



TAREA3: Modelado -Adm. SQL SERVER

1.- Análisis previo

1.1.- Especificaciones y requerimientos

1.2.- Supuestos semánticos

2.- Modelo lógico

3.- Modelo relacional

4.- Modelo físico

5.- Script

6.- 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

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 adjudicándosele 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.

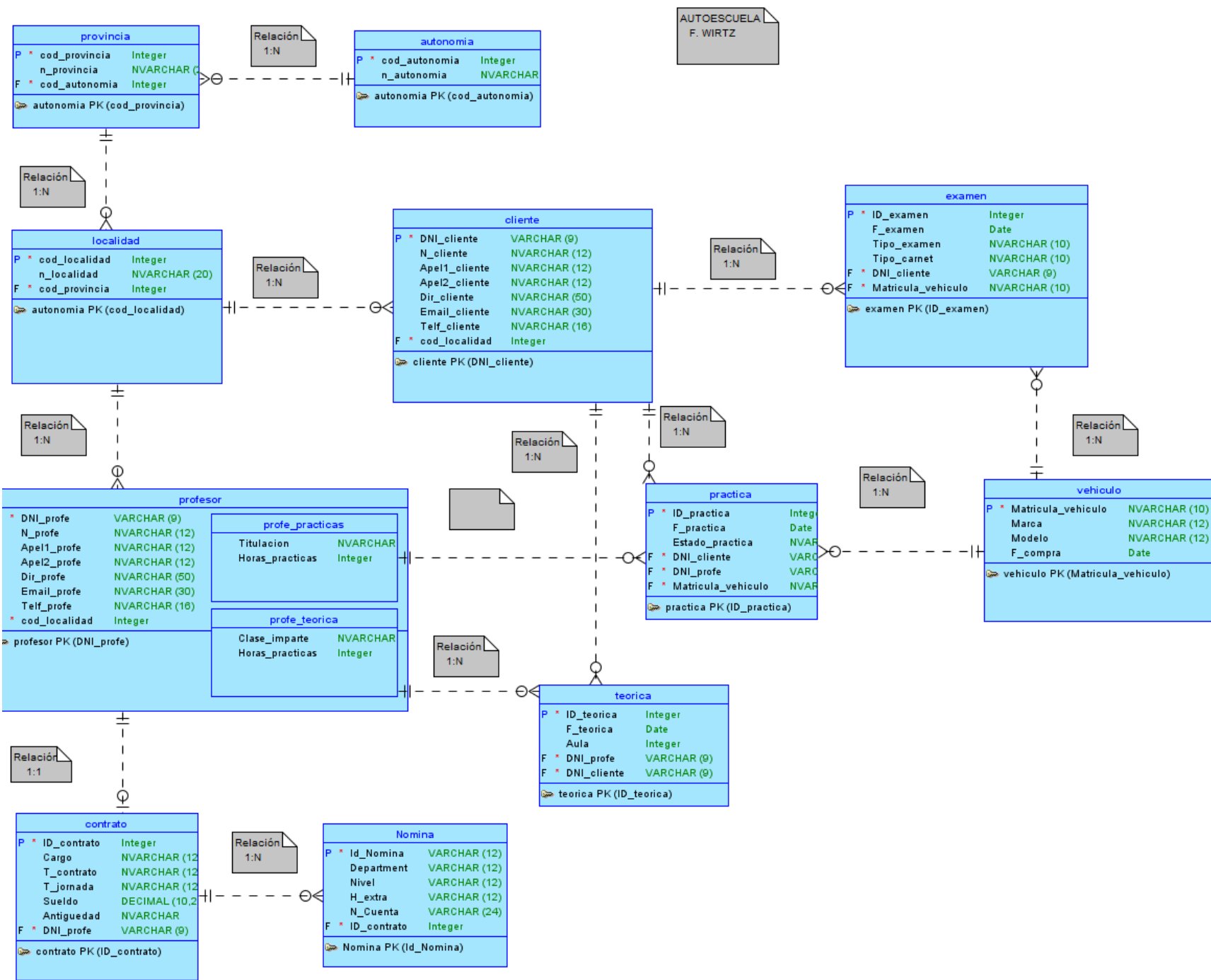
Un alumno podrá recibir clases teóricas y prácticas a la vez, ya que puede estar pendiente de examen teórico e ir empezando a hacer prácticas. Además un alumno podrá presentarse a varios exámenes pero cada examen solo tendrá un alumno (aunque en el aula o en en vehículo del examen practico concurren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

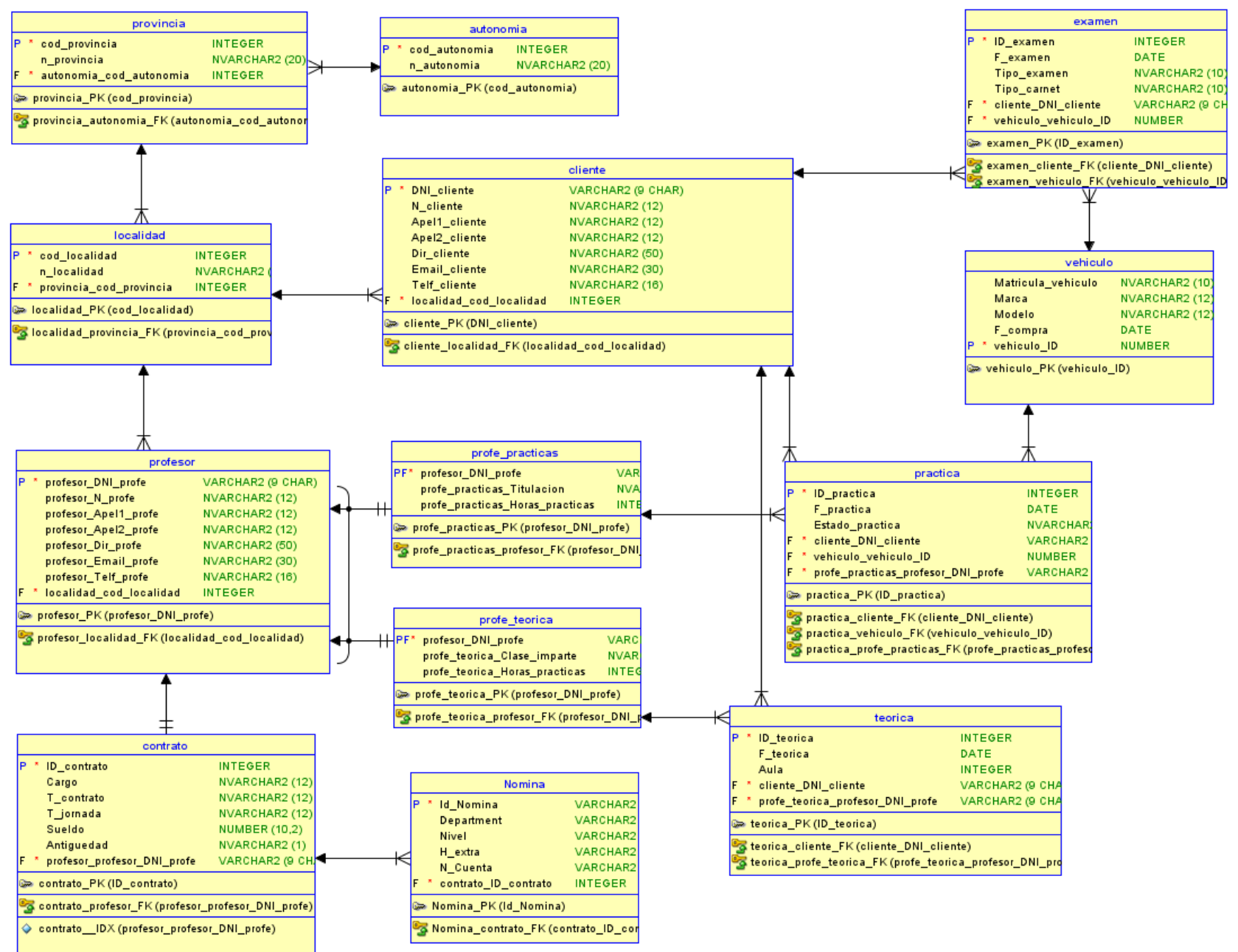
2.- Modelo lógico

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 practica 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.

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



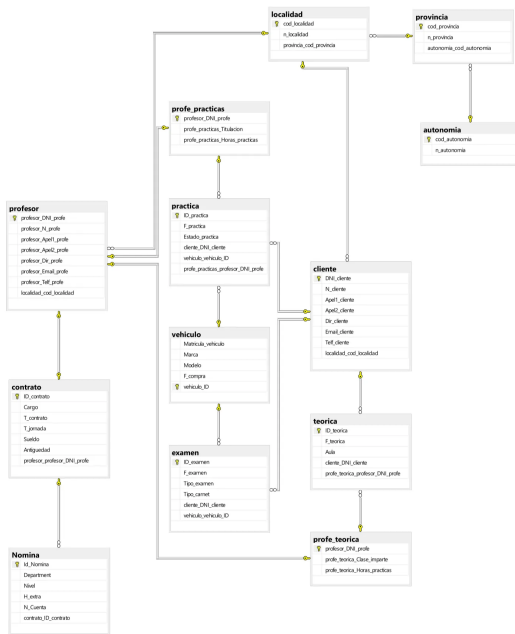
4.- Modelo físico

Una vez hayamos hecho el modelado lógico en Datamodeler y lo hayamos pasado a relacional, generamos 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 autoescuela;
GO
CREATE DATABASE autoescuela;
GO
Use autoescuela;
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:



5.- Script

```
CREATE DATABASE autoescuela;
GO
USE autoescuela;
GO
```

```
CREATE TABLE autonomia
(
    cod_autonomia INTEGER NOT NULL ,
    n_autonomia NVARCHAR (20)
)

GO
```

```
ALTER TABLE autonomia ADD CONSTRAINT autonomia_PK PRIMARY KEY CLUSTERED (cod_autonomia)
WITH (
    ALLOW_PAGE_LOCKS = ON ,
    ALLOW_ROW_LOCKS = ON )
GO
```

```
CREATE TABLE cliente
(
    DNI_cliente VARCHAR (9) NOT NULL ,
    N_cliente NVARCHAR (12) ,
    Apellido_cliente NVARCHAR (12) ,
    Apellido2_cliente NVARCHAR (12) ,
    Dir_cliente NVARCHAR (50) ,
    Email_cliente NVARCHAR (30) ,
    Telf_cliente NVARCHAR (16) ,
    localidad_cod_localidad INTEGER NOT NULL
)

GO
```

```
ALTER TABLE cliente ADD CONSTRAINT cliente_PK PRIMARY KEY CLUSTERED (DNI_cliente)
WITH (
    ALLOW_PAGE_LOCKS = ON ,
    ALLOW_ROW_LOCKS = ON )
GO
```

```

CREATE TABLE contrato
(
    ID_contrato INTEGER NOT NULL ,
    Cargo NVARCHAR (12) ,
    T_contrato NVARCHAR (12) ,
    T_jornada NVARCHAR (12) ,
    Sueldo DECIMAL (10,2) ,
    Antigüedad NVARCHAR (9),
    profesor_profesor_DNI_profe VARCHAR (9) NOT NULL
)
GO

```

```

CREATE UNIQUE NONCLUSTERED INDEX
    contrato__IDX ON contrato
(
    profesor_profesor_DNI_profe
)
GO

```

```

ALTER TABLE contrato ADD CONSTRAINT contrato_PK PRIMARY KEY CLUSTERED (ID_contrato)
WITH (
    ALLOW_PAGE_LOCKS = ON ,
    ALLOW_ROW_LOCKS = ON )
GO

```

```

CREATE TABLE examen
(
    ID_examen INTEGER NOT NULL ,
    F_examen DATE ,
    Tipo_examen NVARCHAR (10) ,
    Tipo_carnet NVARCHAR (10) ,
    cliente_DNI_cliente VARCHAR (9) NOT NULL ,
    vehiculo_vehiculo_ID NUMERIC (28) NOT NULL
)
GO

```

```

ALTER TABLE examen ADD CONSTRAINT examen_PK PRIMARY KEY CLUSTERED (ID_examen)
WITH (
    ALLOW_PAGE_LOCKS = ON ,
    ALLOW_ROW_LOCKS = ON )
GO

```

```

CREATE TABLE localidad
(
    cod_localidad INTEGER NOT NULL ,
    n_localidad NVARCHAR (20) ,
    provincia_cod_provincia INTEGER NOT NULL
)
GO

```

```

ALTER TABLE localidad ADD CONSTRAINT localidad_PK PRIMARY KEY CLUSTERED (cod_localidad)
WITH (
    ALLOW_PAGE_LOCKS = ON ,
    ALLOW_ROW_LOCKS = ON )
GO

```

```

CREATE TABLE Nomina
(
    Id_Nomina VARCHAR (12) NOT NULL ,
    Department VARCHAR (12) ,
    Nivel VARCHAR (12) ,
    H_extra VARCHAR (12) ,
    N_Cuenta VARCHAR (24) ,
    contrato_ID_contrato INTEGER NOT NULL
)
GO

ALTER TABLE Nomina ADD CONSTRAINT Nomina_PK PRIMARY KEY CLUSTERED (Id_Nomina)
WITH (
    ALLOW_PAGE_LOCKS = ON ,
    ALLOW_ROW_LOCKS = ON )
GO

CREATE TABLE practica
(
    ID_practica INTEGER NOT NULL ,
    F_practica DATE ,
    Estado_practica NVARCHAR (12) ,
    cliente_DNI_cliente VARCHAR (9) NOT NULL ,
    vehiculo_vehiculo_ID NUMERIC (28) NOT NULL ,
    profe_practicas_profesor_DNI_profe VARCHAR (9) NOT NULL
)
GO

ALTER TABLE practica ADD CONSTRAINT practica_PK PRIMARY KEY CLUSTERED (ID_practica)
WITH (
    ALLOW_PAGE_LOCKS = ON ,
    ALLOW_ROW_LOCKS = ON )
GO

CREATE TABLE profe_practicas
(
    profesor_DNI_profe VARCHAR (9) NOT NULL ,
    profe_practicas_Titulacion NVARCHAR (8) ,
    profe_practicas_Horas_practicas INTEGER
)
GO

ALTER TABLE profe_practicas ADD CONSTRAINT profe_practicas_PK PRIMARY KEY CLUSTERED (profesor_DNI_profe)
WITH (
    ALLOW_PAGE_LOCKS = ON ,
    ALLOW_ROW_LOCKS = ON )
GO

CREATE TABLE profe_teorica
(
    profesor_DNI_profe VARCHAR (9) NOT NULL ,
    profe_teorica_Clasa_imparte NVARCHAR (18) ,
    profe_teorica_Horas_practicas INTEGER
)
GO

ALTER TABLE profe_teorica ADD CONSTRAINT profe_teorica_PK PRIMARY KEY CLUSTERED (profesor_DNI_profe)
WITH (
    ALLOW_PAGE_LOCKS = ON ,
    ALLOW_ROW_LOCKS = ON )

```

GO

```
CREATE TABLE profesor
(
    profesor_DNI_profe VARCHAR (9) NOT NULL ,
    profesor_N_profe NVARCHAR (12) ,
    profesor_Apel1_profe NVARCHAR (12) ,
    profesor_Apel2_profe NVARCHAR (12) ,
    profesor_Dir_profe NVARCHAR (50) ,
    profesor_Email_profe NVARCHAR (40) ,
    profesor_Telf_profe NVARCHAR (16) ,
    localidad_cod_localidad INTEGER NOT NULL
)
```

GO

```
ALTER TABLE profesor ADD CONSTRAINT profesor_PK PRIMARY KEY CLUSTERED (profesor_DNI_profe)
WITH (
    ALLOW_PAGE_LOCKS = ON ,
    ALLOW_ROW_LOCKS = ON )
```

GO

```
CREATE TABLE provincia
(
    cod_provincia INTEGER NOT NULL ,
    n_provincia NVARCHAR (20) ,
    autonomia_cod_autonomia INTEGER NOT NULL
)
```

GO

```
ALTER TABLE provincia ADD CONSTRAINT provincia_PK PRIMARY KEY CLUSTERED (cod_provincia)
WITH (
    ALLOW_PAGE_LOCKS = ON ,
    ALLOW_ROW_LOCKS = ON )
```

GO

```
CREATE TABLE teorica
(
    ID_teorica INTEGER NOT NULL ,
    F_teorica DATE ,
    Aula INTEGER ,
    cliente_DNI_cliente VARCHAR (9) NOT NULL ,
    profe_teorica_profesor_DNI_profe VARCHAR (9) NOT NULL
)
```

GO

```
ALTER TABLE teorica ADD CONSTRAINT teorica_PK PRIMARY KEY CLUSTERED (ID_teorica)
WITH (
    ALLOW_PAGE_LOCKS = ON ,
    ALLOW_ROW_LOCKS = ON )
```

GO

```
CREATE TABLE vehiculo
(
    Matricula_vehiculo NVARCHAR (10) ,
    Marca NVARCHAR (12) ,
    Modelo NVARCHAR (12) ,
    F_compra DATE ,
    vehiculo_ID NUMERIC (28) NOT NULL IDENTITY NOT FOR REPLICATION
)
```

GO


```
ALTER TABLE vehiculo ADD CONSTRAINT vehiculo_PK PRIMARY KEY CLUSTERED (vehiculo_ID)
    WITH (
        ALLOW_PAGE_LOCKS = ON ,
        ALLOW_ROW_LOCKS = ON )
GO
```

```
ALTER TABLE cliente
    ADD CONSTRAINT cliente_localidad_FK FOREIGN KEY
    (
        localidad_cod_localidad
    )
    REFERENCES localidad
    (
        cod_localidad
    )
    ON DELETE NO ACTION
    ON UPDATE NO ACTION
GO
```

```
ALTER TABLE contrato
    ADD CONSTRAINT contrato_profesor_FK FOREIGN KEY
    (
        profesor_profesor_DNI_profe
    )
    REFERENCES profesor
    (
        profesor_DNI_profe
    )
    ON DELETE NO ACTION
    ON UPDATE NO ACTION
GO
```

```
ALTER TABLE examen
    ADD CONSTRAINT examen_cliente_FK FOREIGN KEY
    (
        cliente_DNI_cliente
    )
    REFERENCES cliente
    (
        DNI_cliente
    )
    ON DELETE NO ACTION
    ON UPDATE NO ACTION
GO
```

```
ALTER TABLE examen
    ADD CONSTRAINT examen_vehiculo_FK FOREIGN KEY
    (
        vehiculo_vehiculo_ID
    )
    REFERENCES vehiculo
    (
        vehiculo_ID
    )
    ON DELETE NO ACTION
    ON UPDATE NO ACTION
GO
```

```
ALTER TABLE localidad
```



```

ADD CONSTRAINT localidad_provincia_FK FOREIGN KEY
(
    provincia_cod_provincia
)
REFERENCES provincia
(
    cod_provincia
)
ON DELETE NO ACTION
ON UPDATE NO ACTION
GO

ALTER TABLE Nomina
ADD CONSTRAINT Nomina_contrato_FK FOREIGN KEY
(
    contrato_ID_contrato
)
REFERENCES contrato
(
    ID_contrato
)
ON DELETE NO ACTION
ON UPDATE NO ACTION
GO

ALTER TABLE practica
ADD CONSTRAINT practica_cliente_FK FOREIGN KEY
(
    cliente_DNI_cliente
)
REFERENCES cliente
(
    DNI_cliente
)
ON DELETE NO ACTION
ON UPDATE NO ACTION
GO

ALTER TABLE practica
ADD CONSTRAINT practica_profe_practicas_FK FOREIGN KEY
(
    profe_practicas_profesor_DNI_profe
)
REFERENCES profe_practicas
(
    profesor_DNI_profe
)
ON DELETE NO ACTION
ON UPDATE NO ACTION
GO

ALTER TABLE practica
ADD CONSTRAINT practica_vehiculo_FK FOREIGN KEY
(
    vehiculo_vehiculo_ID
)
REFERENCES vehiculo
(
    vehiculo_ID
)

```

```

        ON DELETE NO ACTION
        ON UPDATE NO ACTION
GO

ALTER TABLE profe_practicas
    ADD CONSTRAINT profe_practicas_profesor_FK FOREIGN KEY
    (
        profesor_DNI_profe
    )
    REFERENCES profesor
    (
        profesor_DNI_profe
    )
    ON DELETE NO ACTION
    ON UPDATE NO ACTION
GO

ALTER TABLE profe_teorica
    ADD CONSTRAINT profe_teorica_profesor_FK FOREIGN KEY
    (
        profesor_DNI_profe
    )
    REFERENCES profesor
    (
        profesor_DNI_profe
    )
    ON DELETE NO ACTION
    ON UPDATE NO ACTION
GO

ALTER TABLE profesor
    ADD CONSTRAINT profesor_localidad_FK FOREIGN KEY
    (
        localidad_cod_localidad
    )
    REFERENCES localidad
    (
        cod_localidad
    )
    ON DELETE NO ACTION
    ON UPDATE NO ACTION
GO

ALTER TABLE provincia
    ADD CONSTRAINT provincia_autonomia_FK FOREIGN KEY
    (
        autonomia_cod_autonomia
    )
    REFERENCES autonomia
    (
        cod_autonomia
    )
    ON DELETE NO ACTION
    ON UPDATE NO ACTION
GO

ALTER TABLE teorica
    ADD CONSTRAINT teorica_cliente_FK FOREIGN KEY
    (
        cliente_DNI_cliente

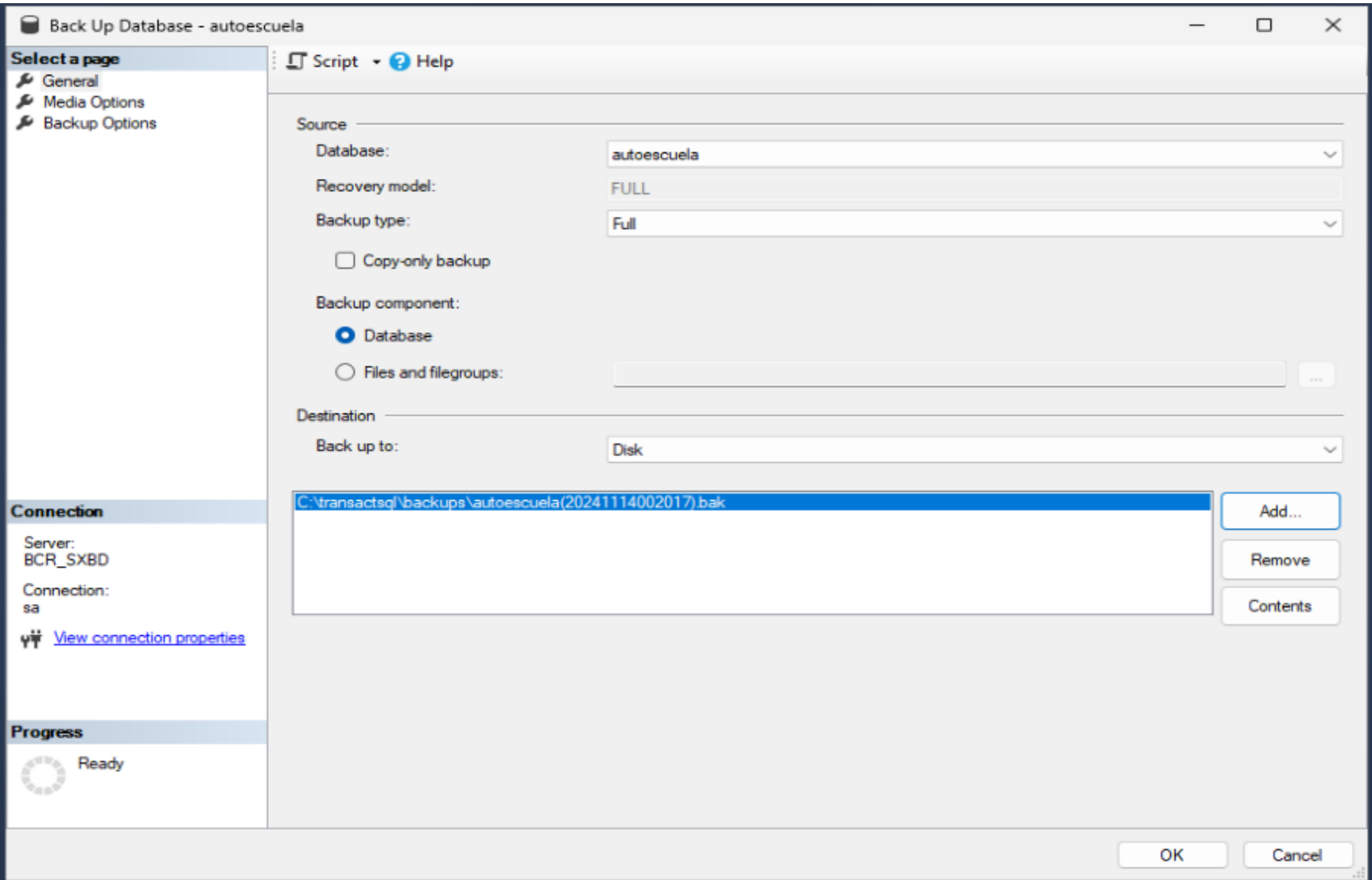
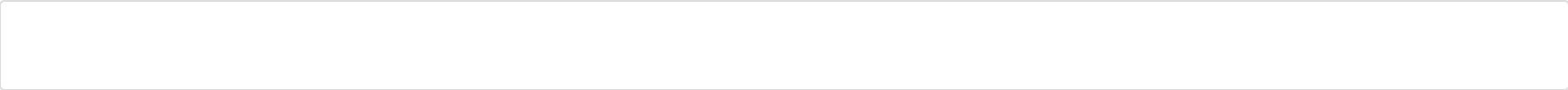
```

```
)
REFERENCES cliente
(
    DNI_cliente
)
ON DELETE NO ACTION
ON UPDATE NO ACTION
GO

ALTER TABLE teorica
ADD CONSTRAINT teorica_profe_teorica_FK FOREIGN KEY
(
    profe_teorica_profesor_DNI_profe
)
REFERENCES profe_teorica
(
    profesor_DNI_profe
)
ON DELETE NO ACTION
ON UPDATE NO ACTION
GO
```

6.- 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.

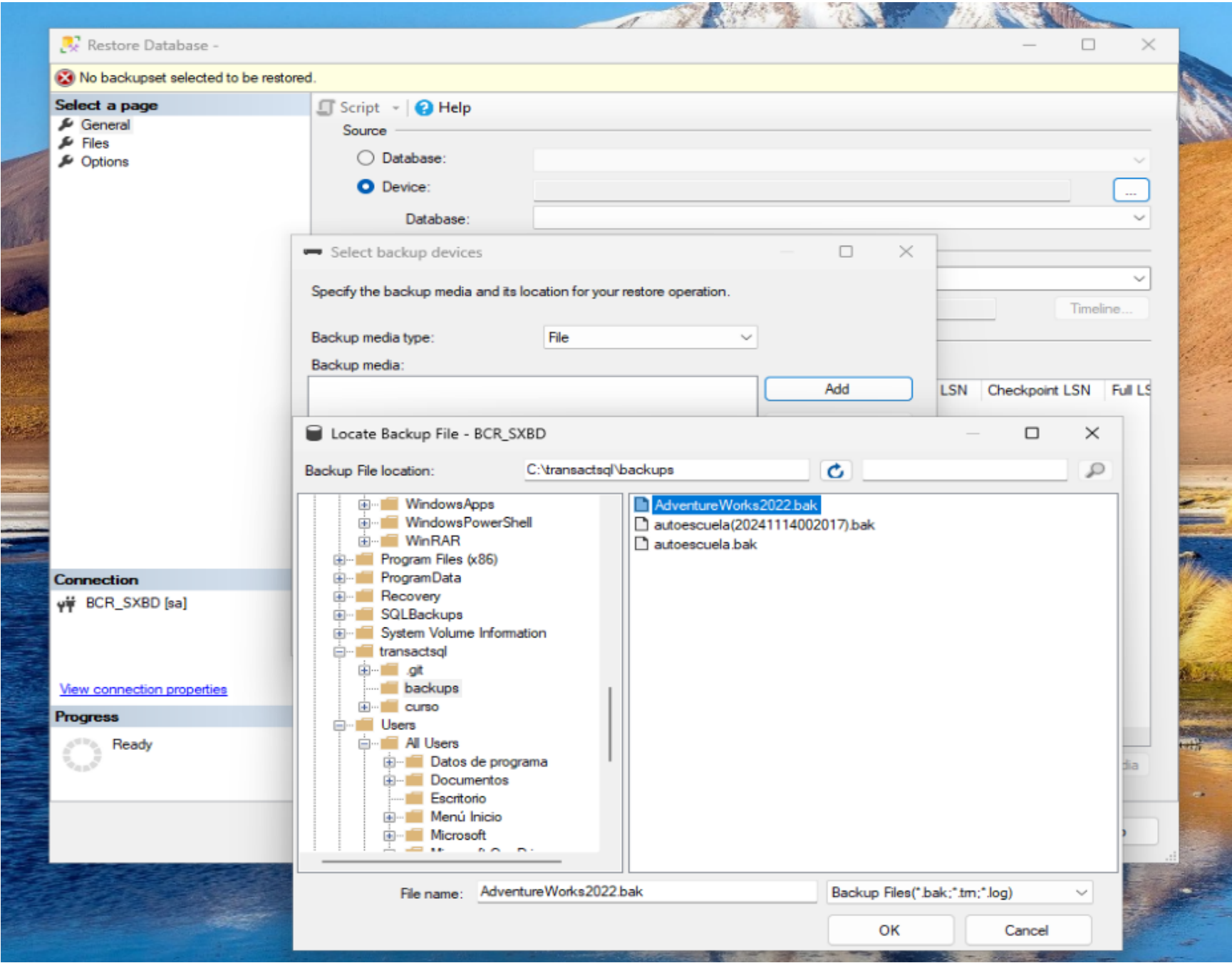
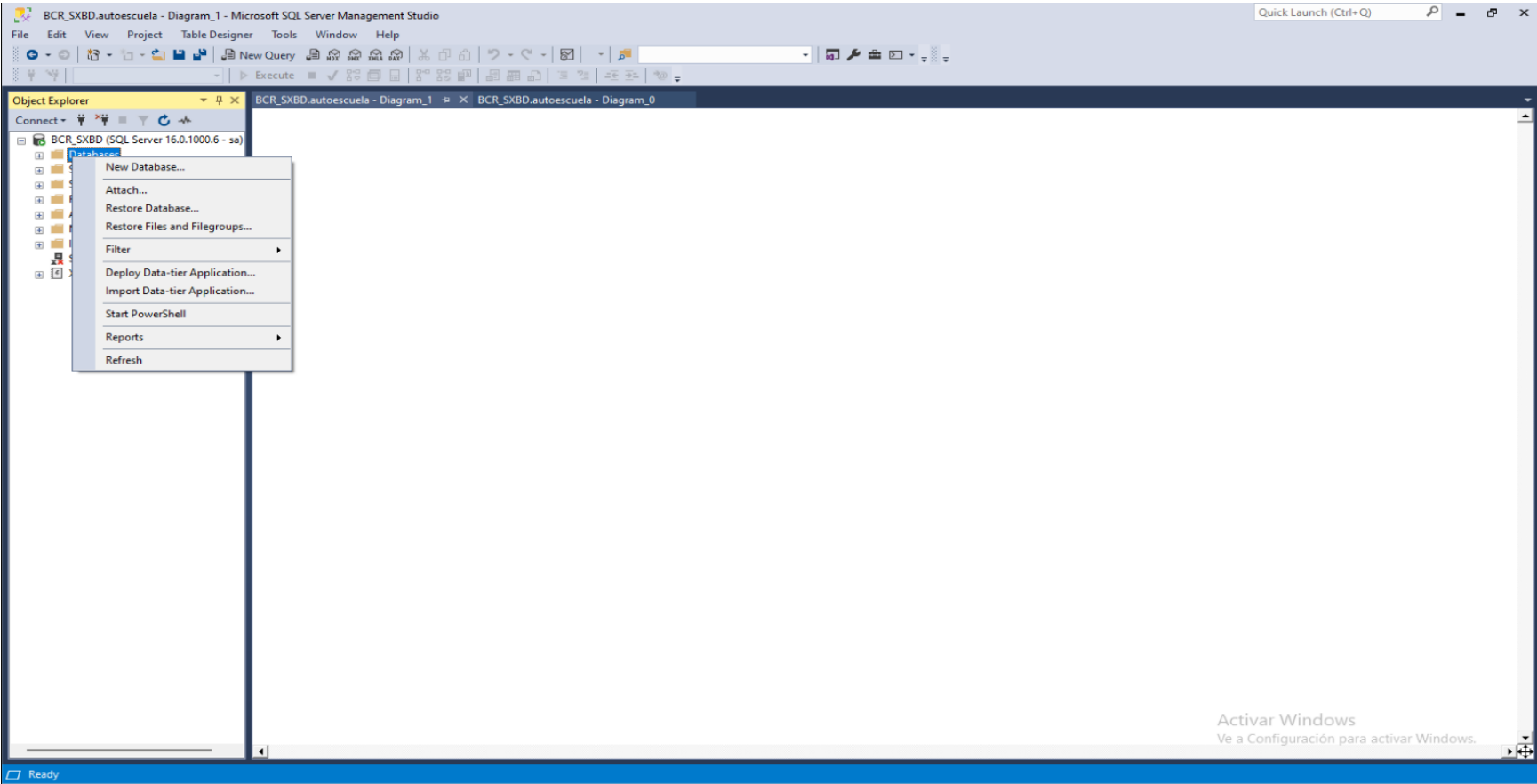


Instalación base de datos de ejemplo

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

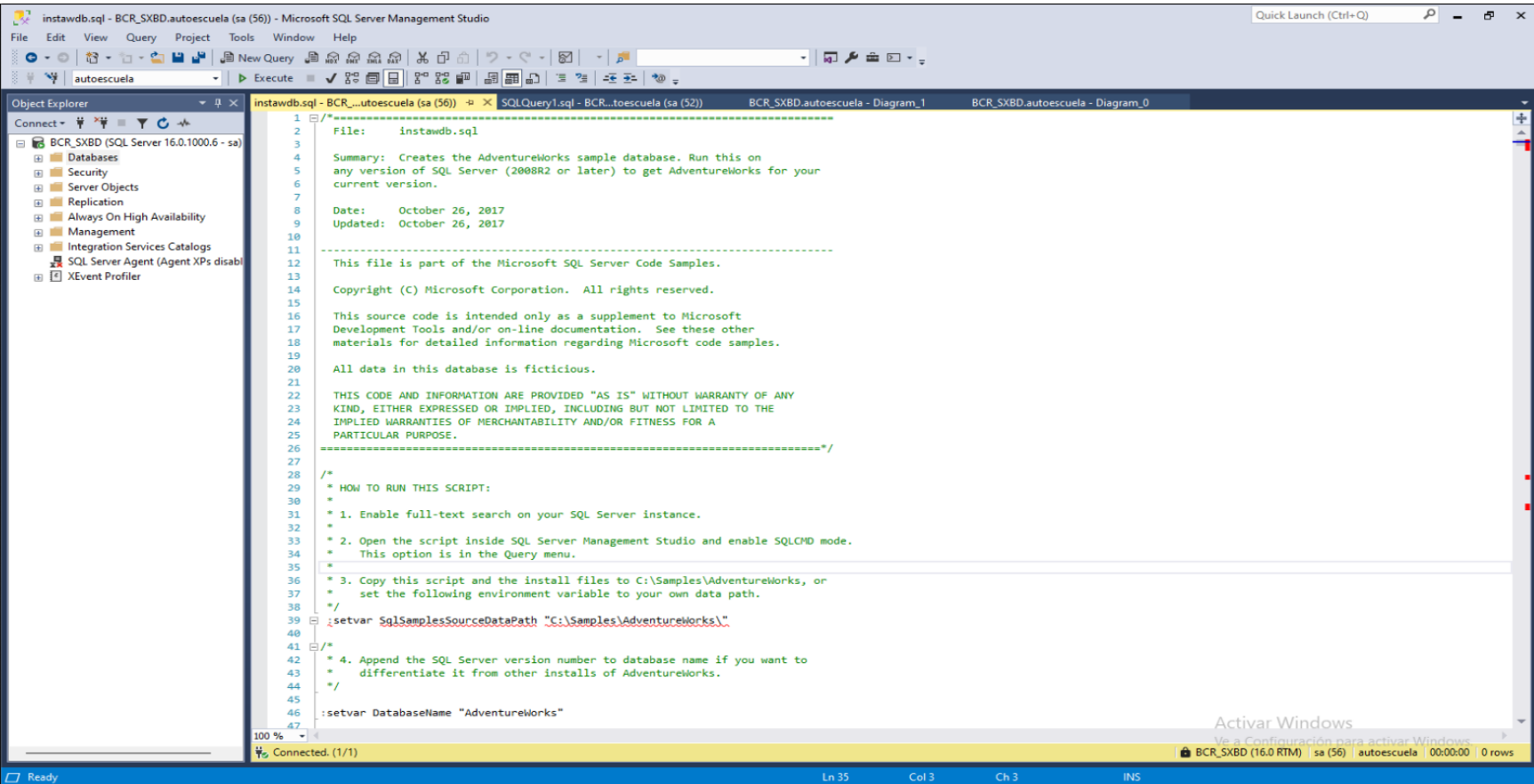
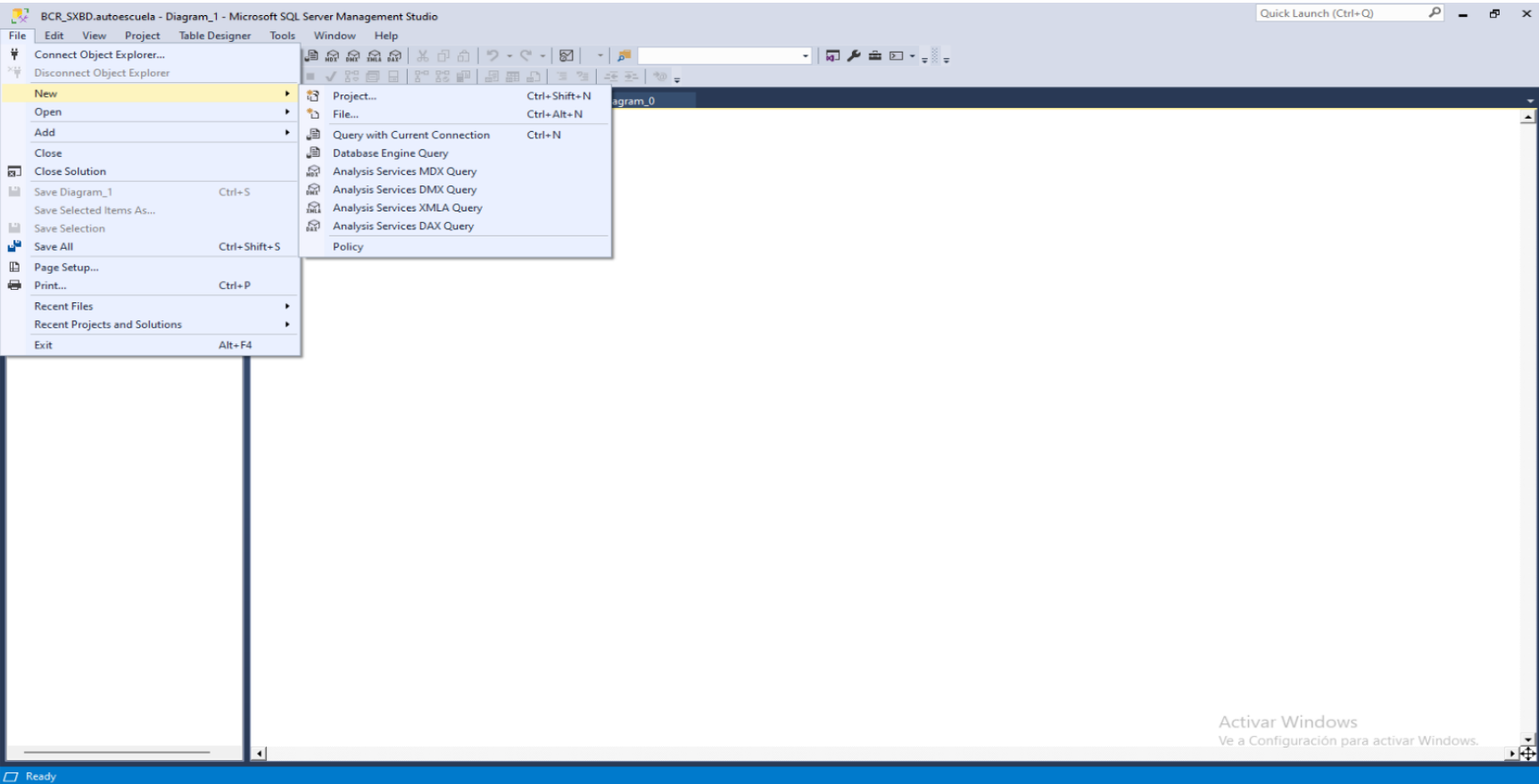
Restore Backup desde SSMS

Comenzamos por realizar un backup directamente desde SSMS, una vez hayamos descargado el archivo .bak, vamos a “Databases” en la barra de navegación laterar, presionamos con el boton 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 qwery y una vez abierto lo ejecutamos



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 Powershel y ejecutamos:

```
Restore-Sqldatabase -Serverinstance "localhost" -Database Adventureworks -Backupfile "C:\
```

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

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
```

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 boton 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 mas comunes es que SSMS no tenga permisos suficientes para acceder a la carpeta donde se encuentran los archivos.

