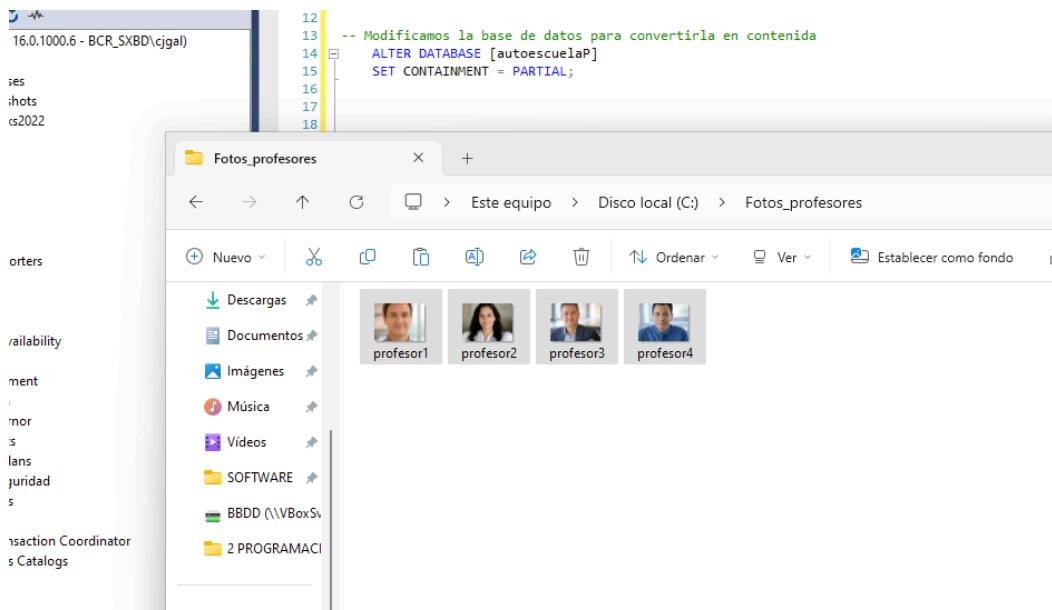
Debemos recordar que para guardar este tipo de archivos, lo haremos en variables tipo VARBINARY(MAX).

```
EXECUTE sp_configure 'show advanced options',1;
```

Debemos crear una carpeta para almacenar los archivos, podemos hacerlo con 'xp_cmdshell' pero debemos tener mucho cuidado porque con este comando, se permite ejecutar cualquier sentencia de la shell de windows y eso implica que cualquier persona que acceda a la base de datos podría ejecutar un format por ejemplo.

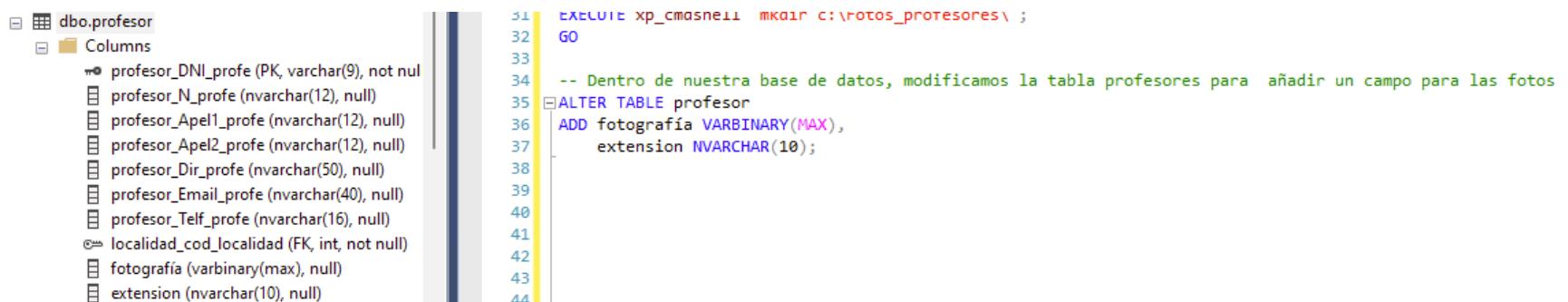
```
EXECUTE autoescuelap xp_cmdshell 'mkdir c:\Fotos_profesores'
GO
```

Por este motivo, crearemos la carpeta por fuera de SSMS y añadiremos las fotos, en este caso serán las fotos de los profesores de la autoescuela.



Añadimos la columna “fotografía” a la tabla profesor, donde almacenaremos las fotografías.

```
ALTER TABLE profesor  
ADD fotografía VARBINARY(MAX),  
extension NVARCHAR(10);  
GO;
```



Ahora insertamos los datos:

```
-- Insertamos fotografías  
UPDATE profesor  
SET fotografía = (SELECT BULKCOLUMN  
FROM OPENROWSET(BULK N'C:\Fotos_profesores\profesor1.JPG', SINGLE_BLOB) AS archivo),  
extension = 'JPG'  
WHERE profesor_DNI_profe = '10987654R';  
GO  
  
UPDATE profesor  
SET fotografía = (SELECT BULKCOLUMN  
FROM OPENROWSET(BULK N'C:\Fotos_profesores\profesor2.JPG', SINGLE_BLOB) AS archivo),  
extension = 'JPG'  
WHERE profesor_DNI_profe = '09876543Q';  
GO  
  
UPDATE profesor  
SET fotografía = (SELECT BULKCOLUMN  
FROM OPENROWSET(BULK N'C:\Fotos_profesores\profesor3.JPG', SINGLE_BLOB) AS archivo),  
extension = 'JPG'  
WHERE profesor_DNI_profe = '32109876T';  
GO  
-- Comprobamos las inserciones
```

```
SELECT profesor_N_profe, profesor_Apel1_profe, profesor_Apel2_profe, fotografía, extension FROM profesor;
GO
```

```
50
39 UPDATE profesor
40 SET fotografía = (SELECT BULKCOLUMN
41 FROM OPENROWSET(BULK N'C:\Fotos_profesores\profesor1.JPG', SINGLE_BLOB) AS archivo),
42 extension = 'JPG'
43 WHERE profesor_DNI_profe = '10987654R';
44 GO
45
46 UPDATE profesor
47 SET fotografía = (SELECT BULKCOLUMN
48 FROM OPENROWSET(BULK N'C:\Fotos_profesores\profesor2.JPG', SINGLE_BLOB) AS archivo),
49 extension = 'JPG'
50 WHERE profesor_DNI_profe = '09876543Q';
51 GO
52
53 UPDATE profesor
54 SET fotografía = (SELECT BULKCOLUMN
55 FROM OPENROWSET(BULK N'C:\Fotos_profesores\profesor3.JPG', SINGLE_BLOB) AS archivo),
56 extension = 'JPG'
57 WHERE profesor_DNI_profe = '32109876T';
58 GO
59
60 SELECT profesor_N_profe, profesor_Apel1_profe, profesor_Apel2_profe, fotografía, extension FROM profesor;
61 GO
```

	profesor_N_profe	profesor_Apel1_profe	profesor_Apel2_profe	fotografía	extension
1	Carmen	Moreno	Alonso	0xFFD8FFE000104A46494600010200000100010000FFFE029...	JPG
2	Alberto	Jiménez	Rodríguez	0xFFD8FFE000104A46494600010200000100010000FFFE028...	JPG
3	Isabel	Pérez	Martín	NULL	NULL
4	Miguel	Díaz	Sánchez	0xFFD8FFE000104A46494600010200000100010000FFFE025...	JPG
5	Elena	González	Ruiz	NULL	NULL
6	Irene	Fernández	López	NULL	NULL

FILESTREAM

Lo primero que haremos será activar filestream (si no lo hemos hecho en el configuration manager)

```
-- activamos FILESTREAM (si no lo hemos hecho en configuration manager)
EXEC sp_configure filestream_access_level, 2
RECONFIGURE
GO
```

Creamos un FILEGROUP que contenga Filestream

```
ALTER DATABASE autoescuelaP
ADD FILEGROUP fg_imagenes CONTAINS FILESTREAM
GO
```

Asociamos un archivo físico al FILEGROUP

```
ALTER DATABASE AutoescuelaP
ADD FILE (NAME = 'Imagenes_FS',
FILENAME = 'C:\DATA\Imagenes_FS')
TO FILEGROUP fg_imagenes
GO
```

Comprobamos:

	name	data_space_id	type	type_desc	is_default	is_system	filegroup_guid
1	fg_imagenes	19	FD	FILESTREAM_DATA_FILEGROUP	0	0	261F3497-46AC-46D1-B1FA-C5511DF5F6C1
2	AutoescuelaP_mod	18	FX	MEMORY_OPTIMIZED_DATA_FILEGROUP	1	0	3C1E257E-CFF8-4C58-9064-006584478978
3	Madrid	17	FG	ROWS_FILEGROUP	0	0	6A6ACE24-FC26-4382-8837-A9DF5C9FB509
4	Pontevedra	16	FG	ROWS_FILEGROUP	0	0	8F3600AF-7ED6-4713-8143-9B6F14213D5E
5	Orense	15	FG	ROWS_FILEGROUP	0	0	15710CBE-B8AD-415B-952F-23CAE152E9FC

Creamos la tabla:

****IMPORTANTE** añadir ID2 y acordarse de definir el campo del archivo como **FILESTREAM**

```
CREATE TABLE [dbo].[profesor_FS](
    [ID2] UNIQUEIDENTIFIER ROWGUIDCOL NOT NULL UNIQUE,
    [profesor_DNI_profe] [varchar](9) NOT NULL PRIMARY KEY,
    [profesor_N_profe] [nvarchar](12) NULL,
    [profesor_Apel1_profe] [nvarchar](12) NULL,
    [profesor_Apel2_profe] [nvarchar](12) NULL,
    [profesor_Email_profe] [nvarchar](40) NULL,
    [profesor_Telf_profe] [nvarchar](16) NULL,
    [localidad_cod_localidad] [int] NOT NULL,
    [fotografía] [varbinary](max) FILESTREAM,
    [extension] [nvarchar](10) NULL)
GO
```

Añadimos los registros a la tabla:

```
INSERT INTO profesor_FS (ID2,profesor_DNI_profe,profesor_N_profe,profesor_Apel1_profe,localidad_cod_localid
SELECT NEWID(),'32840700N','Borja','Costa',10, BULKCOLUMN,'JPG'
FROM OPENROWSET(BULK 'C:\Fotos_profesores\Profesor1.JPG', SINGLE_BLOB) AS FOTO
GO
```

```
INSERT INTO profesor_FS (ID2,profesor_DNI_profe,profesor_N_profe,profesor_Apel1_profe,localidad_cod_localid
SELECT NEWID(),'13855702H','Jose','Crego',10, BULKCOLUMN,'JPG'
FROM OPENROWSET(BULK 'C:\Fotos_profesores\Profesor2.JPG', SINGLE_BLOB) AS FOTO
```

Otro ejemplo FILESTREAM

Continuamos con otro ejemplo de FILESTREAM, vamos a añadir los campos a la tabla profesores. En este caso, como ya tenemos las relaciones establecidas y los datos añadidos, hemos optado por crear una nueva tabla para evitar conflictos, ya que si intentamos añadir un campo not null a una tabla que ya tiene relaciones y datos, nos dará el error de que no pueden existir campos nulos.

```
-- Creamos el filegroup
ALTER DATABASE [autoescuelap]
ADD FILEGROUP [Fotoprofes] CONTAINS FILESTREAM;
GO
-- añadimos el fichero donde vamos a almacenar el filegroup (las imágenes)
ALTER DATABASE [autoescuelap]
ADD FILE (NAME='FOTOGRAFIAS',
FILENAME = 'C:\FOTOGRAFIAS')
```

```
TO FILEGROUP [Fotoprofes]
```

```
GO
```

Usaremos para este ejemplo la tabla Orla.

```
CREATE TABLE Orla
(ID INT IDENTITY,
ID2 UNIQUEIDENTIFIER ROWGUIDCOL NOT NULL UNIQUE,
nombre varchar(255),
fotografía VARBINARY(MAX) FILESTREAM,
extension CHAR(5)
)
GO
```

Ahora insertaremos los datos

```
INSERT INTO Orla (ID2,nombre,fotografía,extension)
SELECT NEWID(), 'Borja', BULKCOLUMN, 'JPG'
FROM OPENROWSET(BULK 'c:\FOTOGRAFIAS', SINGLE_BLOB) AS fotografia
GO

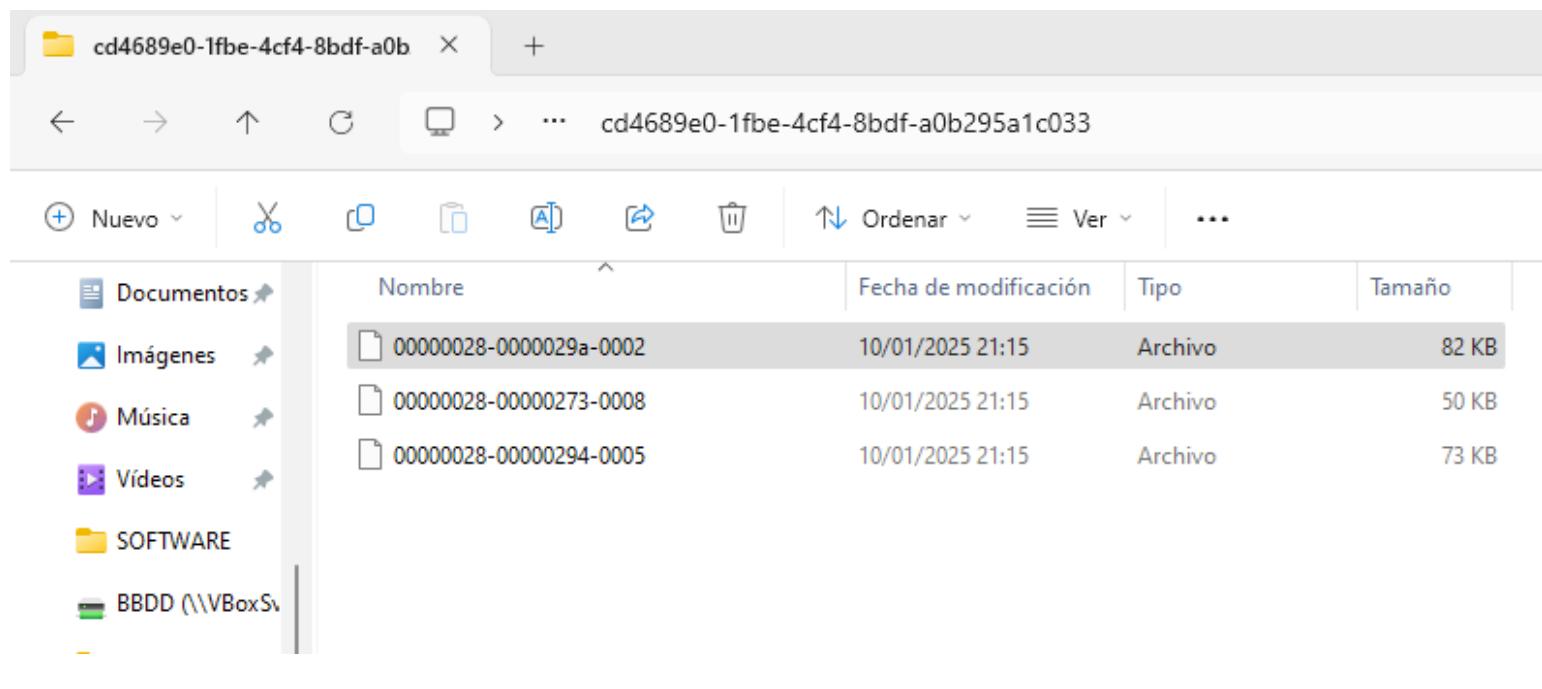
INSERT INTO Orla (ID2,nombre,fotografía,extension)
SELECT NEWID(), 'Marta', BULKCOLUMN, 'JPG'
FROM OPENROWSET(BULK 'c:\Fotos_profesores\profesor2.jpg', SINGLE_BLOB) AS fotografia
GO

INSERT INTO Orla (ID2,nombre,fotografía,extension)
SELECT NEWID(), 'Juanjo', BULKCOLUMN, 'JPG'
FROM OPENROWSET(BULK 'c:\Fotos_profesores\profesor3.jpg', SINGLE_BLOB) AS fotografia
GO

--Comprobamos
SELECT * FROM Orla;
GO
```

```
90 -- Insertamos los datos
91
92 INSERT INTO Orla (ID2,nombre,fotografía,extension)
93     SELECT NEWID(), 'Borja', BULKCOLUMN, 'JPG'
94     FROM OPENROWSET(BULK 'c:\Fotos_profesores\profesor1.jpg', SINGLE_BLOB) AS fotografia
95 GO
96
97
98 INSERT INTO Orla (ID2,nombre,fotografía,extension)
99     SELECT NEWID(), 'Marta', BULKCOLUMN, 'JPG'
100    FROM OPENROWSET(BULK 'c:\Fotos_profesores\profesor2.jpg', SINGLE_BLOB) AS fotografia
101   GO
102
103 INSERT INTO Orla (ID2,nombre,fotografía,extension)
104     SELECT NEWID(), 'Juanjo', BULKCOLUMN, 'JPG'
105     FROM OPENROWSET(BULK 'c:\Fotos_profesores\profesor3.jpg', SINGLE_BLOB) AS fotografia
106   GO
107
108 --Comprobamos
109 SELECT * FROM Orla;
110 GO
```

ID	ID2	nombre	fotografía	extension
1	C7131541-21E3-4E21-8C30-4F509A4FEC26	Borja	0xFFD8FFE000104A46494600010200000100010000FFFE028...	JPG
2	533C4CF0-4517-45A1-B238-338FFB4B28D9	Marta	0xFFD8FFE000104A46494600010200000100010000FFFE029...	JPG
3	C9AC8409-44A9-4492-A6AF-72A7935F38EE	Juanjo	0xFFD8FFE000104A46494600010200000100010000FFFE025...	JPG



FILETABLES

Podría decirse que FILETABLE es una especie de FILESTREAM mejorado, filetable permite acceder a los archivos desde fuera de SQL server, ya sea con el explorador de windows o bien con otras aplicaciones.

Lo primero que haremos para usar FILETABLE es activar FILESTREAM y adjudicar un directorio lógico como referencia para almacenar y gestionar las imágenes en el sistema de archivos.

```
--Activamos filestream
ALTER DATABASE [autoescuelap]
SET FILESTREAM (DIRECTORY_NAME = 'FotografiaStore')
WITH ROLLBACK IMMEDIATE
GO

ALTER DATABASE [autoescuelap]
SET FILESTREAM(NON_TRANSACTED_ACCESS = FULL,
DIRECTORY_NAME = 'FOTOGRAFIA')
WITH ROLLBACK IMMEDIATE
GO
```

Ahora crearemos una tabla para organizar los elementos que vamos a almacenar

```
--Creamos tablas FILETABLE
CREATE TABLE FotografiaStore AS FILETABLE
WITH
(
    FileTable_Directory = 'FotografiaStore',
    FileTable_Collate_Filename = database_default,
    FILETABLE_STREAMID_UNIQUE_CONSTRAINT_NAME=UQ_stream_id
);
GO
```

Ahora ya solo nos quedaría acceder a la carpeta que acabamos de crear y copiar en ella todas las imágenes que queramos insertar.

```

112 --FILETABLE
113 --Activamos filestream
114 ALTER DATABASE [autoescuelap]
115 SET FILESTREAM ( DIRECTORY_NAME = 'FOTOGRAFIA' )
116 WITH ROLLBACK IMMEDIATE
117 GO
118
119 --ALTER DATABASE [autoescuelap]
120     SET FILESTREAM(NON_TRANSACTED_ACCESS = FULL,
121     DIRECTORY_NAME = 'FOTOGRAFIA')
122 WITH ROLLBACK IMMEDIATE
123 GO
124
125 --Creamos tablas FILETABLE
126 CREATE TABLE FotografiaStore AS FILETABLE
127 WITH
128 (
    FileTable_Directory = 'FotografiaStore',
    FileTable_Collate_Filename = database_default,
    FILETABLE_STREAMID_UNIQUE_CONSTRAINT_NAME=UQ_stream_id
)
129
130
131
132 );
133 GO
134
135 -- Una vez hayamos copiado en la carpeta las imágenes que queramos, comprobamos:
136 SELECT * FROM [dbo].[FotografiaStore]
137 GO
138

```

Results

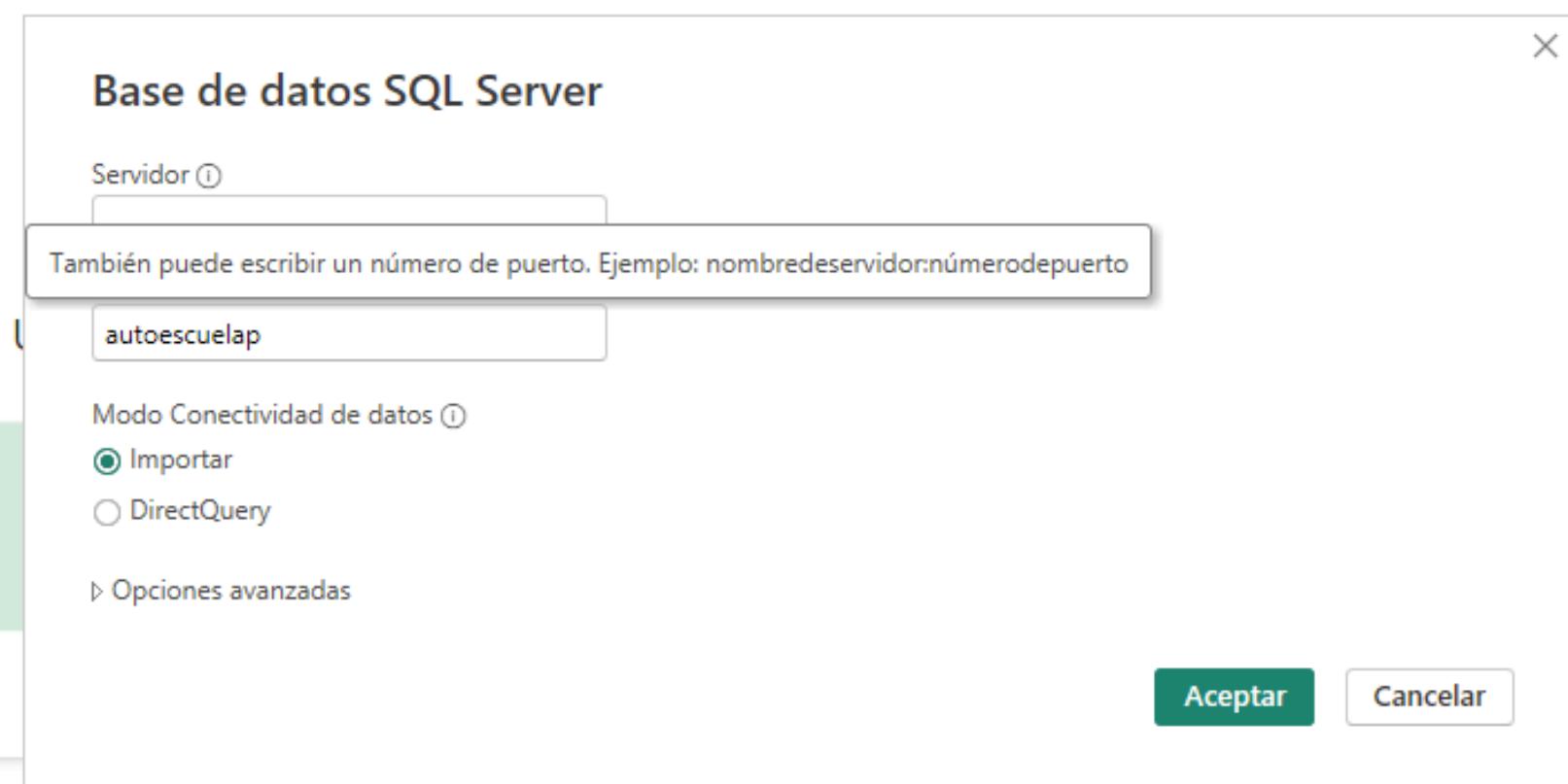
stream_id	file_stream	name	path_locator	parent_path_locator	file_type	cached_file_size
1	0F12FFB-A2CF-EF11-ABAB-0800270EC263	profesor1.jpg	0xCAF8F3D62497BB4FD253364FD02A1AFA8CA0142360	NULL	jpg	50274
2	1112FFB-A2CF-EF11-ABAB-0800270EC263	profesor2.jpg	0xFF65B1CD91C913AFD1511ABD7896F93F6C987320	NULL	jpg	74180
3	1312FFB-A2CF-EF11-ABAB-0800270EC263	profesor3.jpg	0xFFC383E57048BB2FC99156D531F89CFA468E2173A0	NULL	jpg	83189
4	1512FFB-A2CF-EF11-ABAB-0800270EC263	profesor4.jpg	0xFD9C420DFA1339EFC18F762A6D8398FA9C90144660	NULL	jpg	61720

POWERBI

Power BI es una herramienta de análisis de datos y visualización de negocios desarrollada por **Microsoft**. Está diseñada para ayudar a las empresas y a los profesionales a recopilar, transformar, analizar y visualizar datos en informes y paneles interactivos. Es una de las herramientas más populares para la **inteligencia empresarial (BI, Business Intelligence)** debido a su facilidad de uso y su integración con otros productos de Microsoft como Excel, Azure y Microsoft 365.

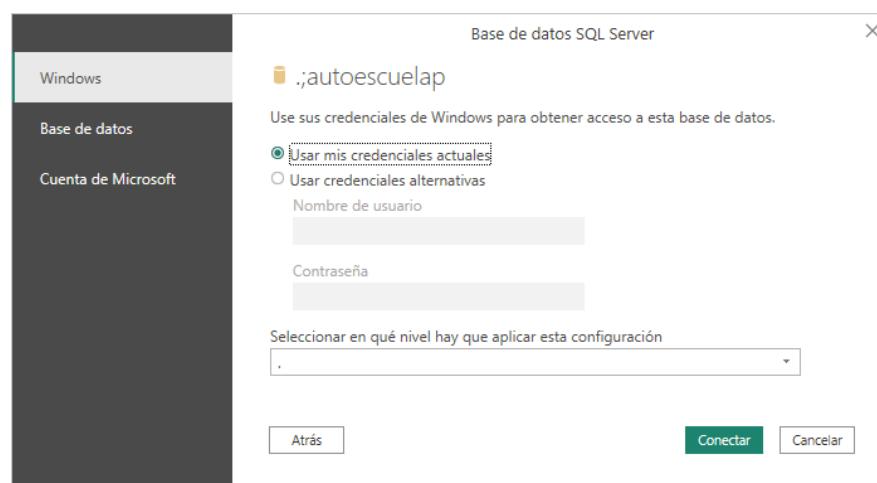
Lo primero que haremos será acceder a la web oficial de POWERBI, descargarlo e instalarlo. Una vez instalado, lo abrimos y nos encontramos con esta pantalla principal:

Seleccionamos SQL Server para añadir la conexión a nuestra base de datos. Añadimos los datos de nuestro servidor y de nuestra base de datos (en este caso pondremos un "." en servidor y nuestra base de datos autoescuelap).



[Obtener datos de otro origen →](#)

En la siguiente pantalla escogemos “Usar mis credenciales actuales”, nos va a salir un warning pero lo ignoramos y continuamos.



Una vez se carguen los datos de nuestra base de datos, seleccionaremos la tabla donde se encuentran las imágenes y se verá de la siguiente manera:

profesor_Telf_profe	localidad_cod_localidad	fotografía	extensión
609876543		10	Binary JPG
610987654		9	Binary JPG
621098765		8	null null
632109876		7	Binary JPG
643210987		6	null null
654321098		5	null null
665432109		4	null null
676543210		3	null null
687654321		2	null null
698765432		1	null null

Debemos de tener en cuenta que hay tres pasos que debemos hacer:

- Cambiar el tipo de VARBINARY a TEXT
- Agregar una columna extra y añadir la fórmula
- Cambiar a url de imagen

Pinchamos en Transformar datos y se nos mostrará el editor de Power Query, donde podremos modificar los datos y la tabla.

The screenshot shows the Microsoft Power Query Editor interface. On the left, there's a table named 'profesor' with columns: localidad_cod_localidad, fotografía, extensión, contrato, localidad, and profe_prácticas. The 'fotografía' column contains binary data (JPG files). On the right, the 'Configuración de la consulta' pane is open, showing the 'Nombre' field set to 'profesor' and the 'Pasos aplicados' section which includes 'Origen' and 'Navegación'. The ribbon at the top has 'Inicio' selected, along with other tabs like 'Transformar', 'Agregar columna', 'Vista', 'Herramientas', and 'Ayuda'. The status bar at the bottom indicates '14 COLUMNAS, 10 FILAS' and 'Generación de perfiles de columnas basada en las 1000 primeras filas'.

Lo que queremos es que se nos muestren las fotos de los profesores. Para ello, primero cambiamos el tipo de dato a texto.

This screenshot shows the 'fotografía' column selected in the Power Query Editor. A context menu is open over the column header, with 'Cambiar tipo' highlighted. A dropdown menu lists various data types: Número decimal, Número decimal fijo, Número entero, Porcentaje, Fecha/Hora, Fecha, Hora, Fecha/Hora/Zona horaria, Duración, Texto, Verdadero/Falso, Binario, and Usar configuración regional... The 'Configuración de la consulta' pane on the right is visible, showing the same settings as the previous screenshot. The status bar at the bottom indicates '14 COLUMNAS, 10 FILAS' and 'Generación de perfiles de columnas basada en las 1000 primeras filas'.

Sin título - Editor de Power Query

Inicio

- Cerrar y aplicar
- Nuevo origen
- Orígenes recientes
- Especificar datos
- Configuración de origen de datos
- Administrar parámetros
- Actualizar vista previa
- Propiedades
- Editor avanzado
- Elegir columnas
- Quitar columnas
- Conservar filas
- Quitar filas
- Reducir filas
- Dividir columna
- Agrupar por
- Reemplazar los valores

Tipo de datos: Texto

Usar la primera fila como encabezado

Transformar

Consultas [1]

profesor

	localidad_cod_localidad	fotografía	extension	contrato	localidad	profe_prac
1		10 /9j/4AAQSkZJRgABAgAAAQAB...	JPG	Table	Value	Table
2		9 /9j/4AAQSkZJRgABAgAAAQAB...	JPG	Table	Value	Table
3		8 null	Table	Value	Table	Table
4		7 /9j/4AAQSkZJRgABAgAAAQAB...	JPG	Table	Value	Table
5		6 null	Table	Value	Table	Table
6		5 null	Table	Value	Table	Table
7		4 null	Table	Value	Table	Table
8		3 null	Table	Value	Table	Table
9		2 null	Table	Value	Table	Table
10		1 null	Table	Value	Table	Table

14 COLUMNAS, 10 FILAS Generación de perfiles de columnas basada en las 1000 primeras filas

VISTA PREVIA DESCARGADA EL SÁBADO, 11 DE ENERO DE 2025

Configuración de la consulta

PROPIEDADES

- Nombre: profesor
- Todas las propiedades

PASOS APLICADOS

- Origen
- Navegación
- Tipo cambiado

Ahora pinchamos en “Añadir Columna” y luego en “Columna personalizada” y añadiremos la siguiente fórmula y presionamos “Aceptar”.

Sin título - Editor de Power Query

Agregar columna

Columna a partir de los ejemplos

Columna personalizada

Columna condicional

Columna de índice

Formato

Combinar columnas

Extrar

Redondeo

Trigonometría

Estadísticas

Estándar

Científico

Información

Fecha

Hora

Duración

De número

De fecha y hora

Text

Visión

Azure Machine Learning

Analytics

Conclusiones de IA

Consultas [1]

profesor

	localidad_cod_localidad	fotografía	extension	contrato	localidad
1		10 /9j/4AAQSkZJRgABAgAAAQAB...	JPG	Table	Value
2		9 /9j/4AAQSkZJRgABAgAAAQAB...	JPG	Table	Value
3		8 null	Table	Value	Table
4		7 /9j/4AAQSkZJRgABAgAAAQAB...	JPG	Table	Value
5		6 null	Table	Value	Table
6		5 null	Table	Value	Table
7		4 null	Table	Value	Table
8		3 null	Table	Value	Table
9		2 null	Table	Value	Table
10		1 null	Table	Value	Table

Columna personalizada

Agreega una columna que se calcula a partir de otras columnas.

Nuevo nombre de columna: Imagen

Fórmula de columna personalizada: = "data:image/JPG;base64," & [fotografía]

Columnas disponibles:

- profesor_Email_profe
- profesor_Telf_profe
- localidad_cod_localidad
- fotografía
- extension
- contrato
- localidad

Información sobre fórmulas de Power Query

✓ No se han detectado errores de sintaxis.

Aceptar Cancelar

14 COLUMNAS, 10 FILAS Generación de perfiles de columnas basada en las 1000 primeras filas

VISTA PREVIA DESCARGADA EL SÁBADO, 11 DE ENERO DE 2025

Y ya solo nos quedaría presionar en "Cerrar y aplicar". Ahora cambiaremos a "Vista de informe" en el lateral izquierdo, aquí será donde diseñaremos la manera en que se nos muestra el resultado. Para ello, en el menú lateral de visualizaciones, seleccionamos "Tabla" y a continuación seleccionamos en "Datos" los datos que queremos que se nos muestren. En este caso hemos hecho la prueba con el nombre de los profesores y las imágenes (Que contienen las fotos de cada uno de ellos, estas las hemos generado con Gork, la IA de X).

Por alguna razón, tras realizar varias pruebas, no hemos conseguido que se muestren las fotos completamente, podría deberse a que el ordenador desde el cual trabajamos no tenga los requisitos mínimos requeridos, como los 16Gb de RAM.

9.- Particiones

Mecanismo que permite dividir una tabla o un índice en unidades más pequeñas denominadas particiones. Esto tiene ciertos beneficios como mejorar rendimiento, reducción de espacio de almacenamiento, mejor escalabilidad... Las divisiones han de

ser **siempre de forma horizontal**, es decir, por filas en función de un campo (fecha, valores, hash...)

1.- Creamos directorios para almacenar los archivos de la base de datos

(Podemos usar cmdshell):

```
EXECUTE sp_configure 'xp_cmdshell', 1;
GO
RECONFIGURE;
GO
xp_cmdshell 'mkdir C:\DATA\'
```

Como puede llevar a confusión, vamos a diferenciar entre filegroups y particiones:

Característica	FILEGROUP	PARTICIÓN
Definición	Un grupo lógico de archivos de datos donde se almacenan tablas e índices.	Una división de una tabla o un índice en varias partes según un rango de valores.
Objetivo	Distribuir físicamente los datos en distintos discos para mejorar rendimiento y administración.	Mejorar el acceso a los datos separándolos en diferentes particiones según criterios específicos.
Alcance	Afecta a toda la base de datos, permitiendo almacenar diferentes objetos en distintos FILEGROUPS.	Se aplica a una única tabla o índice para dividir sus datos.
Ejemplo de uso	Poner los datos de una base en distintos discos para optimizar el rendimiento.	Separar registros de una tabla por fechas para consultas más rápidas.
Configuración	Se crean al definir la base de datos (CREATE DATABASE).	Se crean al definir una tabla particionada (CREATE PARTITION FUNCTION).

Cuando tenemos la base de datos almacenada en un único disco o cuando la base de datos es demasiado pequeña, no compensa usar filegroups porque no se notarán beneficios.

Ahora vamos a crear los archivos de la base de datos:

```
CREATE DATABASE [AutoescuelaP]
ON PRIMARY ( NAME = 'AutoescuelaP',
    FILENAME = 'C:\Data\AutoescuelaP_Fijo.mdf' ,
    SIZE = 15360KB , MAXSIZE = UNLIMITED, FILEGROWTH = 0)
LOG ON ( NAME = 'AutoescuelaP_log',
    FILENAME = 'C:\Data\AutoescuelaP_log.ldf' ,
    SIZE = 10176KB , MAXSIZE = 2048GB , FILEGROWTH = 10%)
GO
```

O bien si tenemos la base de datos, creamos los archivos y se vinculan los filegroups a cada archivo

```
ALTER DATABASE [autoescuelaP] ADD FILE ( NAME = 'Antiguos', FILENAME = 'c:\DATA\Antiguos.ndf', SIZE = 5M
GO

ALTER DATABASE [autoescuelaP] ADD FILE ( NAME = 'Altas_2023', FILENAME = 'c:\DATA\Altas_2023.ndf', SIZE =
GO

ALTER DATABASE [autoescuelaP] ADD FILE ( NAME = 'Altas_2024', FILENAME = 'c:\DATA\Altas_2024.ndf', SIZE =
GO
```

```
ALTER DATABASE [autoescuelaP] ADD FILE ( NAME = 'Altas_2025', FILENAME = 'c:\DATA\Altas_2025.ndf', SIZE :  
GO
```

****IMPORTANTE:** Tener creado previamente el directorio donde vamos a crear los archivos, sino nos dará error.

```

42 ALTER DATABASE [autoescuelaP] ADD FILE ( NAME = 'Antiguos', FILENAME = 'c:\DATA\Antiguos.ndf', SIZE = 5MB, MAXSIZE = 100MB, FILEGROWTH = 2MB ) TO FILEGROUP [Antiguos]
43 GO
44
45 ALTER DATABASE [autoescuelaP] ADD FILE ( NAME = 'Altas_2023', FILENAME = 'c:\DATA\Altas_2023.ndf', SIZE = 5MB, MAXSIZE = 100MB, FILEGROWTH = 2MB ) TO FILEGROUP [Altas_2023]
46 GO
47
48 ALTER DATABASE [autoescuelaP] ADD FILE ( NAME = 'Altas_2024', FILENAME = 'c:\DATA\Altas_2024.ndf', SIZE = 5MB, MAXSIZE = 100MB, FILEGROWTH = 2MB ) TO FILEGROUP [Altas_2024]
49 GO
50
51 ALTER DATABASE [autoescuelaP] ADD FILE ( NAME = 'Altas_2025', FILENAME = 'c:\DATA\Altas_2025.ndf', SIZE = 5MB, MAXSIZE = 100MB, FILEGROWTH = 2MB ) TO FILEGROUP [Altas_2025]
52 GO
53
54
55 select file_id, name, physical_name
56 from sys.database_files
57 GO
58
59
60
61

```

file_id	name	physical_name
1	autoescuela	C:\Program Files\Microsoft SQL Server\MSSQL16.MSS...
2	autoescuela_log	C:\Program Files\Microsoft SQL Server\MSSQL16.MSS...
3	Antiguos	c:\DATA\Antiguos.ndf
4	Altas_2023	c:\DATA\Altas_2023.ndf
5	Altas_2024	c:\DATA\Altas_2024.ndf
6	Altas_2025	c:\DATA\Altas_2025.ndf
7	FOTOGRAFIAS	C:\FOTOGRAFIAS

Ahora creamos la tabla para añadir los registros y posteriormente comprobamos como directamente cada registro se ha distribuido en cada partición

```

93 CREATE TABLE Altas_matricula
94 (
    id_alta int identity (1,1),
    nombre varchar(20),
    apellido varchar (20),
    fecha_alta datetime
)
95 ON altas_fecha
96     (fecha_alta)
97
98 GO
99
100
101
102
103 -- Pedimos a la IA que nos genere registros de ejemplo para rellenar la tabla
104 INSERT INTO Altas_matricula (nombre, apellido, fecha_alta) VALUES ('Carlos', 'García', '2019-03-10');...
105 GO
106
107 INSERT INTO Altas_matricula (nombre, apellido, fecha_alta)...
108 GO
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176

```

Numero_Particion	Registros
1	25
2	8
3	7
4	20

```

178 SELECT *, $Partition.FN_altas_fecha(fecha_alta) AS Partition
179 FROM Altas_matricula
180 GO

```

100 %

	id_alta	nombre	apellido	fecha_alta	Partition
22	274	Laura	González	2020-10-15 00:00:00.000	1
23	275	Antonio	Rodríguez	2018-09-21 00:00:00.000	1
24	276	Ana	Pérez	2021-07-23 00:00:00.000	1
25	280	Luis	Jiménez	2020-12-05 00:00:00.000	1
26	245	Antonio	Rodríguez	2023-02-03 00:00:00.000	2
27	247	David	Sánchez	2023-08-05 00:00:00.000	2
28	254	Laura	González	2023-01-25 00:00:00.000	2
29	257	David	Sánchez	2023-10-14 00:00:00.000	2
30	266	Ana	Pérez	2023-12-17 00:00:00.000	2

Query executed successfully.

SPLIT

Split lo que hace es dividir una partición, vamos a establecer por ejemplo una partición para los registros del año 2022

```

177
178 ALTER PARTITION FUNCTION FN_altas_fecha()
179   SPLIT RANGE ('2022-01-01');
180 GO

```

100 %

Messages

Msg 7710, Level 16, State 1, Line 178
Warning: The partition scheme 'altas_fecha' does not have any next used filegroup. Partition scheme has not been changed.
Completion time: 2025-03-04T17:46:43.0867501+01:00

Como vemos, debemos de tener un filegroup libre para asignar la partición que queremos crear, de lo contrario se nos mostrará un warning. En este caso, no hemos dejado ninguno libre, por lo que hemos tenido que crear el filegroup nuevo para la partición que se va a generar, adjudicarle el archivo y modificar el esquema de las particiones.

```

175 ALTER DATABASE [autoescuelaP] ADD FILEGROUP [Altas_2022]
176 GO
177
178 ALTER DATABASE [autoescuelaP] ADD FILE ( NAME = 'Altas_2022', FILENAME = 'c:\DATA\Altas_2022.ndf', SIZE = 5MB, MAXSIZE = 100MB, FILEGROWTH = 2MB ) TO FILEGROUP [Altas_2022]
179 GO
180
181 ALTER PARTITION SCHEME altas_fecha
182   NEXT USED Altas_2022;
183
184
185 ALTER PARTITION FUNCTION FN_altas_fecha()
186   SPLIT RANGE ('2022-01-01');
187 GO
188
189 SELECT *, $Partition.FN_altas_fecha(fecha_alta) AS Partition
190 FROM Altas_matricula
191 GO

```

100 %

	id_alta	nombre	apellido	fecha_alta	Partition
13	269	Franc...	Díaz	2021-03-04 00:00:00.000	1
14	271	Carlos	García	2021-06-14 00:00:00.000	1
15	274	Laura	González	2020-10-15 00:00:00.000	1
16	275	Antonio	Rodríguez	2018-09-21 00:00:00.000	1
17	276	Ana	Pérez	2021-07-23 00:00:00.000	1
18	280	Luis	Jiménez	2020-12-05 00:00:00.000	1
19	243	José	López	2022-07-24 00:00:00.000	2
20	249	Franc...	Díaz	2022-04-07 00:00:00.000	2
21	255	Antonio	Rodríguez	2022-05-30 00:00:00.000	2

Query executed successfully.

MERGE

Merge lo que hace es fusionar particiones, elimina la barrera que separa dos particiones y las une. En este caso vamos a suprimir la partición que hemos hecho con split.

```

196 ALTER PARTITION FUNCTION FN_Altas_Fecha () 
197 MERGE RANGE ('2022-01-01');
198 GO
199
200 SELECT *, $Partition.FN_altas_fecha(fecha_alta) AS Partition
201 FROM Altas_matricula
202 GO

```

100 % ▶

	id_alta	nombre	apellido	fecha_alta	Partition
22	274	Laura	González	2020-10-15 00:00:00.000	1
23	275	Antonio	Rodriguez	2018-09-21 00:00:00.000	1
24	276	Ana	Pérez	2021-07-23 00:00:00.000	1
25	280	Luis	Jiménez	2020-12-05 00:00:00.000	1
26	245	Antonio	Rodriguez	2023-02-03 00:00:00.000	2
27	247	David	Sánchez	2023-08-05 00:00:00.000	2
28	254	Laura	González	2023-01-25 00:00:00.000	2
29	257	David	Sánchez	2023-10-14 00:00:00.000	2
30	266	Ana	Pérez	2023-12-17 00:00:00.000	2

Query executed successfully.

SWITCH

Switch nos sirve para intercambiar particiones, en este caso lo vamos a probar para llevar la partición 1 a la tabla que guarda las altas antiguas y así dejar en la tabla las particiones de los ultimos 3 años.

```

211 CREATE TABLE Archivo_antiguos
212 ( id_alta int identity (1,1),
213 nombre varchar(20),
214 apellido varchar (20),
215 fecha_alta datetime )
216 ON Antiguos
217 go
218
219 ALTER TABLE Altas_matricula
220 SWITCH Partition 1 to Archivo_antiguos
221 go
222
223 select * from Altas_matricula
224 go
225
226

```

100 % ▶

	id_alta	nombre	apellido	fecha_alta
1	245	Antonio	Rodríguez	2023-02-03 00:00:00.000
2	247	David	Sánchez	2023-08-05 00:00:00.000
3	254	Laura	González	2023-01-25 00:00:00.000
4	257	David	Sánchez	2023-10-14 00:00:00.000
5	266	Ana	Pérez	2023-12-17 00:00:00.000
6	270	Luis	Jiménez	2023-02-11 00:00:00.000
7	273	José	López	2023-06-09 00:00:00.000
8	278	Carmen	Hemández	2023-11-14 00:00:00.000
9	246	Ana	Pérez	2024-01-15 00:00:00.000

Query executed successfully.

TRUNCATE

Truncate nos sirve para eliminar particiones, es la operación mas simple. Vamos a probar a borrar la partición 4 que es la que tiene las altas del último año.

```

229  TRUNCATE TABLE Altas_matricula
230      WITH (PARTITIONS (4));
231  GO
232
233
234  select * from Altas_matricula
235  GO
236
237
238
```

100 % ▾

Results Messages

	id_alta	nombre	apellido	fecha_alta
7	273	José	López	2023-06-09 00:00:00.000
8	278	Carmen	Hemández	2023-11-14 00:00:00.000
9	246	Ana	Pérez	2024-01-15 00:00:00.000
10	252	Maria	Martinez	2024-04-15 00:00:00.000
11	256	Ana	Pérez	2024-03-22 00:00:00.000
12	261	Carlos	Garcia	2024-10-11 00:00:00.000
13	267	David	Sánchez	2024-01-28 00:00:00.000
14	277	David	Sánchez	2024-02-17 00:00:00.000
15	279	Franc...	Diaz	2024-09-28 00:00:00.000

Query executed successfully.

```

237  SELECT *, $Partition.FN_altas_fecha(fecha_alta) AS Partition
238  FROM Altas_matricula
239  GO
240
241
```

100 % ▾

Results Messages

	id_alta	nombre	apellido	fecha_alta	Partition
7	273	José	López	2023-06-09 00:00:00.000	2
8	278	Carmen	Hemández	2023-11-14 00:00:00.000	2
9	246	Ana	Pérez	2024-01-15 00:00:00.000	3
10	252	Maria	Martinez	2024-04-15 00:00:00.000	3
11	256	Ana	Pérez	2024-03-22 00:00:00.000	3
12	261	Carlos	Garcia	2024-10-11 00:00:00.000	3
13	267	David	Sánchez	2024-01-28 00:00:00.000	3
14	277	David	Sánchez	2024-02-17 00:00:00.000	3
15	279	Franc...	Diaz	2024-09-28 00:00:00.000	3

Query executed successfully.

En una tabla particionada, la PK debe incluir siempre la columna mediante la cual se partitiona la tabla. Es decir, que si designamos una PK en dicha tabla, uno de los datos debe de ser el discriminando. En nuestro ejemplo, si designamos como PK solo el campo "id_alta" nos dará un error.

```

243  CREATE TABLE Altas_matricula
244  (
245      id_alta INT IDENTITY (1,1),
246      nombre VARCHAR(20),
247      apellido VARCHAR(20),
248      fecha_alta DATETIME,
249      CONSTRAINT PK_Altas_matricula PRIMARY KEY CLUSTERED (id_alta)
250  )
251  ON altas_fecha (fecha_alta);
252  GO
253
254
```

100 % ▾

Messages

Msg 1908, Level 16, State 1, Line 243
Column 'fecha_alta' is partitioning column of the index 'PK_Altas_matricula'. Partition columns for a unique index must be a subset of the index key.
Msg 1750, Level 16, State 1, Line 243
Could not create constraint or index. See previous errors.

Completion time: 2025-03-04T20:21:16.2050472+01:00

REPASO CREAR PARTICIÓN

1.- Creamos los fliesgroups para cada partición.

```
ALTER DATABASE [nombre_BD] ADD FILEGROUP [nombre_FG]
GO
```

2.- Creamos archivos y les vinculamos los filesgroups.

```
ALTER DATABASE [nombre_BD] ADD FILE ( NAME = 'nombre_archivo', FILENAME = 'ruta_archivo', SIZE = 5MB, N
GO
```

3.- Creamos la función de partición.

```
CREATE PARTITION FUNCTION nombre_funcion (tipo_de_dato)
AS RANGE RIGHT/LEFT
FOR VALUES ('valores_de_rangos')GO
```

4.- Creamos el esquema de la partición.

```
CREATE PARTITION SCHEME nombre_esquema
AS PARTITION nombre_funcion
TO (FG1,FG2,FG3...)
GO
```

5.- Creamos la tabla e introducimos datos.

```
CREATE TABLE nombre_tabla
( id_alta int identity (1,1),
nombre varchar(20),
apellido varchar (20),
fecha_alta datetime )
ON nombre_esquema
(nombre_dato_que_particiona)
GO
```

Siempre hay que incluir el discriminando de la partición dentro de la PK de la tabla particionada

Otro ejemplo Particiones (integer)

Probaremos otro ejemplo, esta vez realizaremos la partición de la tabla cliente en función de su código de localidad, si la localidad es gallega (códigos entre el 1 y el 4), el cliente se guardará en la partición correspondiente a esa localidad y si la localidad no es gallega se guardará en una partición a parte (ya que serán clientes minoritarios).

Para ello:

```
-- 1.- CREO FILEGROUPS

ALTER DATABASE [autoescuelaP] ADD FILEGROUP [Otras]
GO

ALTER DATABASE [autoescuelaP] ADD FILEGROUP [Coruna]
GO

ALTER DATABASE [autoescuelaP] ADD FILEGROUP [Lugo]
GO

ALTER DATABASE [autoescuelaP] ADD FILEGROUP [Orense]
GO
```

```
ALTER DATABASE [autoescuelaP] ADD FILEGROUP [Pontevedra]
GO
```

-- 2.- CREO LOS ARCHIVOS Y VINCULO LOS FILEGROUPS
-- Se crean archivos para añadir cada filegroup a un archivo

```
ALTER DATABASE [autoescuelaP] ADD FILE ( NAME = 'Otras', FILENAME = 'c:\DATA\otras', SIZE = 5MB, MAXSIZE = 50MB, FILEGROU
GO
```

```
ALTER DATABASE [autoescuelaP] ADD FILE ( NAME = 'Coruna', FILENAME = 'c:\DATA\coruna.ndf', SIZE = 5MB, MAXSIZE = 50MB, FILEGROU
GO
```

```
ALTER DATABASE [autoescuelaP] ADD FILE ( NAME = 'Lugo', FILENAME = 'c:\DATA\lugo.ndf', SIZE = 5MB, MAXSIZE = 50MB, FILEGROU
GO
```

```
ALTER DATABASE [autoescuelaP] ADD FILE ( NAME = 'Orense', FILENAME = 'c:\DATA\orense.ndf', SIZE = 5MB, MAXSIZE = 50MB, FILEGROU
GO
```

```
ALTER DATABASE [autoescuelaP] ADD FILE ( NAME = 'Pontevedra', FILENAME = 'c:\DATA\pontevedra.ndf', SIZE = 5MB, MAXSIZE = 50MB, FILEGROU
GO
```

-- 3.- CREO FUNCION PARTICION
-- en este caso segúnel código de la localidad

```
DROP PARTITION FUNCTION FN_Localidades;
GO
```

```
CREATE PARTITION FUNCTION FN_Localidades (integer)
AS RANGE LEFT -- En este caso
    FOR VALUES (1,2,3,4); -- del 5 en adelante se guarda en la partición restante
GO
```

-- 4.- CREO EL ESQUEMA DE PARTICIÓN
-- Nos mapeará los registros según las particiones

```
CREATE PARTITION SCHEME SC_Localidades
AS PARTITION FN_Localidades
    TO (Coruna,Lugo,Orense,Pontevedra,Otras);
GO
```

```

361
362 SELECT *, $Partition.FN_Localidades (localidad_cod_localidad) AS Partition
363 FROM cliente
364 ORDER BY Partition;
365 GO

```

100 % Results Messages

DNI_cliente	N_cliente	Apel1_cliente	Apel2_cliente	Dir_cliente	Email_cliente	Telf_cliente	localidad_cod_localidad	Partition
7	77889900A	Lucía	Castro	Navarro	Calle del Mar 14	lucia.castro@email.com	677889900	1
8	78901234N	David	González	Jiménez	Calle Ancha 7	david.gonzalez@email.com	678901234	2
9	88990011W	Hugo	Santos	Méndez	Avenida de los Olivos 17	hugo.santos@email.com	688990011	2
10	23456789B	Juan	Martínez	Sánchez	Avenida Principal 45	juan.martinez@email.com	623456789	2
11	23456789Z	Laura	Fernández	Ruiz	Avenida del Sol 8	laura.fernandez@email.c...	623456789	2
12	33445566G	Clara	Ruiz	Castro	Avenida Libertad 4	clara.ruiz@email.com	633445566	2
13	18192021I	Manuel	Campos	Soto	Camino del Norte 12	manuel.campos@email.c...	618192021	2
14	13141516W	Beatriz	Cabrera	Molina	Calle del Bosque 21	beatriz.cabrera@email.com	613141516	2
15	14151617Q	Óscar	Nieto	Rey	Paseo del Sol 13	oscar.nieto@email.com	614151617	3
16	19202122U	Carmen	Crespo	Pascual	Avenida Central 23	carmen.crespo@email.com	619202122	3
17	34567890C	Ana	Fernández	Ruiz	Plaza Mayor 7	ana.fernandez@email.com	634567890	3
18	34567890X	Miguel	Martínez	Sánchez	Paseo de la Castellana...	miguel.martinez@email.com	634567890	3
19	44556677F	Pablo	Jiménez	Santos	Calle Verde 18	pablo.jimenez@email.com	644556677	3
20	89012345M	Elena	Rodríguez	Moreno	Avenida de la Paz 10	elena.rodriguez@email.com	689012345	3

Query executed successfully. | BCR_SXBD (16.0 RTM)

Ln 365 Col 3 Ch 3 INS

Vamos a usar SPLIT para crear una partición para la localidad de MADRID

```

ALTER DATABASE [autoescuelaP] ADD FILEGROUP [Madrid]
GO

```

```

ALTER DATABASE [autoescuelaP] ADD FILE ( NAME = 'Madrid', FILENAME = 'c:\DATA\madrid.ndf', SIZE = 5MB, M
GO

```

```

ALTER PARTITION SCHEME SC_Localidades
NEXT USED Madrid;

```

```

ALTER PARTITION FUNCTION FN_Localidades()
SPLIT RANGE (5);
GO

```

```

379
380 SELECT *, $Partition.FN_Localidades (localidad_cod_localidad) AS Partition
381 FROM cliente
382 ORDER BY Partition;
383 GO
384
385

```

100 % Results Messages

DNI_cliente	N_cliente	Apel1_cliente	Apel2_cliente	Dir_cliente	Email_cliente	Telf_cliente	localidad_cod_localidad	Partition
31	56789012V	Javier	Pérez	Díaz	Plaza Mayor 3	javier.perez@email.com	656789012	5
32	66778899S	Fernando	Torres	Vega	Camino Rojo 11	fernando.torres@email....	666778899	5
33	11121314R	Eva	Peña	Cabrera	Calle del Parque 16	eva.pena@email.com	611121314	5
34	11223344J	Marina	Díaz	Torres	Camino Real 9	marina.diaz@email.com	611223344	5
35	67890123F	David	González	Martín	Calle del Sol 34	david.gonzalez@email....	667890123	6
36	78901234G	Elena	Sánchez	Jiménez	Plaza España 12	elena.sanchez@email....	678901234	7
37	89012345H	Pablo	Díaz	Moreno	Calle Mayor 78	pablo.diaz@email.com	689012345	8

Query executed successfully. | BCR_SXBD (16.0 RTM)

Ahora vamos a intentar copiar los datos de la partición 6 (clientes del resto de España) en una tabla específica para dichos clientes.

```

DROP TABLE cliente_espana;
GO

```

```

CREATE TABLE cliente_espana (
    DNI_cliente VARCHAR(9) PRIMARY KEY,
    N_cliente NVARCHAR(12),
    Apel1_cliente NVARCHAR(12),

```

```

Apel2_cliente NVARCHAR(12),
Dir_cliente NVARCHAR(50),
Email_cliente NVARCHAR(30),
Telf_cliente NVARCHAR(16),
localidad_cod_localidad INT not null
)
ON Otras

```

GO

```

ALTER TABLE cliente
SWITCH Partition 6 to cliente_espana
GO

```

Pero vemos que nos da un error:

*En principio, solo se pueden mover datos dentro de la misma partición

10.- Tablas temporales versión sistema

Las tablas temporales versión sistema en SQL Server son una característica que permite almacenar automáticamente el historial de los datos de una tabla a lo largo del tiempo. Esto facilita la auditoría, el análisis de cambios y la recuperación de datos en un momento específico sin necesidad de implementar manualmente un sistema de control de versiones. Muy parecido a GitHub.

1. Se define una tabla con dos columnas de fecha (`SysStartTime` y `SysEndTime`) para registrar el rango de tiempo en el que un registro es válido.
2. SQL Server crea automáticamente una **tabla de historial** asociada donde almacena las versiones antiguas de los registros.
3. Cuando se **actualiza o elimina** un registro, SQL Server mueve la versión anterior a la tabla de historial antes de modificar el dato en la tabla principal.
4. Se pueden hacer consultas para recuperar datos de un momento específico.

Vamos a crear la tabla contratos en nuestra base de datos, donde almacenaremos los contratos que vayan teniendo los trabajadores y añadiremos la tabla temporal versión sistema

```

CREATE TABLE [dbo].[contratos](
    [ID_contrato] [int] NOT NULL PRIMARY KEY,
    [Cargo] [nvarchar](18) NULL,
    [T_contrato] [nvarchar](12) NULL,
    [T_jornada] [nvarchar](12) NULL,
    [Sueldo] [decimal](10, 2) NULL,
    [Antiguedad] [nvarchar](9) NULL,
    [profesor_profesor_DNI_profe] [varchar](9) NOT NULL,
    SysStartTime DATETIME2 GENERATED ALWAYS AS ROW START NOT NULL,
    SysEndTime DATETIME2 GENERATED ALWAYS AS ROW END NOT NULL,
    CONSTRAINT DF_contratos_SysStartTime DEFAULT (GETDATE())
)
```

```
PERIOD FOR SYSTEM_TIME (SysStartTime, SysEndTime)
```

```
WITH (SYSTEM_VERSIONING = ON (HISTORY_TABLE = dbo.contratos_historial));
GO
```

**SysStartTime y SysEndTime: Son columnas especiales que almacenan automáticamente el tiempo en que cada registro fue válido.

PERIOD FOR SYSTEM_TIME: Define el rango de validez de cada fila.

SYSTEM_VERSIONING = ON: Activa el versionado temporal y genera automáticamente la tabla de historial.

**Si queremos modificar una tabla para añadirle tablas temporales version de sistema, debemos tener en cuenta que primero debemos añadir la tabla y luego activar el control de versiones, si intentamos hacerlo todo a la vez nos dará un error.

Ahora haremos modificaciones en la tabla contratos para aumentar sueldos, antiguedad y cambiar algún cargo:

```
UPDATE [dbo].[contratos]
SET Sueldo = 3500.75, Antiguedad = '18 años'
WHERE ID_contrato = 1;
```

```
UPDATE [dbo].[contratos]
SET Sueldo = 4600.50, Antiguedad = '10 años'
WHERE ID_contrato = 2;
GO
```

```
UPDATE [dbo].[contratos]
SET Cargo = 'Coordinador'
WHERE ID_contrato = 1;
GO
```

```
PRINT GETUTCDATE()
GO
```

```
--Mar 22 2025 11:03AM
--Completion time: 2025-03-22T12:03:59.0400540+01:00
```

```
UPDATE [dbo].[contratos]
SET Sueldo = 3500.75, Antiguedad = '18 años'
WHERE ID_contrato = 1;
```

```
UPDATE [dbo].[contratos]
SET Sueldo = 4600.50, Antiguedad = '10 años'
WHERE ID_contrato = 2;
GO
```

```
PRINT GETUTCDATE()
GO
--Mar 22 2025 11:04AM
--Completion time: 2025-03-22T12:04:42.3275737+01:00
```

Ahora vamos a ver lo que se refleja en la tabla historial y buscaremos en concreto los cambios que han sufrido los contratos de un trabajador en concreto:

```

PRINT GETUTCDATE()
GO

--Mar 22 2025 11:03AM
--Completion time: 2025-03-22T12:03:59.0400540+01:00

UPDATE [dbo].[contratos]
SET Sueldo = 3500.75, Antiguedad = '18 años'
WHERE ID_contrato = 1;
GO

UPDATE [dbo].[contratos]
SET Sueldo = 4600.50, Antiguedad = '10 años'
WHERE ID_contrato = 2;
GO

PRINT GETUTCDATE()
GO

--Mar 22 2025 11:04AM
--Completion time: 2025-03-22T12:04:42.3275737+01:00

SELECT * FROM contratos_historial;
GO

```

100 %

	ID_contrato	Cargo	T_contrato	T_jornada	Sueldo	Antiguedad	profesor_profesor_DNI_profe	SysStartTime	SysEndTime
1	1	Profesor	Fijo	Completa	2200.50	10 años	12345678A	2025-03-22 10:57:31.9786589	2025-03-22 11:00:26.4551783
2	2	Profesor	Temporal	Parcial	1500.00	3 años	23456789B	2025-03-22 10:57:31.9786589	2025-03-22 11:00:26.4551783
3	1	Profesor	Fijo	Completa	2400.75	12 años	12345678A	2025-03-22 11:00:26.4551783	2025-03-22 11:00:26.4780030
4	2	Profesor	Temporal	Parcial	1600.50	5 años	23456789B	2025-03-22 11:00:26.4551783	2025-03-22 11:04:12.2350513
5	1	Coordinador	Fijo	Completa	2400.75	12 años	12345678A	2025-03-22 11:00:26.4780030	2025-03-22 11:04:12.2350513
6	1	Coordinador	Fijo	Completa	3500.75	18 años	12345678A	2025-03-22 11:04:12.2350513	2025-03-22 11:04:39.8401580
7	2	Profesor	Temporal	Parcial	4600.50	10 años	23456789B	2025-03-22 11:04:12.2350513	2025-03-22 11:04:39.8401580

```

SELECT * FROM contratos_historial
WHERE ID_contrato = 1;
GO

```

100 %

	ID_contrato	Cargo	T_contrato	T_jornada	Sueldo	Antiguedad	profesor_profesor_DNI_profe	SysStartTime	SysEndTime
1	1	Profesor	Fijo	Completa	2200.50	10 años	12345678A	2025-03-22 10:57:31.9786589	2025-03-22 11:00:26.4551783
2	1	Profesor	Fijo	Completa	2400.75	12 años	12345678A	2025-03-22 11:00:26.4551783	2025-03-22 11:00:26.4780030
3	1	Coordinador	Fijo	Completa	2400.75	12 años	12345678A	2025-03-22 11:00:26.4780030	2025-03-22 11:04:12.2350513
4	1	Coordinador	Fijo	Completa	3500.75	18 años	12345678A	2025-03-22 11:04:12.2350513	2025-03-22 11:04:39.8401580

Como vemos, se registran todos los cambios que ha ido sufriendo la tabla y el periodo en el que ha permanecido así ese cambio.

Ahora vamos a eliminar uno de los profesores de la tabla porque se ha jubilado y vamos a comprobar que desaparece de la tabla contratos pero no del historial.

```

DELETE FROM contratos
WHERE ID_contrato = 1;
GO

```

SELECT * FROM contratos;
GO

100 %

Results Messages

ID_contrato	Cargo	T_contrato	T_jomada	Sueldo	Antiguedad	profesor_profesor_DNI_profe	SysStartTime	SysEndTime
1	2	Profesor	Temporal	Parcial	4600.50	10 años	23456789B	2025-03-22 11:04:39.8401580 9999-12-31 23:59:59.9999999
2	3	Profesor	Fijo	Completa	2500.75	7 años	34567890C	2025-03-22 10:57:31.9786589 9999-12-31 23:59:59.9999999
3	4	Admin	Fijo	Completa	2700.00	5 años	45678901D	2025-03-22 10:57:31.9786589 9999-12-31 23:59:59.9999999
4	5	Profesor	Temporal	Parcial	1800.30	2 años	56789012E	2025-03-22 10:57:31.9786589 9999-12-31 23:59:59.9999999
5	6	Coordinador	Fijo	Completa	3000.00	12 años	67890123F	2025-03-22 10:57:31.9786589 9999-12-31 23:59:59.9999999
6	7	Profesor	Fijo	Parcial	2000.50	8 años	78901234G	2025-03-22 10:57:31.9786589 9999-12-31 23:59:59.9999999
7	8	Profesor	Temporal	Completa	1600.80	4 años	89012345H	2025-03-22 10:57:31.9786589 9999-12-31 23:59:59.9999999
8	9	Secretario	Fijo	Completa	1900.90	6 años	90123456I	2025-03-22 10:57:31.9786589 9999-12-31 23:59:59.9999999
9	10	Profesor	Temporal	Parcial	1400.75	1 año	11234567J	2025-03-22 10:57:31.9786589 9999-12-31 23:59:59.9999999
10	11	J.Estudios	Fijo	Completa	3500.00	15 años	22345678K	2025-03-22 10:57:31.9786589 9999-12-31 23:59:59.9999999
11	12	Profesor	Temporal	Parcial	1750.60	3 años	33456789L	2025-03-22 10:57:31.9786589 9999-12-31 23:59:59.9999999
12	13	Contable	Fijo	Completa	2800.00	9 años	44567890M	2025-03-22 10:57:31.9786589 9999-12-31 23:59:59.9999999

En la tabla contratos ya no figura el profesor ya que no tiene ningún contrato activo, sin embargo en la tabla historial figura y figuran todos los cambios que han sufrido sus contratos a lo largo del tiempo y con las fechas de cada periodo.

SELECT * FROM contratos_historial;
GO

100 %

Results Messages

ID_contrato	Cargo	T_contrato	T_jomada	Sueldo	Antiguedad	profesor_profesor_DNI_profe	SysStartTime	SysEndTime
1	1	Profesor	Fijo	Completa	2200.50	10 años	12345678A	2025-03-22 10:57:31.9786589 2025-03-22 11:00:26.4551783
2	2	Profesor	Temporal	Parcial	1500.00	3 años	23456789B	2025-03-22 10:57:31.9786589 2025-03-22 11:00:26.4551783
3	1	Profesor	Fijo	Completa	2400.75	12 años	12345678A	2025-03-22 11:00:26.4551783 2025-03-22 11:00:26.4780030
4	2	Profesor	Temporal	Parcial	1600.50	5 años	23456789B	2025-03-22 11:00:26.4551783 2025-03-22 11:04:12.2350513
5	1	Coordinador	Fijo	Completa	2400.75	12 años	12345678A	2025-03-22 11:00:26.4780030 2025-03-22 11:04:12.2350513
6	1	Coordinador	Fijo	Completa	3500.75	18 años	12345678A	2025-03-22 11:04:12.2350513 2025-03-22 11:04:39.8401580
7	2	Profesor	Temporal	Parcial	4600.50	10 años	23456789B	2025-03-22 11:04:12.2350513 2025-03-22 11:04:39.8401580
8	1	Coordinador	Fijo	Completa	3500.75	18 años	12345678A	2025-03-22 11:04:39.8401580 2025-03-22 11:20:33.2694193

Ahora vamos a consultar todo lo que ha ocurrido en la tabla desde la creación de la misma:

```
SELECT * FROM contratos
FOR system_time ALL
GO
```

SELECT * FROM contratos
FOR system_time ALL
GO

100 %

Results Messages

ID_contrato	Cargo	T_contrato	T_jomada	Sueldo	Antiguedad	profesor_profesor_DNI_profe	SysStartTime	SysEndTime
9	10	Profesor	Temporal	Parcial	1400.75	1 año	11234567J	2025-03-22 10:57:31.9786589 9999-12-31 23:59:59.9999999
10	11	J.Estudios	Fijo	Completa	3500.00	15 años	22345678K	2025-03-22 10:57:31.9786589 9999-12-31 23:59:59.9999999
11	12	Profesor	Temporal	Parcial	1750.60	3 años	33456789L	2025-03-22 10:57:31.9786589 9999-12-31 23:59:59.9999999
12	13	Contable	Fijo	Completa	2800.00	9 años	44567890M	2025-03-22 10:57:31.9786589 9999-12-31 23:59:59.9999999
13	14	Profesor	Fijo	Completa	2300.40	11 años	55678901N	2025-03-22 10:57:31.9786589 9999-12-31 23:59:59.9999999
14	15	Profesor	Temporal	Parcial	1550.20	2 años	66789012O	2025-03-22 10:57:31.9786589 9999-12-31 23:59:59.9999999
15	1	Profesor	Fijo	Completa	2200.50	10 años	12345678A	2025-03-22 10:57:31.9786589 2025-03-22 11:00:26.4551783
16	2	Profesor	Temporal	Parcial	1500.00	3 años	23456789B	2025-03-22 10:57:31.9786589 2025-03-22 11:00:26.4551783
17	1	Profesor	Fijo	Completa	2400.75	12 años	12345678A	2025-03-22 11:00:26.4551783 2025-03-22 11:00:26.4780030
18	2	Profesor	Temporal	Parcial	1600.50	5 años	23456789B	2025-03-22 11:00:26.4551783 2025-03-22 11:04:12.2350513
19	1	Coordinador	Fijo	Completa	2400.75	12 años	12345678A	2025-03-22 11:00:26.4780030 2025-03-22 11:04:12.2350513
20	1	Coordinador	Fijo	Completa	3500.75	18 años	12345678A	2025-03-22 11:04:12.2350513 2025-03-22 11:04:39.8401580
21	2	Profesor	Temporal	Parcial	4600.50	10 años	23456789B	2025-03-22 11:04:12.2350513 2025-03-22 11:04:39.8401580
22	1	Coordinador	Fijo	Completa	3500.75	18 años	12345678A	2025-03-22 11:04:39.8401580 2025-03-22 11:20:33.2694193

AS OF

Ahora vamos a pedirle al sistema que nos muestre como estaba la tabla en un determinado momento del tiempo:

```

SELECT * FROM contratos
FOR system_time AS OF '2025-03-22T10:57:40'
ORDER BY ID_contrato;
GO

```

-- Estado de la tabla en un momento determinado

```

SELECT * FROM contratos
FOR system_time AS OF '2025-03-22T10:57:40'
ORDER BY ID_contrato;
GO

```

100 %

	ID_contrato	Cargo	T_contrato	T_jomada	Sueldo	Antiguedad	profesor_profesor_DNI_profe	SysStartTime	SysEndTime
1	1	Profesor	Fijo	Completa	2200.50	10 años	12345678A	2025-03-22 10:57:31.9786589	2025-03-22 11:00:26.4551783
2	2	Profesor	Temporal	Parcial	1500.00	3 años	23456789B	2025-03-22 10:57:31.9786589	2025-03-22 11:00:26.4551783
3	3	Profesor	Fijo	Completa	2500.75	7 años	34567890C	2025-03-22 10:57:31.9786589	9999-12-31 23:59:59.9999999
4	4	Admin	Fijo	Completa	2700.00	5 años	45678901D	2025-03-22 10:57:31.9786589	9999-12-31 23:59:59.9999999
5	5	Profesor	Temporal	Parcial	1800.30	2 años	56789012E	2025-03-22 10:57:31.9786589	9999-12-31 23:59:59.9999999
6	6	Coordinador	Fijo	Completa	3000.00	12 años	67890123F	2025-03-22 10:57:31.9786589	9999-12-31 23:59:59.9999999
7	7	Profesor	Fijo	Parcial	2000.50	8 años	78901234G	2025-03-22 10:57:31.9786589	9999-12-31 23:59:59.9999999
8	8	Profesor	Temporal	Completa	1600.80	4 años	89012345H	2025-03-22 10:57:31.9786589	9999-12-31 23:59:59.9999999
9	9	Secretario	Fijo	Completa	1900.90	6 años	90123456I	2025-03-22 10:57:31.9786589	9999-12-31 23:59:59.9999999
10	10	Profesor	Temporal	Parcial	1400.75	1 año	11234567J	2025-03-22 10:57:31.9786589	9999-12-31 23:59:59.9999999
11	11	J.Estudios	Fijo	Completa	3500.00	15 años	22345678K	2025-03-22 10:57:31.9786589	9999-12-31 23:59:59.9999999
12	12	Profesor	Temporal	Parcial	1750.60	3 años	33456789L	2025-03-22 10:57:31.9786589	9999-12-31 23:59:59.9999999
13	13	Contable	Fijo	Completa	2800.00	9 años	44567890M	2025-03-22 10:57:31.9786589	9999-12-31 23:59:59.9999999
14	14	Profesor	Fijo	Completa	2300.40	11 años	55678901N	2025-03-22 10:57:31.9786589	9999-12-31 23:59:59.9999999
15	15	Profesor	Temporal	Parcial	1550.20	2 años	66789012O	2025-03-22 10:57:31.9786589	9999-12-31 23:59:59.9999999

Aquí cambiamos la fecha por la de hace apenas unos minutos y vemos los cambios.

```

SELECT * FROM contratos
FOR system_time AS OF '2025-03-22T12:36:40'
ORDER BY ID_contrato;
GO

```

100 %

	ID_contrato	Cargo	T_contrato	T_jomada	Sueldo	Antiguedad	profesor_profesor_DNI_profe	SysStartTime	SysEndTime
1	2	Profesor	Temporal	Parcial	4600.50	10 años	23456789B	2025-03-22 11:04:39.8401580	9999-12-31 23:59:59.9999999
2	3	Profesor	Fijo	Completa	2500.75	7 años	34567890C	2025-03-22 10:57:31.9786589	9999-12-31 23:59:59.9999999
3	4	Admin	Fijo	Completa	2700.00	5 años	45678901D	2025-03-22 10:57:31.9786589	9999-12-31 23:59:59.9999999
4	5	Profesor	Temporal	Parcial	1800.30	2 años	56789012E	2025-03-22 10:57:31.9786589	9999-12-31 23:59:59.9999999
5	6	Coordinador	Fijo	Completa	3000.00	12 años	67890123F	2025-03-22 10:57:31.9786589	9999-12-31 23:59:59.9999999
6	7	Profesor	Fijo	Parcial	2000.50	8 años	78901234G	2025-03-22 10:57:31.9786589	9999-12-31 23:59:59.9999999
7	8	Profesor	Temporal	Completa	1600.80	4 años	89012345H	2025-03-22 10:57:31.9786589	9999-12-31 23:59:59.9999999
8	9	Secretario	Fijo	Completa	1900.90	6 años	90123456I	2025-03-22 10:57:31.9786589	9999-12-31 23:59:59.9999999
9	10	Profesor	Temporal	Parcial	1400.75	1 año	11234567J	2025-03-22 10:57:31.9786589	9999-12-31 23:59:59.9999999
10	11	J.Estudios	Fijo	Completa	3500.00	15 años	22345678K	2025-03-22 10:57:31.9786589	9999-12-31 23:59:59.9999999
11	12	Profesor	Temporal	Parcial	1750.60	3 años	33456789L	2025-03-22 10:57:31.9786589	9999-12-31 23:59:59.9999999
12	13	Contable	Fijo	Completa	2800.00	9 años	44567890M	2025-03-22 10:57:31.9786589	9999-12-31 23:59:59.9999999
13	14	Profesor	Fijo	Completa	2300.40	11 años	55678901N	2025-03-22 10:57:31.9786589	9999-12-31 23:59:59.9999999
14	15	Profesor	Temporal	Parcial	1550.20	2 años	66789012O	2025-03-22 10:57:31.9786589	9999-12-31 23:59:59.9999999

FROM

Podemos usar "FROM" para extraer los datos en un periodo de tiempo específico

```

SELECT * FROM contratos
FOR system_time FROM '2025-03-21' TO '2025-03-23'
ORDER BY ID_contrato;
GO

```

```

SELECT * FROM contratos
FOR system_time FROM '2025-03-21' TO '2025-03-23'
ORDER BY ID_contrato;
GO

```

100 %

	ID_contrato	Cargo	T_contrato	T_jomada	Sueldo	Antiguedad	profesor_profesor_DNI_profe	SysStartTime	SysEndTime
1	1	Profesor	Fijo	Completa	2200.50	10 años	12345678A	2025-03-22 10:57:31.9786589	2025-03-22 11:00:26.4551783
2	1	Profesor	Fijo	Completa	2400.75	12 años	12345678A	2025-03-22 11:00:26.4551783	2025-03-22 11:00:26.4780030
3	1	Coordinador	Fijo	Completa	2400.75	12 años	12345678A	2025-03-22 11:00:26.4780030	2025-03-22 11:04:12.2350513
4	1	Coordinador	Fijo	Completa	3500.75	18 años	12345678A	2025-03-22 11:04:12.2350513	2025-03-22 11:04:39.8401580
5	1	Coordinador	Fijo	Completa	3500.75	18 años	12345678A	2025-03-22 11:04:39.8401580	2025-03-22 11:20:33.2694193
6	2	Profesor	Temporal	Parcial	4600.50	10 años	23456789B	2025-03-22 11:04:12.2350513	2025-03-22 11:04:39.8401580
7	2	Profesor	Temporal	Parcial	1600.50	5 años	23456789B	2025-03-22 11:00:26.4551783	2025-03-22 11:04:12.2350513
8	2	Profesor	Temporal	Parcial	1500.00	3 años	23456789B	2025-03-22 10:57:31.9786589	2025-03-22 11:00:26.4551783
9	2	Profesor	Temporal	Parcial	4600.50	10 años	23456789B	2025-03-22 11:04:39.8401580	9999-12-31 23:59:59.9999999
10	3	Profesor	Fijo	Completa	2500.75	7 años	34567890C	2025-03-22 10:57:31.9786589	9999-12-31 23:59:59.9999999
11	4	Admin	Fijo	Completa	2700.00	5 años	45678901D	2025-03-22 10:57:31.9786589	9999-12-31 23:59:59.9999999
12	5	Profesor	Temporal	Parcial	1800.30	2 años	56789012E	2025-03-22 10:57:31.9786589	9999-12-31 23:59:59.9999999
13	6	Coordinador	Fijo	Completa	3000.00	12 años	67890123F	2025-03-22 10:57:31.9786589	9999-12-31 23:59:59.9999999
14	7	Profesor	Fijo	Parcial	2000.50	8 años	78901234G	2025-03-22 10:57:31.9786589	9999-12-31 23:59:59.9999999
15	8	Profesor	Temporal	Completa	1600.80	4 años	89012345H	2025-03-22 10:57:31.9786589	9999-12-31 23:59:59.9999999
16	9	Secretario	Fijo	Completa	1900.90	6 años	90123456I	2025-03-22 10:57:31.9786589	9999-12-31 23:59:59.9999999
17	10	Profesor	Temporal	Parcial	1400.75	1 año	11234567J	2025-03-22 10:57:31.9786589	9999-12-31 23:59:59.9999999

BETWEEN

Si usamos "BETWEEN" en vez de "FROM" lo que nos muestra es los registros activos en ese periodo de tiempo, si durante ese periodo un profesor no tuviese contrato no aparecería:

```

SELECT * FROM contratos
FOR system_time BETWEEN '2025-03-21' AND '2025-03-23'
ORDER BY ID_contrato;
GO

```

```

SELECT * FROM contratos
FOR system_time BETWEEN '2025-03-21' AND '2025-03-23'
ORDER BY ID_contrato;
GO

```

100 %

	ID_contrato	Cargo	T_contrato	T_jomada	Sueldo	Antiguedad	profesor_profesor_DNI_profe	SysStartTime	SysEndTime
1	1	Profesor	Fijo	Completa	2200.50	10 años	12345678A	2025-03-22 10:57:31.9786589	2025-03-22 11:00:26.4551783
2	1	Profesor	Fijo	Completa	2400.75	12 años	12345678A	2025-03-22 11:00:26.4551783	2025-03-22 11:00:26.4780030
3	1	Coordinador	Fijo	Completa	2400.75	12 años	12345678A	2025-03-22 11:00:26.4780030	2025-03-22 11:04:12.2350513
4	1	Coordinador	Fijo	Completa	3500.75	18 años	12345678A	2025-03-22 11:04:12.2350513	2025-03-22 11:04:39.8401580
5	1	Coordinador	Fijo	Completa	3500.75	18 años	12345678A	2025-03-22 11:04:39.8401580	2025-03-22 11:20:33.2694193
6	2	Profesor	Temporal	Parcial	4600.50	10 años	23456789B	2025-03-22 11:04:12.2350513	2025-03-22 11:04:39.8401580
7	2	Profesor	Temporal	Parcial	1600.50	5 años	23456789B	2025-03-22 11:00:26.4551783	2025-03-22 11:04:12.2350513
8	2	Profesor	Temporal	Parcial	1500.00	3 años	23456789B	2025-03-22 10:57:31.9786589	2025-03-22 11:00:26.4551783
9	2	Profesor	Temporal	Parcial	4600.50	10 años	23456789B	2025-03-22 11:04:39.8401580	9999-12-31 23:59:59.9999999
10	3	Profesor	Fijo	Completa	2500.75	7 años	34567890C	2025-03-22 10:57:31.9786589	9999-12-31 23:59:59.9999999
11	4	Admin	Fijo	Completa	2700.00	5 años	45678901D	2025-03-22 10:57:31.9786589	9999-12-31 23:59:59.9999999
12	5	Profesor	Temporal	Parcial	1800.30	2 años	56789012E	2025-03-22 10:57:31.9786589	9999-12-31 23:59:59.9999999
13	6	Coordinador	Fijo	Completa	3000.00	12 años	67890123F	2025-03-22 10:57:31.9786589	9999-12-31 23:59:59.9999999
14	7	Profesor	Fijo	Parcial	2000.50	8 años	78901234G	2025-03-22 10:57:31.9786589	9999-12-31 23:59:59.9999999
15	8	Profesor	Temporal	Completa	1600.80	4 años	89012345H	2025-03-22 10:57:31.9786589	9999-12-31 23:59:59.9999999
16	9	Secretario	Fijo	Completa	1900.90	6 años	90123456I	2025-03-22 10:57:31.9786589	9999-12-31 23:59:59.9999999
17	10	Profesor	Temporal	Parcial	1400.75	1 año	11234567J	2025-03-22 10:57:31.9786589	9999-12-31 23:59:59.9999999

CONTAINED IN

Selecciona los registros que estuvieron activos COMPLETAMENTE dentro del periodo especificado, es decir, que al inicio del periodo estaban y que al final del mismo también.

```

SELECT * FROM contratos
FOR system_time CONTAINED IN ('2025-03-21','2025-03-23')

```

```
WHERE T_contrato = 'Temporal';
GO
```

The screenshot shows a SQL Server Management Studio window. In the top pane, there is a code editor with the following T-SQL script:

```
SELECT * FROM contratos
FOR system_time CONTAINED IN ('2025-03-21', '2025-03-23')
WHERE T_contrato = 'Temporal';
GO
```

In the bottom pane, there are two tabs: "Results" and "Messages". The "Results" tab is selected and displays a table with the following data:

	ID_contrato	Cargo	T_contrato	T_jomada	Sueldo	Antiguedad	profesor_profesor_DNI_profe	SysStartTime	SysEndTime
1	2	Profesor	Temporal	Parcial	1500.00	3 años	23456789B	2025-03-22 10:57:31.9786589	2025-03-22 11:00:26.4551783
2	2	Profesor	Temporal	Parcial	1600.50	5 años	23456789B	2025-03-22 11:00:26.4551783	2025-03-22 11:04:12.2350513
3	2	Profesor	Temporal	Parcial	4600.50	10 años	23456789B	2025-03-22 11:04:12.2350513	2025-03-22 11:04:39.8401580

Limitaciones

1. Las consultas temporales sobre un Servidor Vinculado no son compatibles.
2. La tabla de historial no puede tener restricciones (clave primaria, clave foránea, restricciones de tabla o de columna).
3. Las sentencias INSERT y UPDATE no pueden hacer referencia a las columnas del período SYSTEM_TIME.
4. El comando TRUNCATE TABLE no es compatible mientras SYSTEM_VERSIONING está activado.
5. No está permitido modificar directamente los datos en una tabla de historial.
6. No se permiten los triggers INSTEAD OF en ninguna de las tablas.
7. El uso de tecnologías de replicación está limitado.

11.- Tablas en memoria

Las **tablas en memoria** (In-Memory OLTP) en SQL Server están diseñadas para mejorar el rendimiento almacenando datos en memoria en lugar de en disco. Estas tablas utilizan un motor optimizado para transacciones rápidas y minimizan la contención de bloqueos, lo que las hace ideales para escenarios de alta concurrencia.

Características principales:

- No generan bloqueos en las transacciones (debido a su arquitectura sin bloqueo).
- Son significativamente más rápidas que las tablas tradicionales en disco.
- Pueden ser **duraderas** (persisten en disco) o **no duraderas** (solo existen en memoria temporalmente).
- Requieren un **grupo de archivos en memoria** en la base de datos.

Para probar el funcionamiento, vamos a crear la tabla ReservaClasesPracticas_Disco y ReservaClasesPracticas_Memoria, haremos una prueba insertando 50.000 registros (aunque son pocos, seguramente ya se note la diferencia).

Primero creamos el filegroup para memoria optimizada:

```
ALTER DATABASE AutoescuelaP
ADD FILEGROUP AutoescuelaP_mod CONTAINS MEMORY_OPTIMIZED_DATA;
GO
```

Añadimos el archivo:

```
ALTER DATABASE AutoescuelaP ADD FILE(
    NAME = 'AutoescuelaP_mod',
    FILENAME = 'C:\Data\AutoescuelaP_mod.ndf'
```

```
)
TO FILEGROUP AutoescuelaP_mod;
GO
```

Creamos la tabla en disco y la tabla en memoria:

****MUY IMPORTANTE acordarse siempre de Filegroup MEMPRY_OPTIMIZED_DATA para la tabla en memoria.**

```
-- Tabla en disco
CREATE TABLE dbo.ReservasClasesPracticas_Disco
(
    ID INT IDENTITY PRIMARY KEY,
    Alumno NVARCHAR(20),
    Fecha DATETIME,
    Vehiculo NVARCHAR(20),
    Estado NVARCHAR(10)
);

-- Tabla en memoria
-- IMPORTANTE acordarse de: FileGroup MEMORY_OPTIMIZED_DATA
ALTER DATABASE CURRENT ADD FILEGROUP PracticasFG CONTAINS MEMORY_OPTIMIZED_DATA;
ALTER DATABASE CURRENT ADD FILE (NAME='Practicas_Data', FILENAME='C:\Data\Practicas_Data') TO FILEGI

CREATE TABLE dbo.ReservasClasesPracticas_Memoria(
    ID INT IDENTITY PRIMARY KEY NONCLUSTERED,
    Alumno NVARCHAR(20),
    Fecha DATETIME,
    Vehiculo NVARCHAR(20),
    Estado NVARCHAR(10)
) WITH (MEMORY_OPTIMIZED = ON, DURABILITY = SCHEMA_AND_DATA);
GO
```

Creamos procedimiento que, dándole la ubicacion, inserte 50.000 registros en la tabla en memoria o en la tabla en disco y nos devuelva el tiempo que tarda en realizar la operación:

```
CREATE OR ALTER PROCEDURE GenerarReservasPracticas
    @TipoTabla NVARCHAR(10) = 'DISCO' -- 'DISCO' o 'MEMORIA'
AS
BEGIN
    DECLARE @contador INT = 1;
    DECLARE @Alumno NVARCHAR(20);
    DECLARE @Fecha DATETIME;
    DECLARE @Vehiculo NVARCHAR(20);
    DECLARE @Estado NVARCHAR(10);
    DECLARE @Inicio DATETIME = GETDATE();

    WHILE @contador <= 50000
    BEGIN
        SET @Alumno = (SELECT TOP 1 Nombre FROM (VALUES ('Borja'), ('Ana'), ('Carlos'), ('Marta'),
        ('Javier'), ('Lucía'), ('David')) AS N(Nombre) ORDER BY NEWID());
        SET @Fecha = DATEADD(DAY, FLOOR(RAND() * 30), GETDATE());
        SET @Vehiculo = RIGHT('0000' + CAST(FLOOR(RAND() * 10000) AS NVARCHAR(4)), 4) + '-' +
            CHAR(65 + FLOOR(RAND() * 26)) +
            CHAR(65 + FLOOR(RAND() * 26)) +
            CHAR(65 + FLOOR(RAND() * 26));
    END
END
```

```

SET @Estado = (SELECT TOP 1 Estado FROM (VALUES ('Pendiente'), ('Completada'), ('Cancelada'))
AS E(Estado) ORDER BY NEWID());

IF @TipoTabla = 'DISCO'
    INSERT INTO dbo.ReservasClasesPracticas_Disco (Alumno, Fecha, Vehiculo, Estado)
    VALUES (@Alumno, @Fecha, @Vehiculo, @Estado);
ELSE
    INSERT INTO dbo.ReservasClasesPracticas_Memoria (Alumno, Fecha, Vehiculo, Estado)
    VALUES (@Alumno, @Fecha, @Vehiculo, @Estado);

SET @contador += 1;
END;

DECLARE @TiempoEjecucion INT = DATEDIFF(MILLISECOND, @Inicio, GETDATE());
SELECT CONCAT('Tiempo de ejecución: ', @TiempoEjecucion, ' ms') AS Resultado;
END;
GO

```

Realizamos ambas pruebas:

Script	Time (ms)
-- Insertar en disco EXEC GenerarReservasPracticas @TipoTabla = 'DISCO';	67900 ms
-- Insertar en memoria EXEC GenerarReservasPracticas @TipoTabla = 'MEMORIA';	64413 ms

Como podemos observar, aunque la operación ha sido con pocos datos, ya se nota la diferencia del disco a memoria.

DURABILITY = SCHEMA_AND_DATA

Ante una caída del sistema o algún problema, se recuperará el esquema y los datos, con la opción SCHEMA_ONLY solo se recuperará el esquema.