TAREA3: Modelado -Adm. SQL SERVER

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 concurran mas alumnos) puesto que el examen es algo personal.

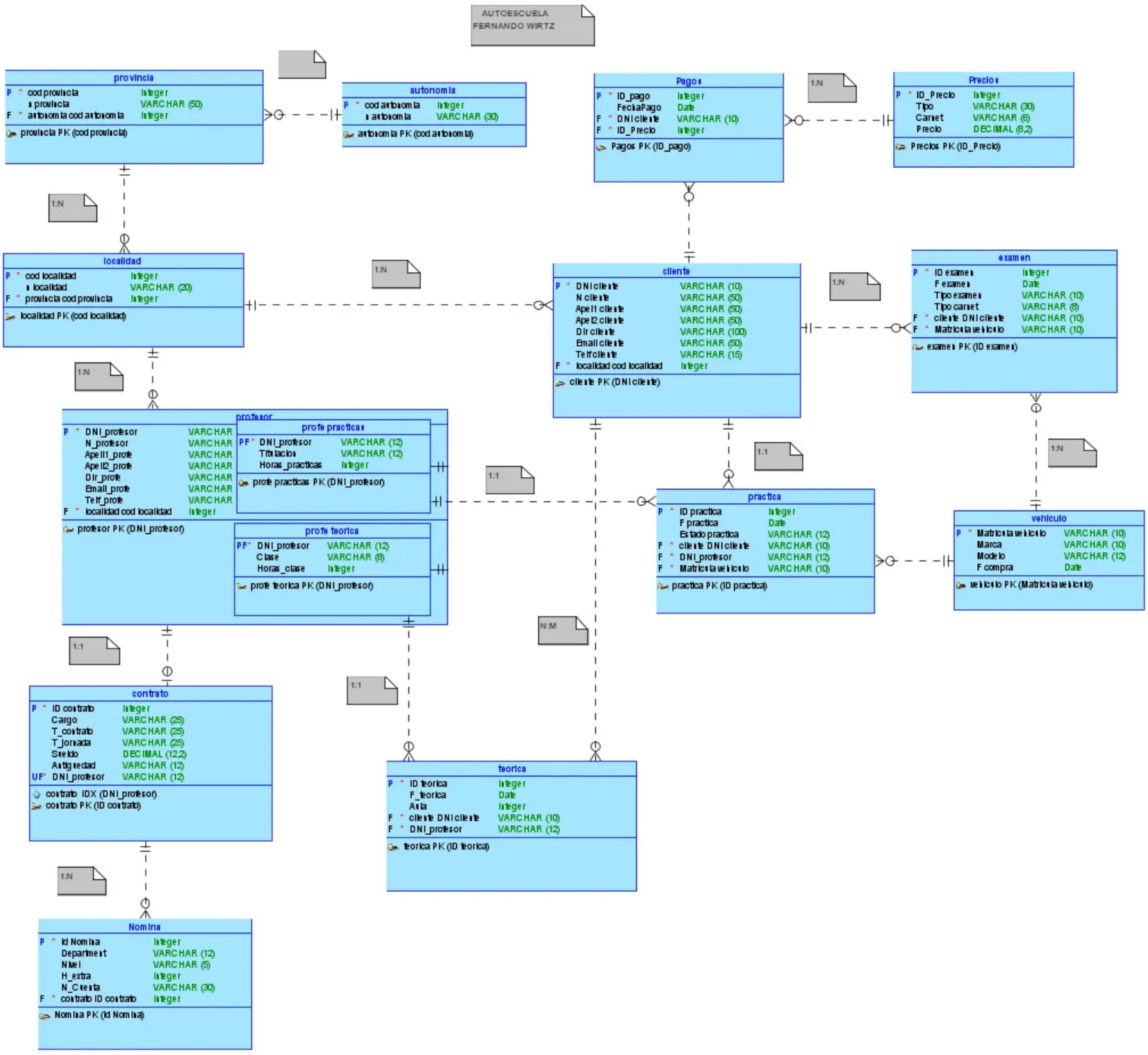
Un vehículo podrá realizar varias prácticas pero no podrá realizarse una práctica sin un vehículo, a su vez, en un examen podrá utilizarse un vehículo y un vehículo podrá ser utilizado en un examen.

2.- Modelo lógico

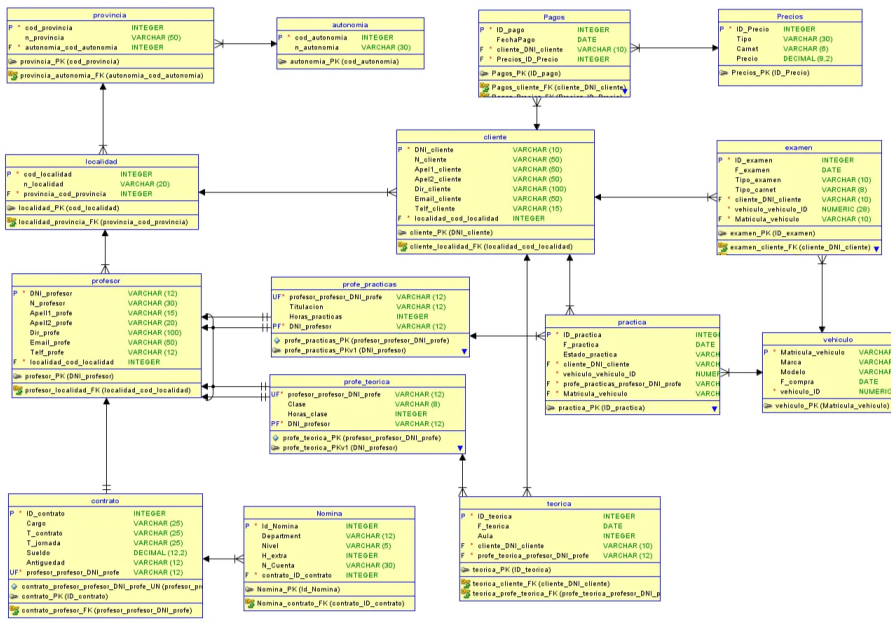
Hemos optado por una especialización con el tipo de profesor, ya que cada profesor imparte clases distintas y un profesor de prácticas no podrá impartir clases teóricas y viceversa, además de esta manera asignamos a cada practica y teoría un profesor de ese departamento. Sin embargo, pensamos en hacer 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 examenes 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 extraido las tablas nómina y contrato de la tabla trabajador y de la misma manera hemos extraido 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 AutoescuelaWirtz;
GO
CREATE DATABASE AutoescuelaWirtz;
GO
Use AutoescuelaWirtz;
GO
```

Ahora ya solo nos queda ejecutar la qwery 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 ( prof

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),
```

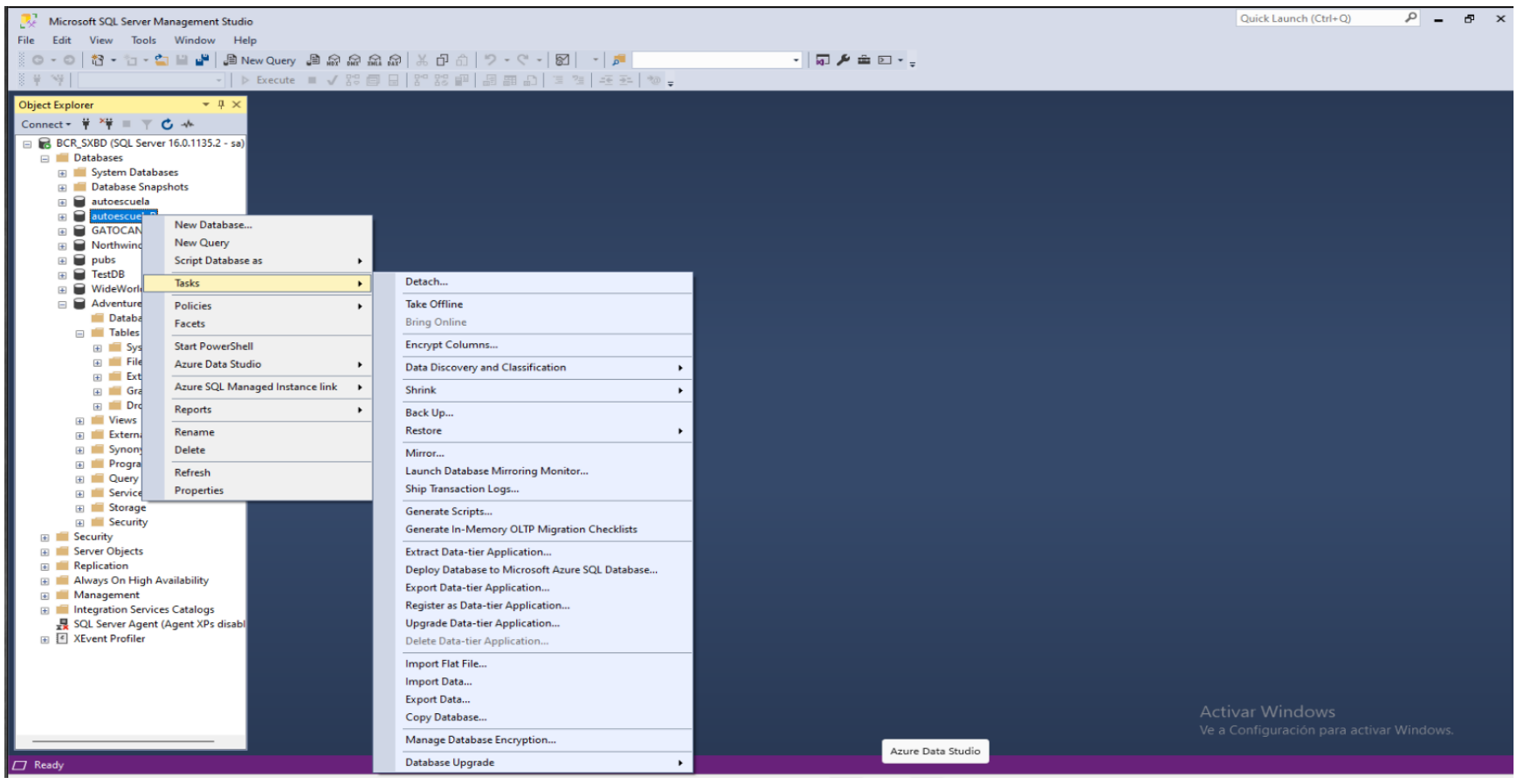
```
    precio    DECIMAL(8, 2)
);

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_profes

ALTER TABLE profe_practicas ADD CONSTRAINT profe_practicas_pk UNIQUE ( profesor_profesor_

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_

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,
```

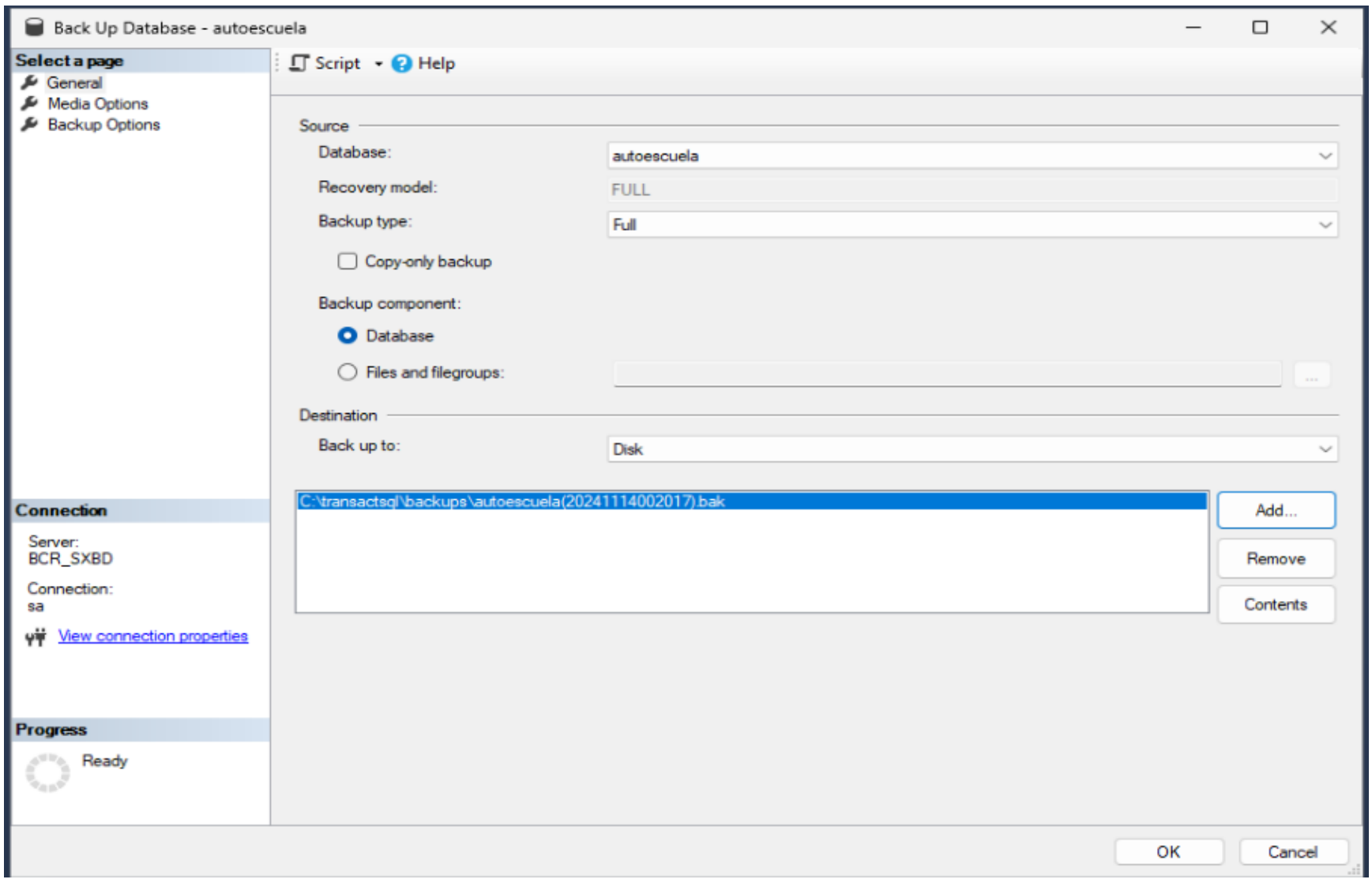
```
    marca               VARCHAR(10),
    modelo              VARCHAR(12),
    f_compra            DATE,
);

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
        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_prof
        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.



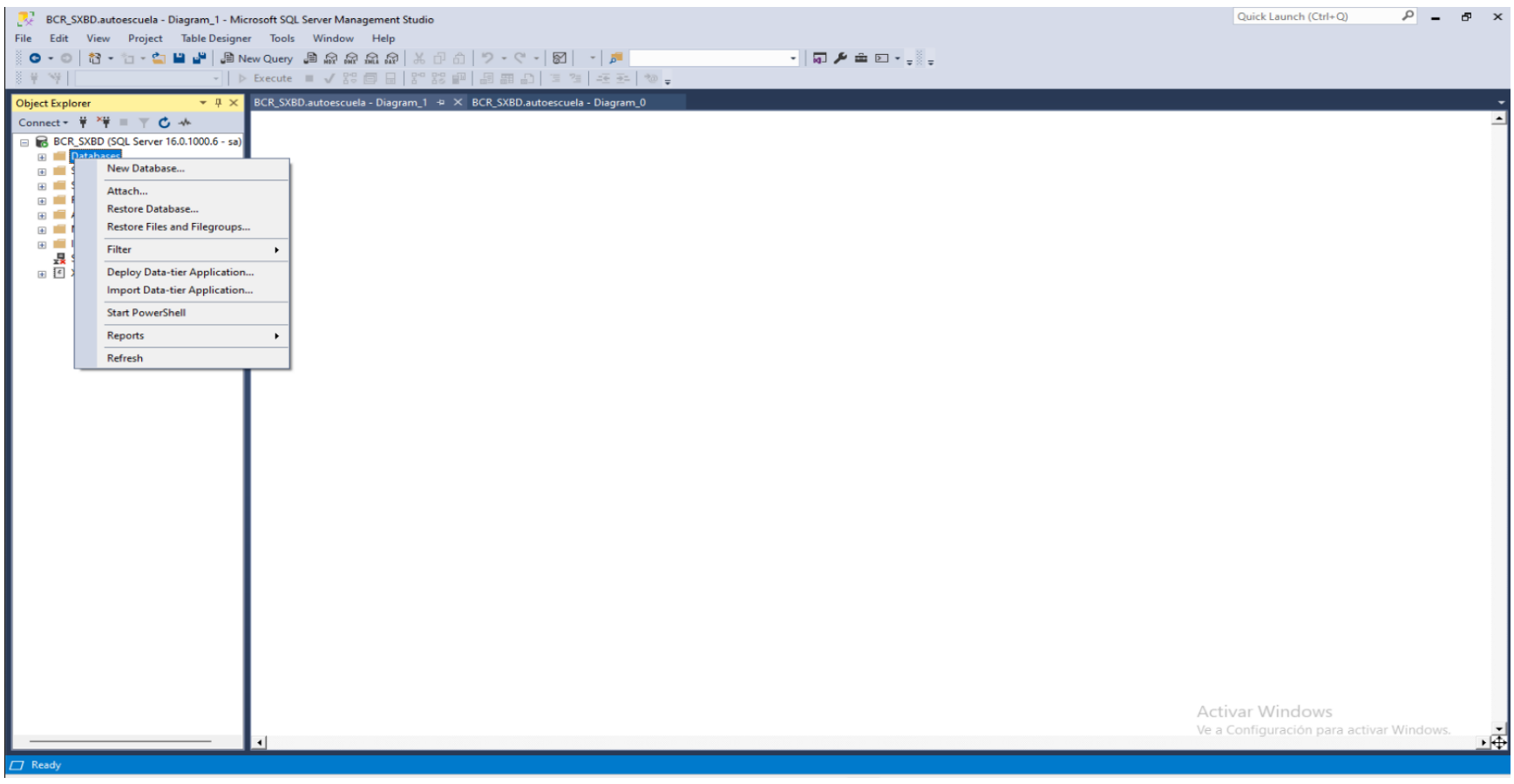


Instalación base de datos de ejemplo

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

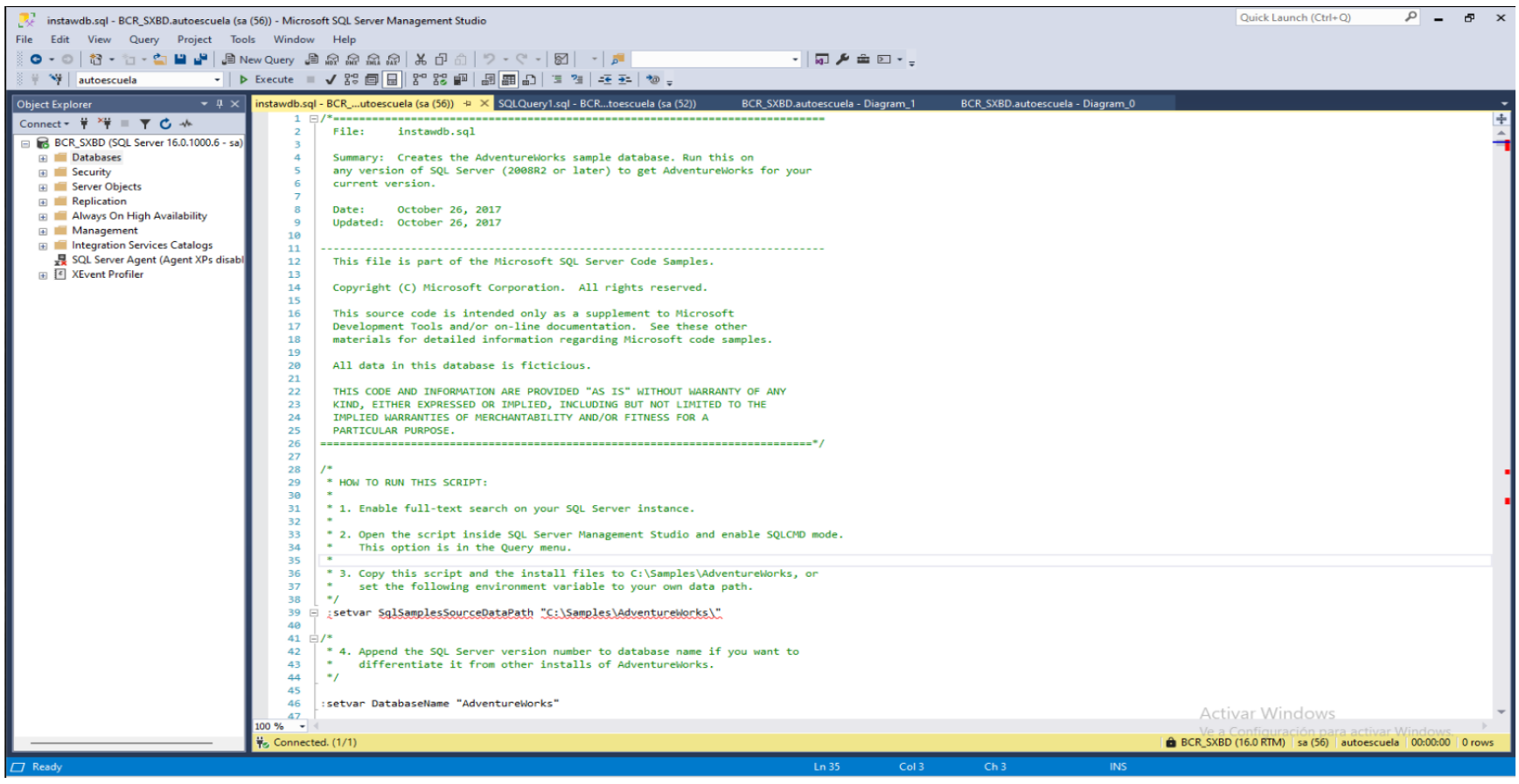
Restore Backup desde SSMS

Comenzamos por realizar un backup directamente desde SSMS, una vez hayamos descargado el archivo .bak, vamos a “Databases” en la barra de navegación 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.

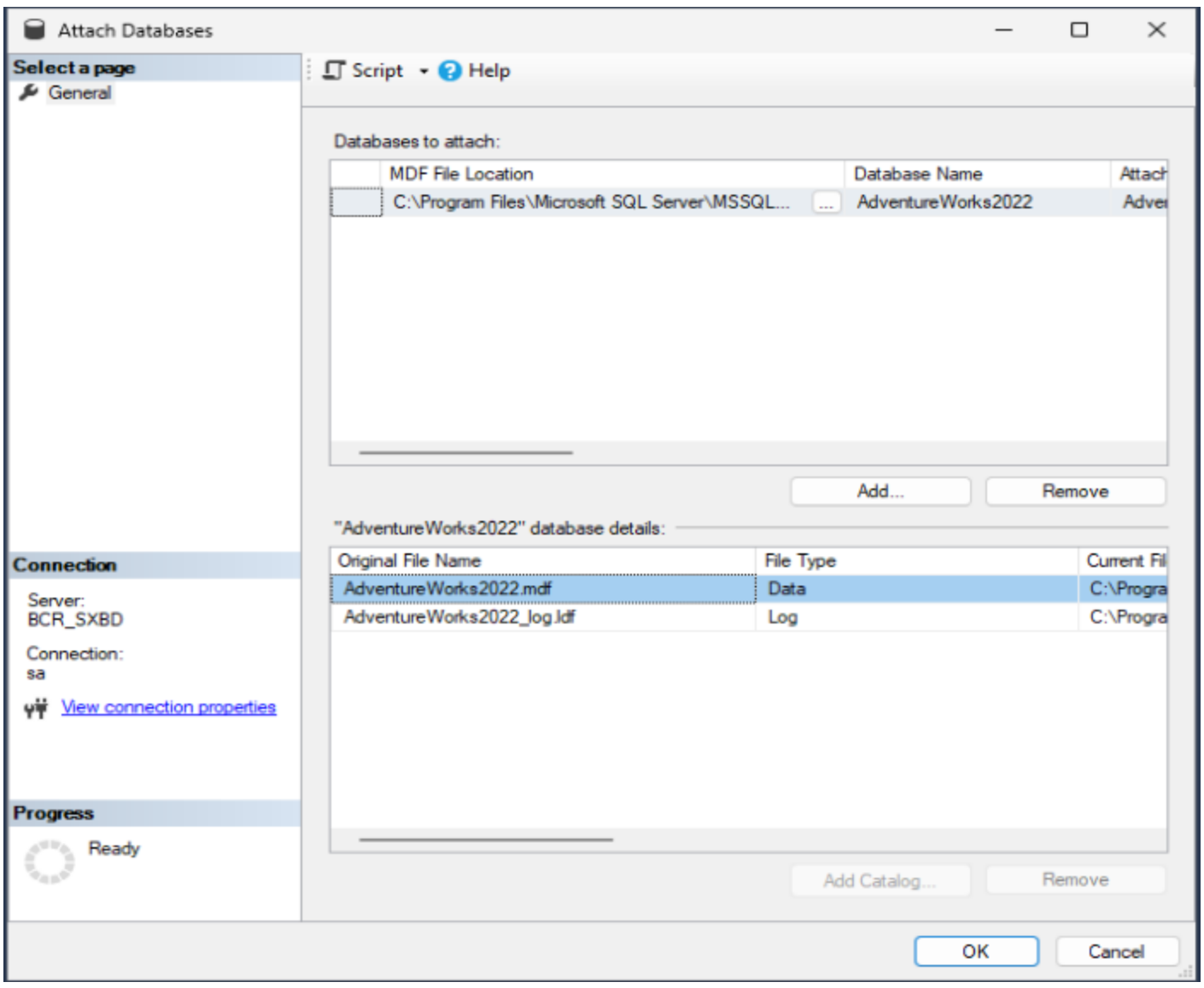
También estaría la opción de hacerlo usando el comando `invoke-Sqlcmd` lo que nos permite ejecutar instancias de SQL en Powershell directamente:

```

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 presionaresmos 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.


Attach Databases
Select a page
General
Script
Help
Databases to attach:
MDF File Location
Database Name
C:\Program Files\Microsoft SQL Server\MSSQL...
AdventureWorks2022
Add...
Remove
"AdventureWorks2022" database details:
Original File Name
File Type
Current Fil
AdventureWorks2022.mdf
Data
C:\Progra
AdventureWorks2022_log.ldf
Log
C:\Progra
Add Catalog...
Remove
Connection
Server:
BCR_SXBD
Connection:
sa
View connection properties
Progress
Ready
OK
Cancel