

Lenguajes de marcas y sistemas de gestión
de la información (LMSGI)

UD6: “*Almacenamiento de información*”:

Operaciones con XML y gestores de
bases de datos relacionales: SQL
Server

1. Índice

Introducción.....	3
Instalación de herramientas.....	4
Operaciones con XML y SQL Server.....	16
1 Consulta de tablas con resultado en formato XML en SQL Server.....	16
1.1 FOR XML AUTO.....	16
1.2 FOR XML RAW.....	17
1.3 FOR XML PATH.....	17
1.4 FOR XML y utilización de espacios de nombres.....	23
2 Lectura de documentos XML con OPENXML.....	24
2.1 Lectura con OPENXML centrada en elementos.....	26
2.1.1 Uso de un patrón ColPattern.....	27
2.1.2 Obtención de una tabla con una única columna de tipo XML.....	29
3 Inserción de datos en una tabla relacional a partir de la lectura de un documento XML.....	29
3.1 Almacenamiento de documentos XML directamente en una columna de tipo XML de una tabla relacional.....	31
3.1.1 Creación de un XML SCHEMA COLLECTION.....	34
RECURSOS.....	36

Introducción

Para ilustrar cómo los gestores de bases de datos relacionales permiten trabajar con documentos XML, vamos a trabajar con uno de ellos en particular: Microsoft SQL Server

En este documento, veremos las instrucciones básicas necesarias para la manipulación de documentos o consultas en formato XML.

En este módulo no se contemplan contenidos sobre teoría de bases de datos, para eso existen en el ciclo módulos específicos, por lo que no se incluye la iniciación al lenguaje de consultas SQL. De todas formas, se verán consultas muy sencillas similares a las sentencias FLWOR de XQuery.

Si algún alumno/a no ha cursado aún el módulo de Bases de Datos o tiene dudas sobre las consultas aquí utilizadas, puede consultar el [Tutorial de SQL de w3schools](#). Se verán principalmente consultas SELECT sencillas e INSERT INTO

Básicamente, las bases de datos relacionales permiten almacenar información en relaciones (habitualmente llamadas tablas por su representación en forma tabular). Cada relación (o tabla) tiene una serie de campos (o columnas) y registros (o filas).

No puede haber dos registros exactamente iguales, por lo que debe haber al menos un campo o conjunto de campos que distingan los registros (o filas). Hay más requisitos formales para que se consideren bases de datos relacionales, así como restricciones entre los datos, pero escapan al ámbito de este módulo.

Cada campo (o columna) tendrá un tipo de datos, dependiendo de si son numéricos, cadenas de caracteres u otros tipos de dato. Los tipos de datos disponibles en Microsoft SQL Server, se pueden consultar [aquí](#). Para nosotros, será de especial importancia el tipo de datos [XML](#), para documentos XML bien formados o fragmentos de documentos XML.

Los ejemplos con los que trabajaremos están en una carpeta de **Recursos** Ejemplos de XML y Microsoft SQL Server Cartafol

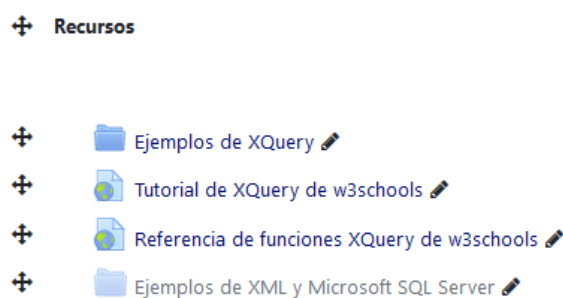


Figura 1

Instalación de herramientas

Vamos a trabajar con **SQL Server Express** y **SQL Server Management Studio**: Una herramienta gráfica que permite realizar consultas y hacer cambios en las bases de datos de forma más intuitiva.

Para ello, descargaremos [la edición gratuita Express](#):

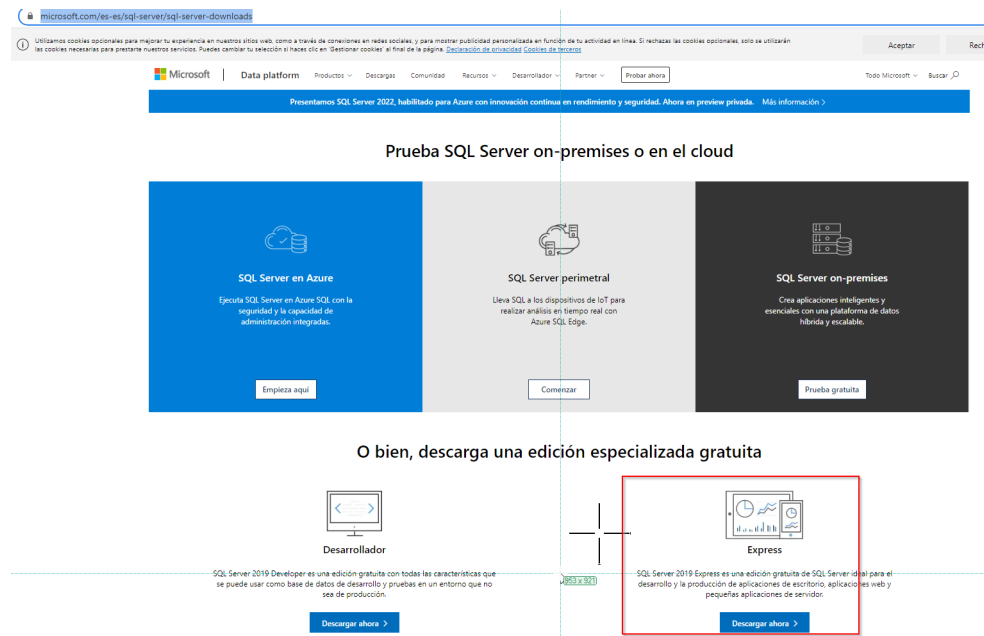


Figura 2

Realizaremos la instalación por defecto y al finalizar la instalación, descargaremos e instalaremos con las opciones por defecto **SQL Server Management Studio (SSMS)** :

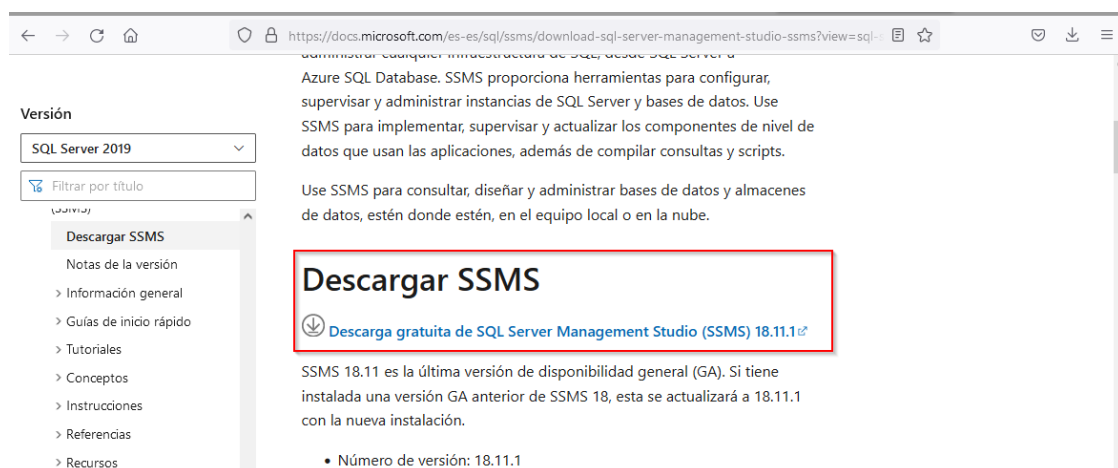


Figura 3

Una vez arrancado el SQL Server Management Studio, iniciamos sesión con Windows Authentication.

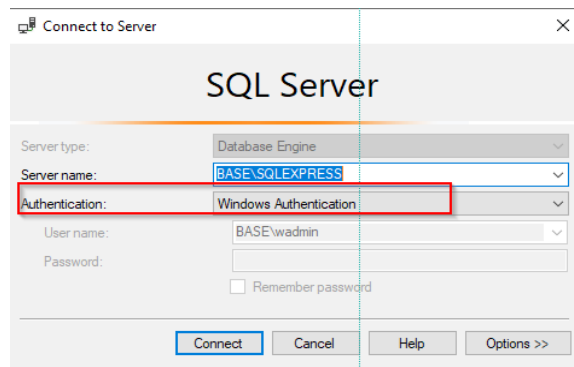


Figura 4

Vamos a utilizar una **base de datos de ejemplo** proporcionada por [Microsoft: AdventureWorksLT2016. Bak](#). Descargamos ese fichero [AdventureWorksLT2016. Bak](#).

Para crear la base de datos, sobre el explorador de objetos, desplegamos hasta ver la carpeta Databases. Sobre ella, botón derecho, **Restore database....**:

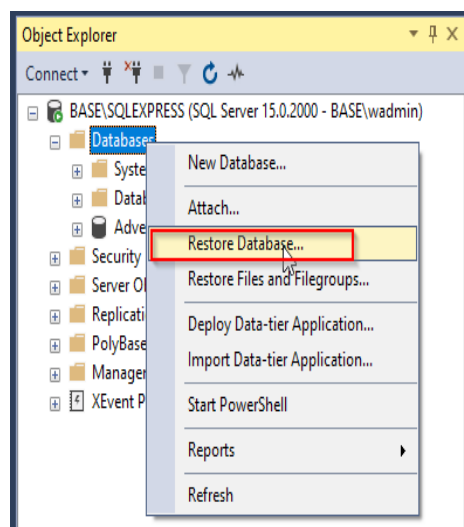


Figura 5

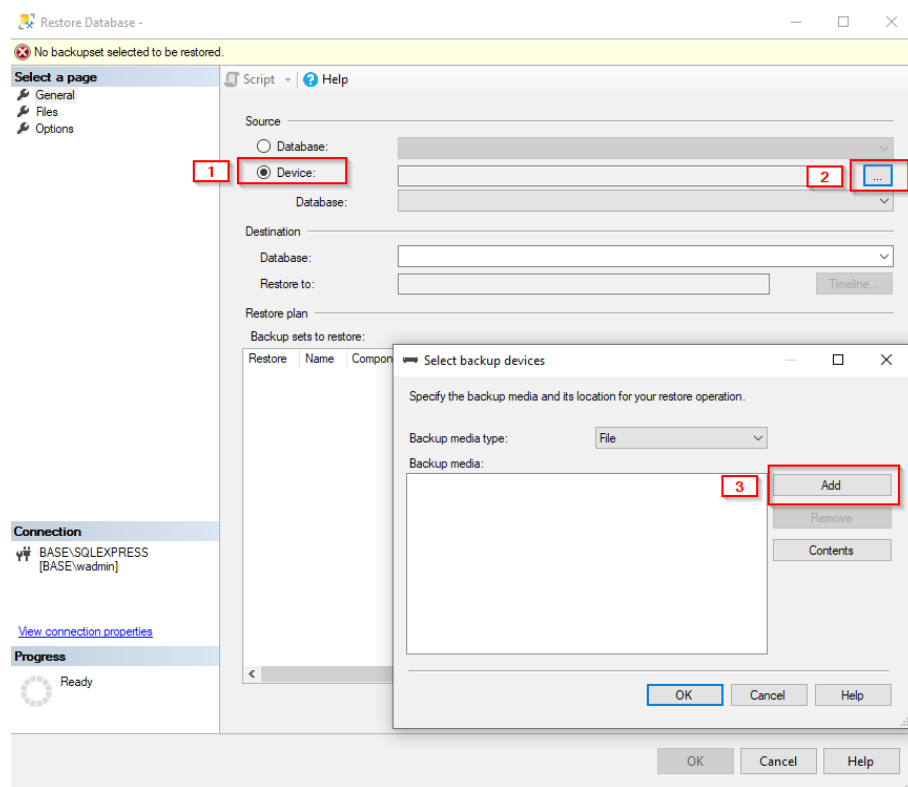


Figura 6:

1. Seleccionamos el radio button de Device para tomar del sistema de ficheros la copia de seguridad
2. Hacemos clic en el botón ...a su derecha
3. En la nueva ventana que se abrirá, hacemos clic en el botón Add

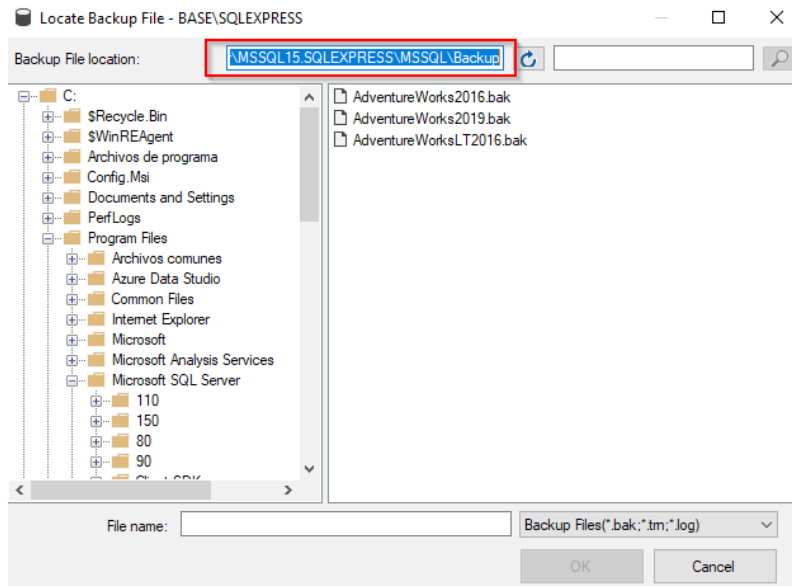


Figura 7: Copiamos la ubicación donde se sitúan los archivos de backup. Será algo similar a la siguiente: C:\Program Files\Microsoft SQL Server\MSSQL15.SQLEXPRESS\MSSQL\Backup.

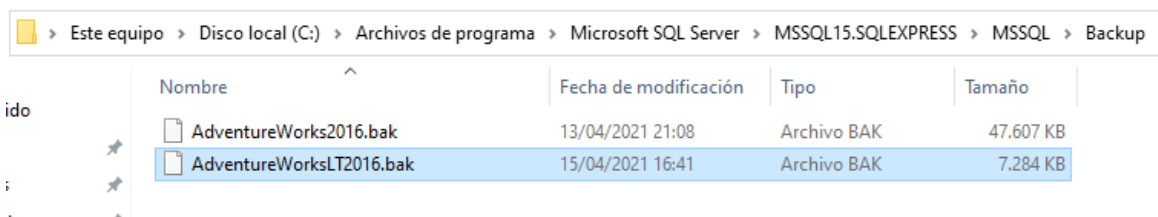


Figura 8: Nos situamos en el explorador de archivos en esa ubicación y pegamos en esa ubicación el archivo AdventureWorksLT2016. Bak que descargamos previamente

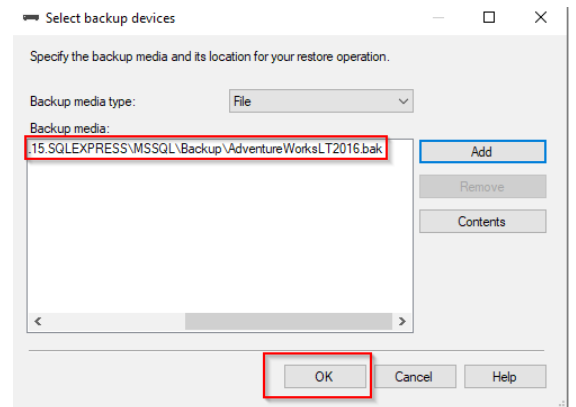
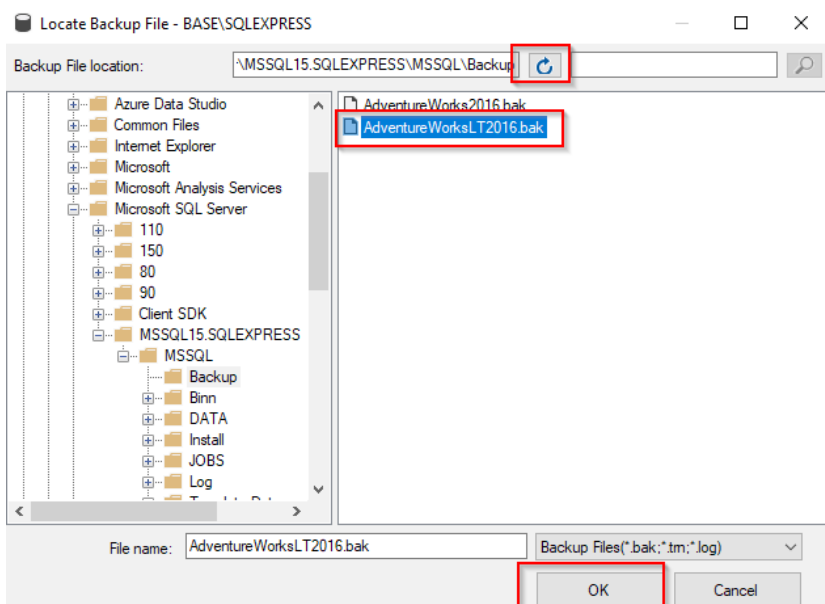


Figura 10: Hacemos clic en OK

Figura 9: Volvemos a la ventana de selección del fichero de backup y hacemos clic en el botón de Refrescar. Deberíamos ver el fichero AdventureWorksLT2016. Bak. Lo seleccionamos y hacemos clic sobre el botón OK

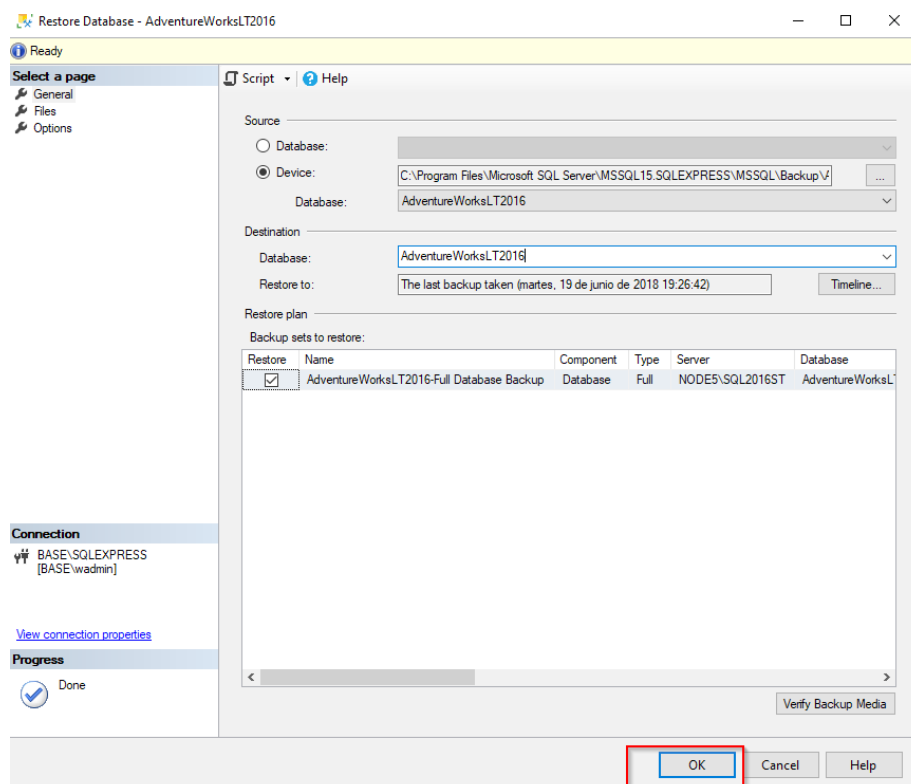


Figura 11: Finalmente hacemos clic en OK y debería restaurarse la BD

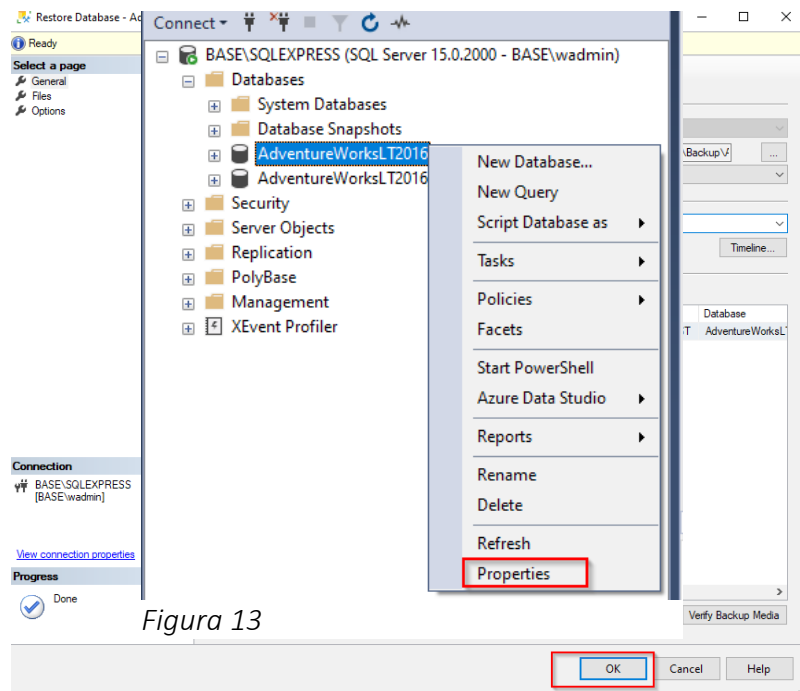


Figura 13

Figura 12: A continuación, se verá la base de datos bajo la carpeta de Databases. Sobre la base de datos AdventureWorksLT2016, botón derecho > Properties

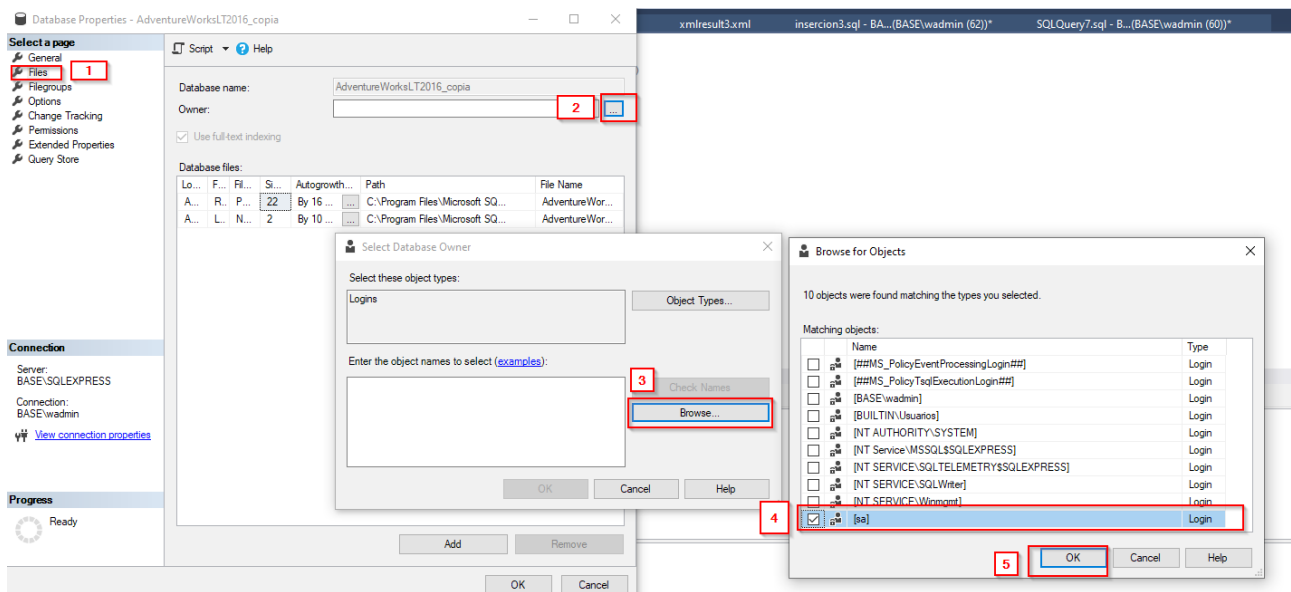


Figura 14:

1. Seleccionamos la página de Files
2. En Owner, hacemos clic sobre el botón ...
3. En la ventana emergente, botón Browse
4. En la ventana emergente, seleccionamos el sa (system administrator) como propietario de la BD
5. Hacemos clic en OK, recursivamente en todas las ventanas abiertas, hasta que se cierre la última.

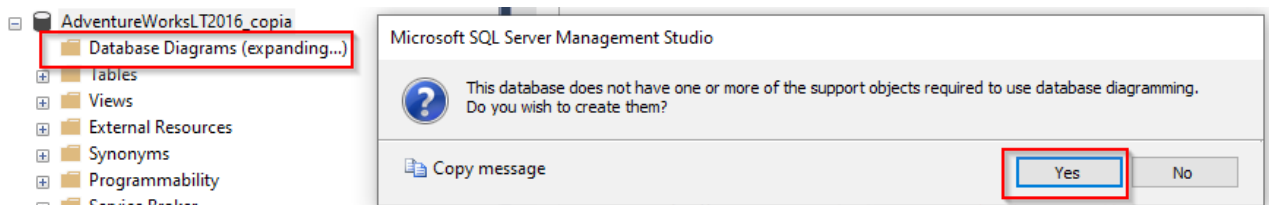


Figura 15: Desplegamos la carpeta de Database Diagrams. Cuando se obtenga el mensaje de la imagen, contestamos Yes para crear n nuevo diagrama.

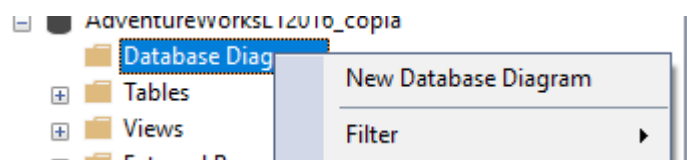


Figura 16: Sobre la carpeta de Database Diagrams, botón derecho > New Database Diagram

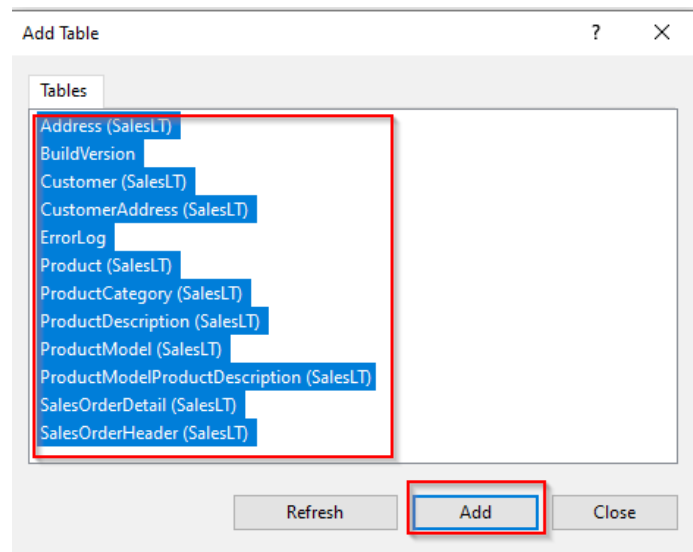


Figura 17: Seleccionamos todas las tablas para que aparezcan todas ellas en el diagrama y hacemos clic sobre Add

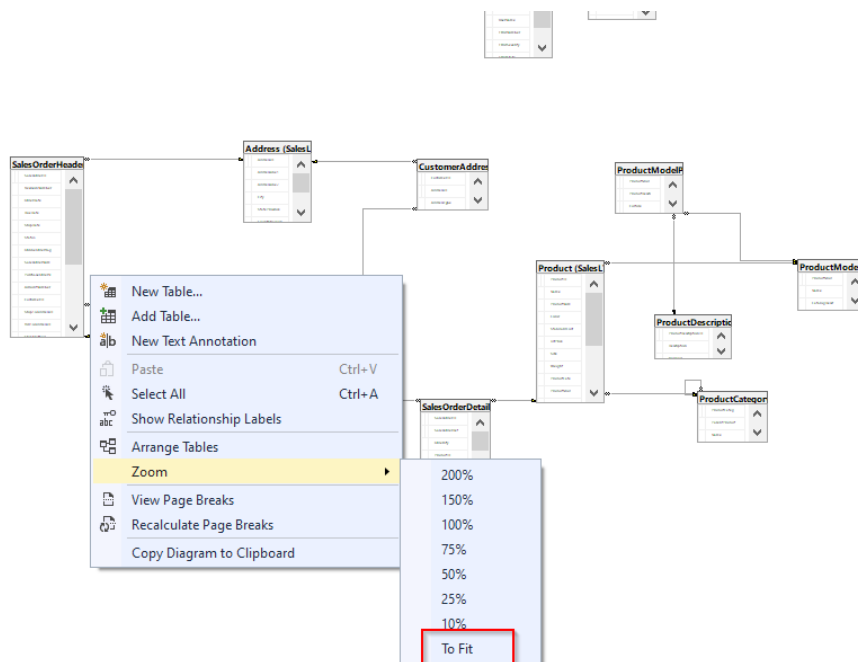


Figura 18: Sobre el diagrama resultante, botón derecho> Zoom> To Fit

A continuación, se pueden reordenar arrastrando las tablas para tener una idea de cómo están relacionados los datos.

El objetivo no es comprender completamente el modelo de datos o las relaciones entre tablas de la base de datos AdventureWorksLT2016 para la realización de esta sección, simplemente nos servirá como contexto para la realización de consultas.

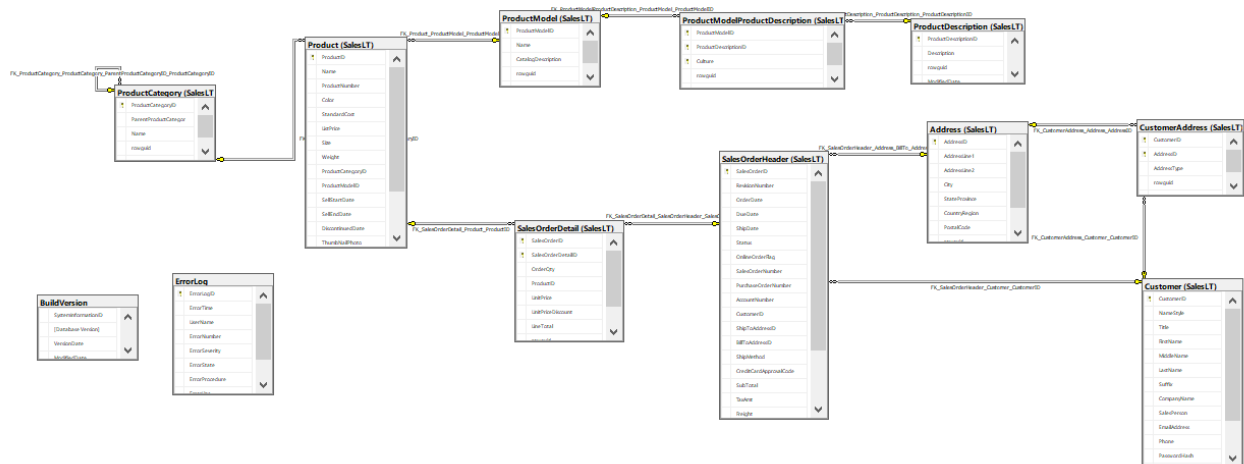


Figura 19: Diagrama de la base de datos de ejemplo AdventureWorksLT2016. bak disponible [aquí](#).

Una vez restaurada la base de datos, desplegamos la carpeta **Tables** en el explorador de objetos. En este caso las el nombre de las tablas va precedido del nombre del **esquema SalesLT**. Un esquema es una colección de objetos, en este caso una colección de tablas.

SalesLT.Address es una tabla llamada Address que pertenece al esquema SalesLT

Cada tabla tiene una carpeta **Columns** con el nombre de las columnas que pertenecen a la tabla y su tipo de datos.

AddressID es un entero, además Primary Key (PK) que no puede tener valores nulos.

Es importante fijarse e inspirarse en el tipo de datos de las columnas ya existentes para hacer los ejercicios que propondremos a continuación.

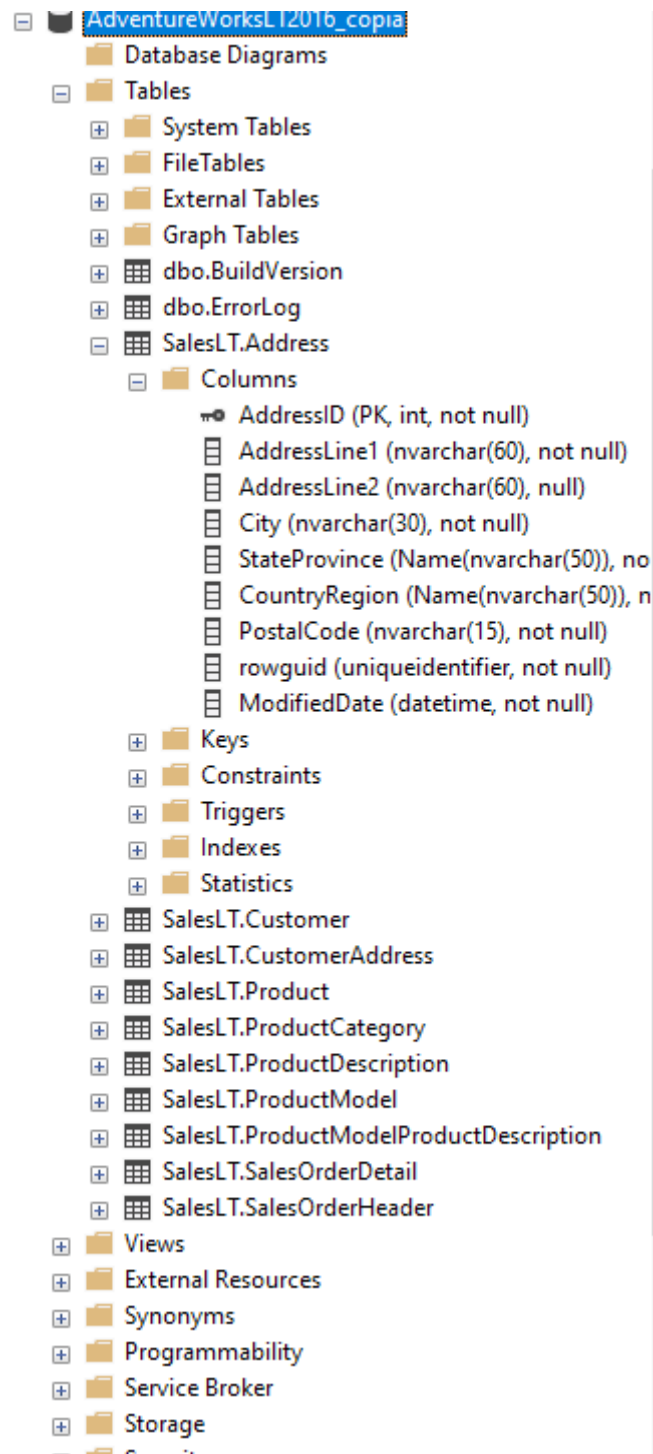


Figura 20

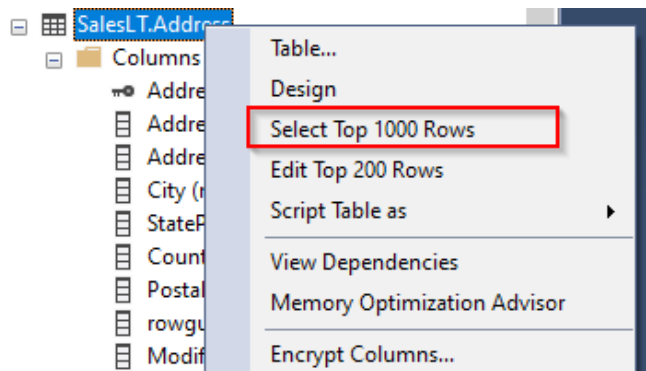


Figura 21: Para ver los datos almacenados en las tablas, sobre una tabla en concreto, botón derecho > Select Top 1000 Rows.

Query window content:

```

/***** Script for SelectTopRows command from SSIS *****/
SELECT TOP (1000) [AddressID]
,[AddressLine1]
,[AddressLine2]
,[City]
,[StateProvince]
,[CountryRegion]
,[PostalCode]
,[rowguid]
,[ModifiedDate]
FROM [AdventureWorksLT2016_copia].[SalesLT].[Address]
  
```

Results grid (first 17 rows shown):

	AddressID	AddressLine1	AddressLine2	City	StateProvince	CountryRegion	PostalCode	rowguid	ModifiedDate
1	9	8713 Yosemite Ct.	NULL	Bothell	Washington	United States	98011	268AF621-76D7-4C78-9441-144FD139821A	2006-07-01 00:00:00.000
2	11	1318 Lasalle Street	NULL	Bothell	Washington	United States	98011	981B3303-ACA2-49C7-9A96-FB6707858269	2007-04-01 00:00:00.000
3	25	9178 Jumping St.	NULL	Dallas	Texas	United States	75201	C8DF38D9-48FD-4654-A8DD-14A67A84D3C6	2006-09-01 00:00:00.000
4	28	9228 Via Del Sol	NULL	Phoenix	Arizona	United States	85004	12AE5EE1-FC3E-4E7E-BBD5-3B970B169774	2005-09-01 00:00:00.000
5	32	26910 Indela Road	NULL	Montreal	Quebec	Canada	H1Y 2H5	84A95F62-3AE8-4E7E-BBD5-5A6F00CD982D	2006-08-01 00:00:00.000
6	185	2681 Eagle Peak	NULL	Bellevue	Washington	United States	98004	78CCF442-2268-46CC-8472-14C44C14E98C	2006-09-01 00:00:00.000
7	297	7943 Walnut Ave	NULL	Renton	Washington	United States	98055	52410DA4-2778-4B1D-A599-95746625CE6D	2006-08-01 00:00:00.000
8	445	6388 Lake City Way	NULL	Burnaby	British Columbia	Canada	V5A 3A6	53572F25-9133-4A8B-A065-102FF35416EE	2006-09-01 00:00:00.000
9	446	52560 Free Street	NULL	Toronto	Ontario	Canada	M4B 1V7	801A1DFC-5125-486B-AA84-CC8D2EC57CA4	2005-08-01 00:00:00.000
10	447	22580 Free Street	NULL	Toronto	Ontario	Canada	M4B 1V7	88CEE379-D8B8-433B-B84E-A35E09435500	2006-08-01 00:00:00.000
11	448	2575 Bloor Street East	NULL	Toronto	Ontario	Canada	M4B 1V6	2DF6D0AD-0926-4F34-A450-9B1083150CBF	2007-08-01 00:00:00.000
12	449	Station E	NULL	Chalk River	Ontario	Canada	K0J 1J0	8B5A7729-CB75-4303-A607-7F979384D94F	2005-08-01 00:00:00.000
13	450	575 Rue St Amable	NULL	Quebec	Quebec	Canada	G1R	5F3C345A-6475-41D5-B178-D88D277338B1	2006-09-01 00:00:00.000
14	451	2512 4th Ave Sw	NULL	Calgary	Alberta	Canada	T2P 2G8	49644F1E-6F90-46D9-8DBB-9DB15F0EF7EC	2006-12-01 00:00:00.000
15	452	55 Lakeshore Blvd East	NULL	Toronto	Ontario	Canada	M4B 1V6	A358652F-0E00-49E6-B6B0-DE1099C43506	2005-09-01 00:00:00.000
16	453	6333 Cote Vertu	NULL	Montreal	Quebec	Canada	H1Y 2H7	355681F2-4D9D-4522-BFA0-1058E64394D8	2007-09-01 00:00:00.000
17	454	2566 Court Street West	NULL	Toronto	Ontario	Canada	M4B 1V6	5E4DC670-988B-4D7A-BE0A-ACEE5E6B6E71	2007-08-01 00:00:00.000

Query executed successfully. 450 rows

Figura 22: Se creará una consulta nueva en SQL para seleccionar los primeros 1000 registros de la tabla y se ejecutará directamente. En la parte inferior se aprecian las columnas y los registros almacenados en la tabla SalesLT.Address.

En la parte inferior derecha, se ve el número de registros que forman parte del resultado de la consulta.

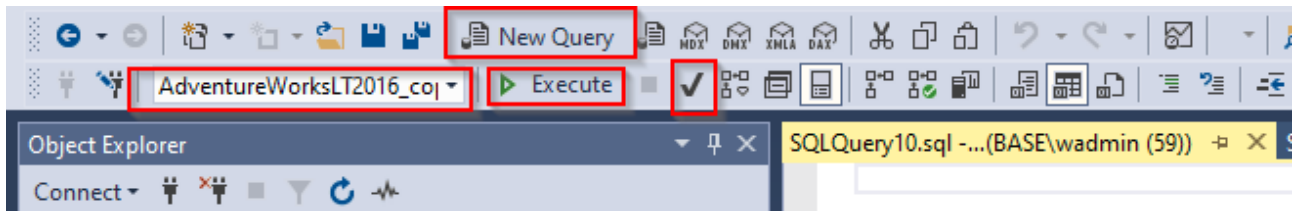


Figura 23: Algunos botones de las barras de herramientas muy útiles son:

1. *New Query*: Abre una nueva pestaña en el editor para introducir una consulta nueva
2. *Lista desplegable de BD*: Permite seleccionar sobre qué BD se va a ejecutar la consulta del editor. Es importante que esté establecida a AdventureWorksLT2016 para estos ejercicios
3. *El check en forma de V*: Comprueba la sintaxis de la consulta del editor.
4. *Execute*: Ejecutará la consulta del editor contra la BD seleccionada en el paso 2. Si no hay texto seleccionado, se ejecutará todo lo que esté escrito en el editor. Si hay texto seleccionado, solo se ejecuta la parte seleccionada.

Operaciones con XML y SQL Server

Veremos a continuación 3 operaciones con documentos XML en SQL Server:

- Consulta de tablas con resultado en formato XML
- Lectura de documentos XML
- Inserción de datos en una tabla relacional a partir de un documento XML

1 Consulta de tablas con resultado en formato XML en SQL Server

Por ejemplo, para ver los registros de la tabla Customer podemos usar la siguiente consulta SQL:

```
SELECT *  
FROM [AdventureWorksLT2016].[SalesLT].[Customer]
```

Results		Messages											
	CustomerID	NameStyle	Title	FirstName	MiddleName	LastName	Suffix	CompanyName	SalesPerson	EmailAddress	Phone	PasswordHash	PasswordSalt
1	1	0	Mr.	Orlando	N	Gee	NULL	A Bike Store	adventure-works\pamela0	orlando0@adventure-works.com	245-555-0173	L/RwXp4w7RWmEgXX-/A7cXaePEPcp+KwQH2JL7w=	1kXYs4=
2	2	0	Mr.	Keith	NULL	Harris	NULL	Progressive Sports	adventure-works\david8	keith0@adventure-works.com	170-555-0127	YPdRdvqeA9y6wyxEsFdhBDNXX6Ckx+CRgbvJtknw=	fs1Z3hY=
3	3	0	Ms.	Donna	F.	Cameras	NULL	Advanced Bike Components	adventure-works\jillian0	donna0@adventure-works.com	279-555-0130	LNoK27abGQe48gGue3EBV/UhYSToV0/s87dCRV7Ljk=	YTNH5Rw=
4	4	0	Ms.	Janet	M.	Gates	NULL	Modular Cycle Systems	adventure-works\jillian0	janet1@adventure-works.com	710-555-0173	BlzTpSNbUW1Lt+L5cWFR7MF6nB2a8WpmGaQpJLOJA=	nm7D5e4=
5	5	0	Mr.	Lucy	NULL	Hamington	NULL	Metropolitan Sports Supply	adventure-works\shu0	lucy0@adventure-works.com	828-555-0186	KlqY15weX3PG8TS5Gddp6LFFVdd3CoRtZMAp0+R4=	cnFKU4w=
6	6	0	Ms.	Rosmarie	J.	Carroll	NULL	Aerobic Exercise Company	adventure-works\linda3	rosmarie0@adventure-works.com	244-555-0112	OKT0eiczCdlzymHH0tyJKGIC/fCILSooSZ8dQ2Y34VM=	ihWF50M=

Figura 24

Es posible recuperar resultados de una consulta SQL en formato XML especificando la cláusula **FOR XML** en la consulta.

En una cláusula FOR XML especificaremos uno de estos 3 modos¹:

- AUTO
- RAW
- PATH

1.1 FOR XML AUTO

```
SELECT *  
FROM [AdventureWorksLT2016].[SalesLT].[Customer]  
FOR XML AUTO
```

¹ Hay un cuarto modo: EXPLICIT que permite un mayor control sobre el documento XML resultado, pero su sintaxis es más compleja y no lo veremos en este módulo. Es posible consultar más información al respecto [aquí](#)


```

XML_F52E2B61-18A...00805F49916B1.xml  SQLQuery1.sql - B...(BASE\wadmin (56))*
<AdventureWorksLT2016.SalesLT.Customer CustomerID="1" NameStyle="0" Title="Mr." FirstName="Orlando" MiddleName="N." LastName="Gee" CompanyName="A Bike Store" SalesPerson="adventure-works\pamela0" EmailAddress="pamela0@adventure-works.com" Phone="5482200020" Address="7097 Chestnut Street, Suite 200, Seattle, WA 98104" />
<AdventureWorksLT2016.SalesLT.Customer CustomerID="2" NameStyle="0" Title="Mr." FirstName="Keith" LastName="Harris" CompanyName="Progressive Sports" SalesPerson="adventure-works\david8" EmailAddress="david8@adventure-works.com" Phone="5482200020" Address="7097 Chestnut Street, Suite 200, Seattle, WA 98104" />
<AdventureWorksLT2016.SalesLT.Customer CustomerID="3" NameStyle="0" Title="Ms." FirstName="Donna" MiddleName="F." LastName="Carreras" CompanyName="Advanced Bike Components" SalesPerson="adventure-works\jill" EmailAddress="jill@adventure-works.com" Phone="5482200020" Address="7097 Chestnut Street, Suite 200, Seattle, WA 98104" />
<AdventureWorksLT2016.SalesLT.Customer CustomerID="4" NameStyle="0" Title="Ms." FirstName="Janet" MiddleName="M." LastName="Gates" CompanyName="Modular Cycle Systems" SalesPerson="adventure-works\jill" EmailAddress="jill@adventure-works.com" Phone="5482200020" Address="7097 Chestnut Street, Suite 200, Seattle, WA 98104" />
<AdventureWorksLT2016.SalesLT.Customer CustomerID="5" NameStyle="0" Title="Mr." FirstName="Lucy" LastName="Harrington" CompanyName="Metropolitan Sports" SalesPerson="adventure-works\jill" EmailAddress="jill@adventure-works.com" Phone="5482200020" Address="7097 Chestnut Street, Suite 200, Seattle, WA 98104" />

```

Figura 25: El resultado incluye las columnas como atributos de un elemento con el nombre de la base_de datos.esquema.nombre_tabla.

1.2 FOR XML RAW

SELECT *

FROM [AdventureWorksLT2016].[SalesLT].[Customer]

FOR XML RAW

```

XML_F52E2B61-18A...00805F49916B2.xml  XML_F52E2B61-18A...00805F49916B1.xml  SQLQuery1.sql - B...(BASE\wadmin (56))*
<row CustomerID="1" NameStyle="0" Title="Mr." FirstName="Orlando" MiddleName="N." LastName="Gee" CompanyName="A Bike S
<row CustomerID="2" NameStyle="0" Title="Mr." FirstName="Keith" LastName="Harris" CompanyName="Progressive Sports" Sal
<row CustomerID="3" NameStyle="0" Title="Ms." FirstName="Donna" MiddleName="F." LastName="Carreras" CompanyName="Advar
<row CustomerID="4" NameStyle="0" Title="Ms." FirstName="Janet" MiddleName="M." LastName="Gates" CompanyName="Modular
<row CustomerID="5" NameStyle="0" Title="Mr." FirstName="Lucy" LastName="Harrington" CompanyName="Metropolitan Sports
<row CustomerID="6" NameStyle="0" Title="Ms." FirstName="Rosmarie" MiddleName="J." LastName="Carroll" CompanyName="Aer
<row CustomerID="7" NameStyle="0" Title="Mr." FirstName="Dominic" MiddleName="P." LastName="Gash" CompanyName="Associ
<row CustomerID="10" NameStyle="0" Title="Ms." FirstName="Kathleen" MiddleName="M." LastName="Garza" CompanyName="Rura
<row CustomerID="11" NameStyle="0" Title="Ms." FirstName="Katherine" LastName="Harding" CompanyName="Sharp Bikes" Sale

```

Figura 26: El modo RAW genera un único elemento <row> por cada fila del conjunto de filas que devuelve la instrucción SELECT y las columnas como atributos

1.3 FOR XML PATH

SELECT *

FROM [AdventureWorksLT2016].[SalesLT].[Customer]

FOR XML PATH

```

<row>
  <CustomerID>1</CustomerID>
  <NameStyle>0</NameStyle>
  <Title>Mr.</Title>
  <FirstName>Orlando</FirstName>
  <MiddleName>N.</MiddleName>
  <LastName>Gee</LastName>
  <CompanyName>A Bike Store</CompanyName>
  <SalesPerson>adventure-works\pamela0</SalesPerson>
  <EmailAddress>orlando0@adventure-works.com</EmailAddress>
  <Phone>245-555-0173</Phone>
  <PasswordHash>L/Rlwzpz4w7RwmEgXX+/A7cXaePEPcp+KwQh12fJL7w=</PasswordHash>
  <PasswordSalt>1KjXYs4=</PasswordSalt>
  <rowguid>3F5AE95E-B87D-4AED-95B4-C3797AFCB74F</rowguid>
  <ModifiedDate>2005-08-01T00:00:00</ModifiedDate>
</row>
<row>
  <CustomerID>2</CustomerID>
  <NameStyle>0</NameStyle>
  <Title>Mr.</Title>
  <FirstName>Keith</FirstName>
  <LastName>Harris</LastName>
  <CompanyName>Progressive Sports</CompanyName>
  <SalesPerson>adventure-works\david8</SalesPerson>
  <EmailAddress>keith0@adventure-works.com</EmailAddress>
  <Phone>170-555-0127</Phone>
  <PasswordHash>YPdtRdvqeAhj6wyxEsFdshBDNXxkCXn+CRgbvJItknw=</PasswordHash>
  <PasswordSalt>fs1ZGhY=</PasswordSalt>
  <rowguid>E552F657-A9AF-4A7D-A645-C429D6E02491</rowguid>
  <ModifiedDate>2006-08-01T00:00:00</ModifiedDate>
</row>
<row>
  <CustomerID>3</CustomerID>
  <NameStyle>0</NameStyle>

```

Figura 27: El resultado incluye elementos en lugar de atributos. Puesto que la cláusula *SELECT* no especifica ningún alias para los nombres de columna, los nombres de elemento secundario generados son los mismos que los nombres de columna correspondientes de la cláusula *SELECT*. Para cada fila del resultado se agrega una etiqueta *<row>*.

Es posible sobrescribir el valor *<row>* por otro de la siguiente forma:

```

SELECT *
FROM [AdventureWorksLT2016].[SalesLT].[Customer]
FOR XML PATH ('Customer')

```

```

<Customer>
  <CustomerID>1</CustomerID>
  <NameStyle>0</NameStyle>
  <Title>Mr.</Title>
  <FirstName>Orlando</FirstName>
  <MiddleName>N.</MiddleName>
  <LastName>Gee</LastName>
  <CompanyName>A Bike Store</CompanyName>
  <SalesPerson>adventure-works\pamela0</SalesPerson>
  <EmailAddress>orlando0@adventure-works.com</EmailAddress>
  <Phone>245-555-0173</Phone>
  <PasswordHash>L/Rlwzpz4w7RwmEgXX+/A7cXaePEPcp+KwQh12fJL7w=</PasswordHash>
  <PasswordSalt>1KjXYs4=</PasswordSalt>
  <rowguid>3F5AE95E-B87D-4AED-95B4-C3797AFCB74F</rowguid>
  <ModifiedDate>2005-08-01T00:00:00</ModifiedDate>
</Customer>
<Customer>
  <CustomerID>2</CustomerID>
  <NameStyle>0</NameStyle>
  <Title>Mr.</Title>
  <FirstName>Keith</FirstName>
  <LastName>Harris</LastName>
  <CompanyName>Progressive Sports</CompanyName>
  <SalesPerson>adventure-works\david8</SalesPerson>
  <EmailAddress>keith0@adventure-works.com</EmailAddress>
  <Phone>170-555-0127</Phone>
  <PasswordHash>YPdtRdvqeAhj6wyxEsFdshBDNXxkCXn+CRgbvJItknw=</PasswordHash>
  <PasswordSalt>fs1ZGhY=</PasswordSalt>
  <rowguid>E552F657-A9AF-4A7D-A645-C429D6E02491</rowguid>
  <ModifiedDate>2006-08-01T00:00:00</ModifiedDate>
</Customer>
<Customer>
  <CustomerID>3</CustomerID>
  <NameStyle>0</NameStyle>

```

Figura 28

Es posible especificar con XPath los nombres de los elementos del XML resultante, por ejemplo con un alias para atributos o para elementos. Por ejemplo:

```
SELECT CustomerID as '@id'  
FirstName AS nombre,  
LastName as apellido,  
Phone as telefono,  
CompanyName as empresa  
FROM [AdventureWorksLT2016].[SalesLT].[Customer]  
WHERE CompanyName LIKE '%Sports%'  
ORDER BY CompanyName  
FOR XML PATH ('Cliente')2
```



```
<Cliente id="403">  
  <nombre>Juanita</nombre>  
  <apellido>Holman</apellido>  
  <telefono>996-555-0196</telefono>  
  <empresa>Affordable Sports Equipment</empresa>  
</Cliente>  
<Cliente id="29853">  
  <nombre>Juanita</nombre>  
  <apellido>Holman</apellido>  
  <telefono>996-555-0196</telefono>  
  <empresa>Affordable Sports Equipment</empresa>  
</Cliente>  
<Cliente id="29850">  
  <nombre>Bob</nombre>  
  <apellido>Hodges</apellido>  
  <telefono>129-555-0120</telefono>  
  <empresa>All Seasons Sports Supply</empresa>  
</Cliente>  
<Cliente id="597">  
  <nombre>Bob</nombre>  
  <apellido>Hodges</apellido>  
  <telefono>129-555-0120</telefono>  
  <empresa>All Seasons Sports Supply</empresa>  
</Cliente>  
<Cliente id="492">
```

Figura 29

A tener en cuenta que los atributos deben aparecer antes de cualquier otro tipo de nodo, como los de elemento y texto, del mismo nivel. La consulta siguiente devolverá un error:

² En este caso se seleccionan algunas columnas de la tabla SalesLT.Customer de las compañías cuyo nombre contengan la palabra "Sports" y los resultados se ordenan por el nombre de la compañía.

```
SELECT
FirstName AS nombre,
CustomerID as '@id',
LastName as apellido,
Phone as telefono,
CompanyName as empresa
FROM [AdventureWorksLT2016].[SalesLT].[Customer]
WHERE CompanyName LIKE '%Sports%'
ORDER BY CompanyName
FOR XML PATH ('Cliente')
```

Podemos agregar un solo elemento de nivel superior especificando la opción **root**:

```
SELECT CustomerID as '@id',  
FirstName AS nombre,  
LastName as apellido,  
Phone as telefono,  
CompanyName as empresa  
FROM [AdventureWorksLT2016].[SalesLT].[Customer]  
WHERE CompanyName LIKE '%Sports%'  
ORDER BY CompanyName  
FOR XML PATH ('Cliente'), root('Clientes')
```



```
<Clientes>  
  <Cliente id="403">  
    <nombre>Juanita</nombre>  
    <apellido>Holman</apellido>  
    <telefono>996-555-0196</telefono>  
    <empresa>Affordable Sports Equipment</empresa>  
  </Cliente>  
  <Cliente id="29853">  
    <nombre>Juanita</nombre>  
    <apellido>Holman</apellido>  
    <telefono>996-555-0196</telefono>  
    <empresa>Affordable Sports Equipment</empresa>  
  </Cliente>  
  <Cliente id="29850">  
    <nombre>Bob</nombre>  
    <apellido>Hodges</apellido>  
    <telefono>129-555-0120</telefono>  
    <empresa>All Seasons Sports Supply</empresa>  
  </Cliente>  
  <Cliente id="597">  
    <nombre>Bob</nombre>  
    <apellido>Hodges</apellido>  
    <telefono>129-555-0120</telefono>  
    <empresa>All Seasons Sports Supply</empresa>  
  </Cliente>  
  <Cliente id="403">
```

Figura 30: De esta forma el documento cumple con el requisito de tener un único elemento raíz.

1.4 FOR XML y utilización de espacios de nombres

Para incluir espacios de nombres en el XML resultante, se define en primer lugar el espacio de nombres con su prefijo mediante **WITH XMLNAMESPACES** y a continuación se usa en el alias de la columna:

```
WITH XMLNAMESPACES (  
    'http://lmsgi05.iessanclemente.net' AS ns5,  
    'http://lmsgi06.iessanclemente.net' AS ns6)  
SELECT CustomerID as '@ns5:id',  
    FirstName AS "ns6:nombre",  
    LastName as apellido,  
    Phone as telefono,  
    CompanyName as empresa  
    FROM [AdventureWorksLT2016].[SalesLT].[Customer]  
    WHERE CompanyName LIKE '%Sports%'  
    ORDER BY CompanyName  
FOR XML PATH ('Cliente'), root ('Clientes')
```



```
<Clientes xmlns:ns6="http://lmsgi06.iessanclemente.net" xmlns:ns5="http://lmsgi05.iessanclemente.net">  
  <ns5:Cliente ns5:id="403">  
    <ns6:nombre>Juanita</ns6:nombre>  
    <apellido>Holman</apellido>  
    <telefono>996-555-0196</telefono>  
    <empresa>Affordable Sports Equipment</empresa>  
  </ns5:Cliente>  
  <ns5:Cliente ns5:id="29853">  
    <ns6:nombre>Juanita</ns6:nombre>  
    <apellido>Holman</apellido>  
    <telefono>996-555-0196</telefono>  
    <empresa>Affordable Sports Equipment</empresa>  
  </ns5:Cliente>  
  <ns5:Cliente ns5:id="29850">  
    <ns6:nombre>Bob</ns6:nombre>  
    <apellido>Hodges</apellido>  
    <telefono>129-555-0120</telefono>  
    <empresa>All Seasons Sports Supply</empresa>  
  </ns5:Cliente>  
  <ns5:Cliente ns5:id="597">  
    <ns6:nombre>Bob</ns6:nombre>  
    <apellido>Hodges</apellido>  
    <telefono>129-555-0120</telefono>  
    <empresa>All Seasons Sports Supply</empresa>  
  </ns5:Cliente>  
  <ns5:Cliente ns5:id="492">  
    <ns6:nombre>Garth</ns6:nombre>  
    <apellido>Fort</apellido>  
    <telefono>768-555-0125</telefono>  
  </ns5:Cliente>  
</Clientes>
```

Figura 31: Resultado de la inclusión de los espacios de nombres

2 Lectura de documentos XML con OPENXML

A partir de un documento XML podemos crear un conjunto de filas **en memoria** similar a una tabla con la representación interna del documento XML que, posteriormente podríamos almacenar en una tabla de la base de datos.

Para escribir consultas dirigidas a un documento XML utilizaremos la palabra clave [OPENXML](#), que proporciona un conjunto de filas en memoria que es similar a una tabla o una vista

Lo primero que deberemos hacer es llamar a [sp_xml_preparedocument](#) que analiza el documento XML y **devuelve un identificador numérico**. El documento analizado es una representación en árbol del modelo de objetos de documento (DOM) de los distintos nodos del documento XML. Este identificador numérico de documento se pasa a **OPENXML**.

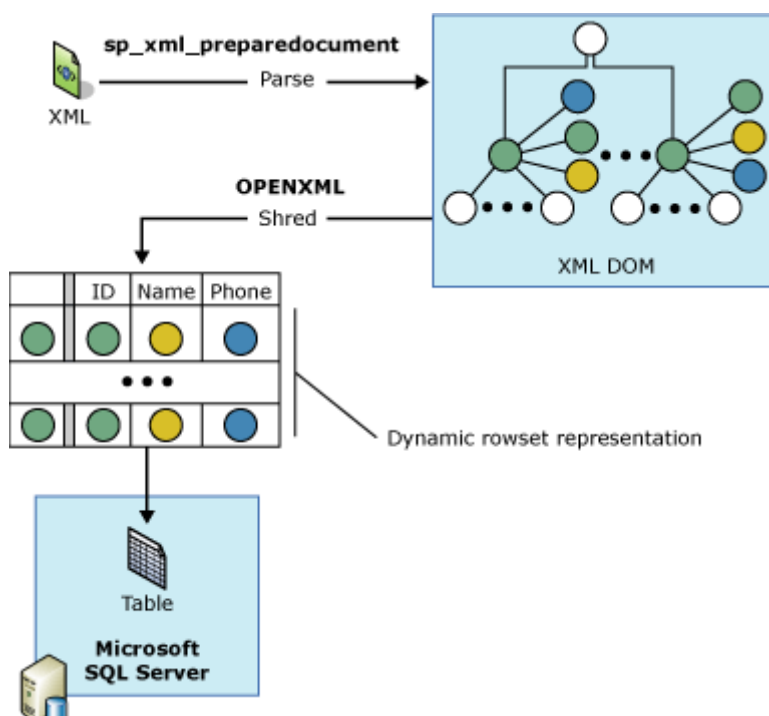


Figura 32: [Fuente](#)

Para leer los datos del archivo XML origen, usaremos **OPENROWSET** con la opción **BULK** y, con **SINGLE_BLOB**. Esto cargará en memoria, de forma masiva, el contenido del fichero en formato binario: varbinary(max)

Veamos un ejemplo completo. Usaremos un documento **addresses.xml** situado en la carpeta **Ejemplos de XML y Microsoft SQL Server Cartafol** de la sección de Recursos

<Addresses>


```

<Address id="9">
  <AddressLine1>Avenida de las Américas 154</AddressLine1>
  <CountryRegion>
    España
    <City>Madrid</City>
    <StateProvince>Madrid</StateProvince>
  </CountryRegion>
  <PostalCode>28080</PostalCode>
</Address>
<Address id="11">
  <AddressLine1>Pi i Margall, S/N</AddressLine1>
  <CountryRegion>
    España
    <City>Barcelona</City>
    <StateProvince>Cataluña</StateProvince>
  </CountryRegion>
  <PostalCode>08003</PostalCode>
</Address>
<Address id="5">
  <AddressLine1>9178 Jumping St.</AddressLine1>
  <CountryRegion>
    España
    <City>Santiago de Compostela</City>
    <StateProvince>Galicia</StateProvince>
  </CountryRegion>
  <PostalCode>15273</PostalCode>
</Address>
</Addresses>

```

Para tener acceso a los datos del fichero XML y obtener resultados en forma de tabla relacional en SQL Server usaremos el siguiente script con TRANSACT-SQL
lectura1.sql (habrá que actualizar la ruta al fichero, dependiendo de dónde lo hayáis descargado):

```
DECLARE @addresses xml
```



```

SELECT @addresses = aliasColumna
FROM OPENROWSET (BULK 'C:\Users\wadmin\Desktop\Ejemplos_UD06\
addresses.xml', SINGLE_BLOB)
AS aliasTabla (aliasColumna)
SELECT @addresses
DECLARE @hdoc int
EXEC sp_xml_preparedocument @hdoc OUTPUT, @addresses

```

```

SELECT *
FROM OPENXML (@hdoc, '/Addresses/Address' , 1)
WITH(
    id INT
)

```

```

EXEC sp_xml_removedocument @hdoc

```

	(No column name)
1	<Addresses><Address id="9"><AddressLine1>Avenida...

	id
1	9
2	11
3	5

Figura 33: Resultado de ejecución de lectura1.sql

2.1 Lectura con OPENXML centrada en elementos

Para indicar que la asignación que debe utilizarse entre los datos XML y el conjunto de filas relacional va a estar centrada en usar los elementos, el tercer argumento de **OPENXML** es un 2, en lugar de 1. Tenéis el ejemplo en **lectura2.sql** (habrá que actualizar la ruta al fichero, dependiendo de dónde lo hayáis descargado):

```

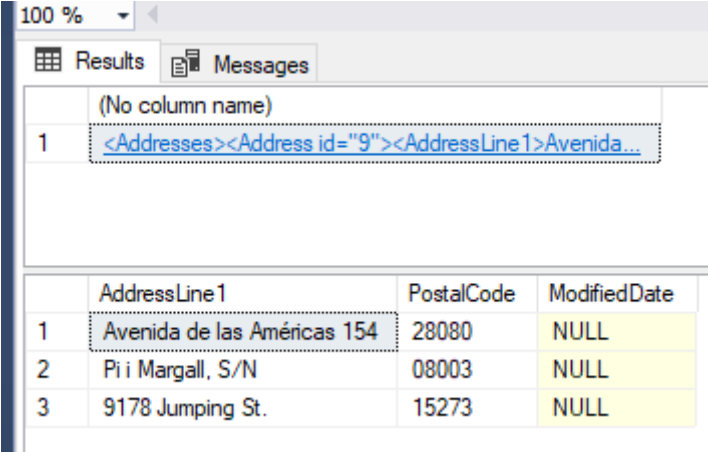
DECLARE @addresses xml
SELECT @addresses = aliasColumna
FROM OPENROWSET (BULK 'C:\Users\wadmin\Desktop\Ejemplos_UD06\
addresses.xml', SINGLE_BLOB)
AS aliasTabla (aliasColumna)

SELECT @addresses

DECLARE @hdoc int
EXEC sp_xml_preparedocument @hdoc OUTPUT, @addresses

SELECT *
FROM OPENXML (@hdoc, '/Addresses/Address' , 2)
WITH(
    AddressLine1 nvarchar(60) ,
    PostalCode nvarchar(15) ,
    ModifiedDate datetime
)
EXEC sp_xml_removedocument @hdoc

```



	AddressLine1	PostalCode	ModifiedDate
1	Avenida de las Américas 154	28080	NULL
2	Pi i Margall, S/N	08003	NULL
3	9178 Jumping St.	15273	NULL

Figura 34: Resultado de ejecutar lectura2.sql

2.1.1 Uso de un patrón ColPattern

Es posible indicar en la sentencia WITH un patrón **relativo a la ruta Xpath especificada en el segundo parámetro de OPENXML**. Este patrón, también llamado **ColPattern**, irá entre comillas simples y será

similar a la sintaxis conocida de XPath válida. De esta forma, también podremos especificar un nombre de columna resultante diferente al del elemento del documento XML. Por ejemplo, el fichero lectura3.sql(habría que actualizar la ruta al fichero, dependiendo de dónde lo hayáis descargado):

```
DECLARE @addresses xml
```

```
SELECT @addresses = aliasColumna
```

```
FROM OPENROWSET (BULK 'C:\Users\wadmin\Desktop\Ejemplos_UD06\addresses.xml',  
SINGLE_BLOB)
```

```
AS aliasTabla (aliasColumna)
```

```
SELECT @addresses
```

```
DECLARE @hdoc int
```

```
EXEC sp_xml_preparedocument @hdoc OUTPUT, @addresses
```

```
SELECT *  
FROM OPENXML (@hdoc, '/Addresses/Address' , 2)  
WITH(  
    ID int '@id',  
    AddressLine1 nvarchar(60) ,  
    CountryRegion nvarchar(50) './CountryRegion/text()',  
    Ciudad nvarchar(30) './CountryRegion/City'  
)
```

```
EXEC sp_xml_removedocument @hdoc
```

	ID	AddressLine1	CountryRegion	Ciudad
1	9	Avenida de las Américas 154	España	Madrid
2	11	Pi i Margall, S/N	España	Barcelona
3	5	9178 Jumping St.	España	Santiago de Compostela

Figura 35: Resultado de ejecutar lectura3.sql. Especial atención a los patrones encerrados entre comillas para navegar dentro del contexto '/Addresses/Address' señalado en OPENXML. Se utiliza la función text() para obtener el texto únicamente del elemento CountryRegion. Además, se usa un nombre diferente al del elemento City: Ciudad

2.1.2 Obtención de una tabla con una única columna de tipo XML

Siguiendo el formato que hemos seguido hasta el momento, es posible leer un documento XML al completo y crear una tabla con una sola columna de tipo de datos XML. Por ejemplo, en el script lectura4.sql (habrá que actualizar la ruta al fichero, dependiendo de dónde lo hayáis descargado):

```
DECLARE @documentoXML xml

SELECT @documentoXML = aliasColumna

FROM OPENROWSET (BULK 'C:\Users\wadmin\Desktop\Ejemplos_UD06\addresses.xml', SINGLE_BLOB)

AS aliasTabla (aliasColumna)

SELECT @documentoXML

DECLARE @hdoc int

EXEC sp_xml_preparedocument @hdoc OUTPUT, @documentoXML
```

```
SELECT *
FROM OPENXML (@hdoc, '/2')
WITH(
    Doc XML '.')
```

```
EXEC sp_xml_removedocument @hdoc
```

El resultado será la representación de una tabla con una única columna llamada Doc:

	Doc
1	<Addresses><Address id="9"><AddressLine1>Avenida de las Américas 154</AddressLine1><CountryRegion> España <City>Madrid</City><StateProvince>Madrid</StateProvince></CountryRegion><P...

Figura 36: Resultado de ejecutar lectura4.sql. Es posible ver el documento XML haciendo clic sobre el texto en azul

3 Inserción de datos en una tabla relacional a partir de la lectura de un documento XML

Hasta el momento, no hemos visto más que la representación de los datos almacenados en memoria de forma similar a una tabla relacional, pero no se ha materializado la inserción del contenido en ninguna tabla de la base de datos.

Vamos a ver un ejemplo de inserción en la tabla **SalesLT.Address** a partir de los datos recuperados del fichero **addresses.xml**

Para ello, utilizaremos el script `insercion1.sql` (habrá que actualizar la ruta al fichero, dependiendo de dónde lo hayáis descargado):

```
DECLARE @addresses xml
SELECT @addresses = aliasColumna
FROM OPENROWSET (BULK 'C:\Users\wadmin\Desktop\Ejemplos_UD06\addresses.xml', SINGLE_BLOB)
AS aliasTabla (aliasColumna)
SELECT @addresses
DECLARE @hdoc int
EXEC sp_xml_preparedocument @hdoc OUTPUT, @addresses
```

```
INSERT INTO AdventureWorksLT2016.SalesLT.Address (AddressLine1, City, StateProvince, CountryRegion, PostalCode)
SELECT AddressLine1, City, StateProvince, CountryRegion, PostalCode
FROM OPENXML (@hdoc, '/Addresses/Address' , 2)
WITH(
    AddressLine1 nvarchar(60) ,
    City nvarchar(30) './CountryRegion/City',
    StateProvince nvarchar(50) './CountryRegion/StateProvince',
    CountryRegion nvarchar(50) './CountryRegion/text()',
    PostalCode nvarchar(15)
)
```

```
EXEC sp_xml_removedocument @hdoc
```

Para comprobar si la inserción ha tenido lugar, ejecutamos la consulta:

```
SELECT *
FROM [AdventureWorksLT2016].[SalesLT].[Address]
WHERE CountryRegion LIKE 'España'
```

Y debería mostrar los registros del documento XML:

Results		Messages							
	AddressID	AddressLine1	AddressLine2	City	StateProvince	CountryRegion	PostalCode	rowguid	ModifiedDate
1	11389	Avenida de las Américas 154	NULL	Madrid	Madrid	España	28080	050F1FF2-3EA2-43F1-995C-8D36BB81084D	2021-04-16 12:57:55.177
2	11390	Pi i Margall, S/N	NULL	Barcelona	Cataluña	España	08003	0DF39550-08E4-428D-BE0B-229127668E44	2021-04-16 12:57:55.177
3	11391	9178 Jumping St.	NULL	Santiago de Compostela	Galicia	España	15273	3B974595-DFB1-4019-8218-F70414B77835	2021-04-16 12:57:55.177

Figura 37: Si se ejecuta más de una vez el script insercion.sql, es posible que se vean un total de registros múltiplo de 3 filas

3.1 Almacenamiento de documentos XML directamente en una columna de tipo XML de una tabla relacional

En la tabla SalesLT.ProductModel se puede observar un ejemplo de una columna, CatalogDescription, cuyo tipo es XML.

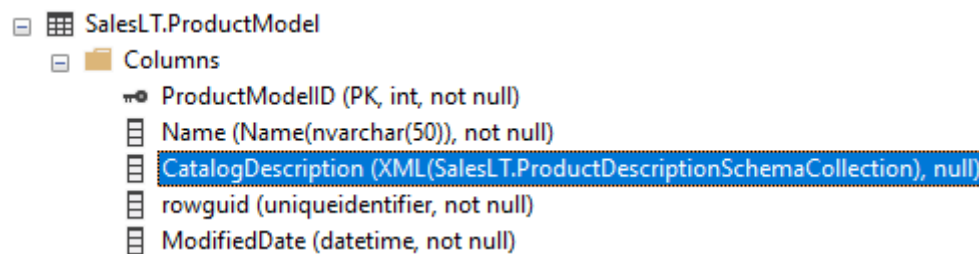


Figura 38: Se observa que la columna CatalogDescription es de tipo XML y además que debe validarse con un esquema que se guarda en una estructura llamada XML SCHEMA COLLECTION: SalesLT.ProductDescriptionSchemaCollection

Al observar el contenido de la tabla SalesLT.ProductModel con la consulta:

```
SELECT *
FROM [AdventureWorksLT2016].[SalesLT].[ProductModel]
WHERE CatalogDescription IS NOT NULL
```

Se observa que efectivamente la columna CatalogDescription contiene un documento XML:

	ProductModelID	Name	CatalogDescription
1	19	Mountain-100	<?xml-stylesheet href="ProductDescription.xsl" type="text/xsl"?><p 1:Produ
2	23	Mountain-500	<?xml-stylesheet href="ProductDescription.xsl" type="text/xsl"?><p 1:Produ
3	25	Road-150	<?xml-stylesheet href="ProductDescription.xsl" type="text/xsl"?><p 1:Produ
4	28	Road-450	<?xml-stylesheet href="ProductDescription.xsl" type="text/xsl"?><p 1:Produ
5	34	Touring-1000	<?xml-stylesheet href="ProductDescription.xsl" type="text/xsl"?><p 1:Produ
6	35	Touring-2000	<?xml-stylesheet href="ProductDescription.xsl" type="text/xsl"?><p 1:Produ

Figura 39: Si se hace clic sobre el texto en azul se podrá ver que se trata de un archivo de XSLT

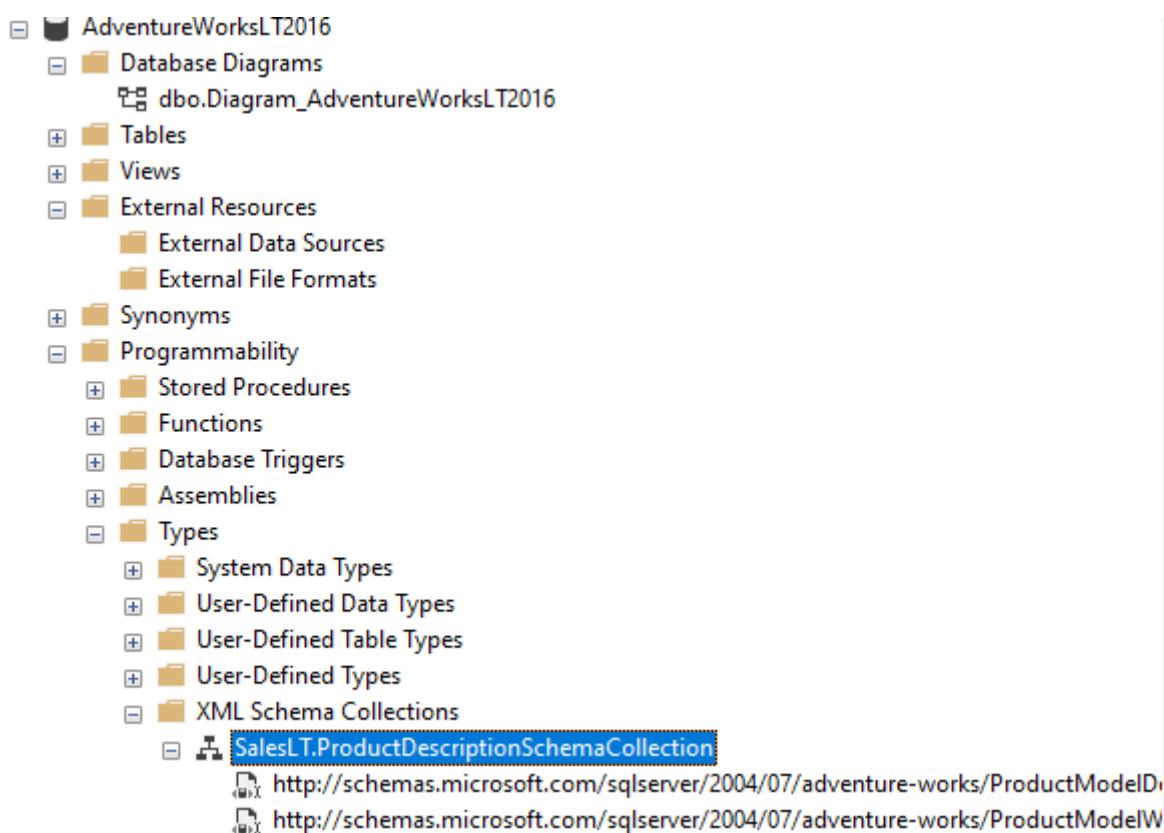


Figura 40: Es posible encontrar el esquema de validación en el explorador gráfico de la base de datos AdventureWorksLT2016 > Programmability > Types > XML Schema Collections

Para introducir datos en esa columna, no podremos introducir cualquier documento XML, sino que solo permitirá introducir el documento XML en la columna de destino si el documento XML realmente valida con el esquema señalado. Veamos un ejemplo con el script **insercion2.sql** (habrá que actualizar la ruta al fichero, dependiendo de dónde lo hayáis descargado) que lee el contenido de **addresses.xml** e intenta insertar el documento XML en la columna CatalogDescription :


```

DECLARE @documentoXML xml
SELECT @documentoXML = aliasColumna
FROM OPENROWSET (BULK 'C:\Users\wadmin\Desktop\Ejemplos_UD06\
addresses.xml', SINGLE_BLOB)
AS aliasTabla (aliasColumna)
SELECT @documentoXML
DECLARE @hdoc int
EXEC sp_xml_preparedocument @hdoc OUTPUT, @documentoXML

```

```

INSERT INTO SalesLT.ProductModel(Name, CatalogDescription)
SELECT 'Cualquier name', Doc
FROM OPENXML (@hdoc, '/' , 2)
WITH(
    Doc XML '.'
)

```

```
EXEC sp_xml_removedocument @hdoc
```

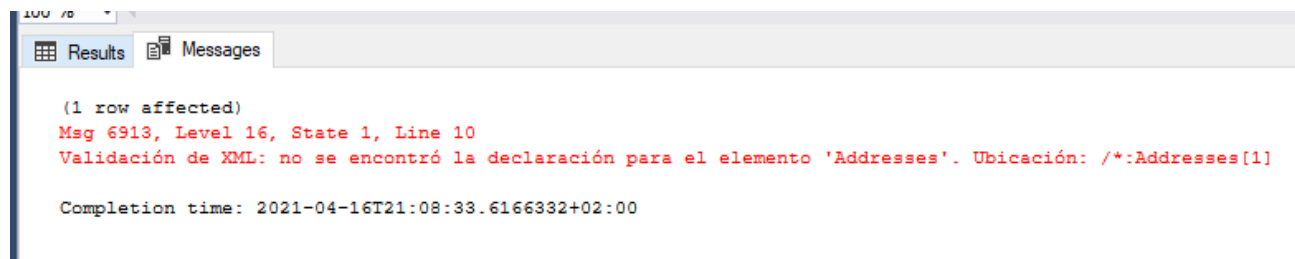


Figura 41: No permite su inserción porque el documento XML no valida con el esquema definido para esa columna

En cambio, si intentamos insertar un documento que sí cumple el esquema, como `CatalogDescription1.xml`, habremos logrado su inserción con `insercion3.sql` (habrá que actualizar la ruta al fichero, dependiendo de dónde lo hayáis descargado):

```

DECLARE @documentoXML xml
SELECT @documentoXML = aliasColumna
FROM OPENROWSET (BULK 'C:\Users\wadmin\Desktop\Ejemplos_UD06\
CatalogDescription1.xml', SINGLE_BLOB)
AS aliasTabla (aliasColumna)
SELECT @documentoXML

```

```

DECLARE @hdoc int
EXEC sp_xml_preparedocument @hdoc OUTPUT, @documentoXML

INSERT INTO SalesLT.ProductModel(Name, CatalogDescription)
SELECT 'Cualquier name', Doc
FROM OPENXML (@hdoc, '/' , 2)
WITH(
    Doc XML '.'
)

EXEC sp_xml_removedocument @hdoc

```

Podéis comprobar que se ha insertado con:

```

SELECT *
FROM [AdventureWorksLT2016].[SalesLT].[ProductModel]
where Name like 'Cualquier name%'

```

Results					
	ProductModelID	Name	CatalogDescription	rowguid	ModifiedDate
1	131	Cualquier name	<?xml-stylesheet href="ProductDescription.xsl" t...	16773150-028D-4195-84BF-EE2122E89C8F	2021-04-16 20:18:26.310

Figura 42: Si ejecutáis repetidas veces el script insercion3.sql, tendréis que cambiar el name porque se aplica una restricción de unicidad en esa columna. Por ejemplo Cualquier name2, 3, etc.

```

(1 row affected)
Msg 2627, Level 14, State 1, Line 10
Infraacción de la restricción UNIQUE KEY 'AK_ProductModel_Name'. No se puede insertar una clave duplicada en el objeto 'SalesLT.ProductModel'. El valor de la clave duplicada es (Cualquier name).
Se terminó la instrucción.
Completion time: 2021-04-23T11:29:19.3855734+02:00

```

Figura 43: Ejemplo de error que se puede dar si se ejecuta exactamente el mismo script insercion3.sql sin cambiar name.

3.1.1 Creación de un XML SCHEMA COLLECTION

Para crear un esquema de validación y almacenarlo en la base de datos, usaremos el siguiente script, disponible en este [enlace](#):

```

CREATE XML SCHEMA COLLECTION ManuInstructionsSchemaCollection AS
N'<?xml version="1.0" encoding="UTF-16">
<xsd:schema targetNamespace="https://schemas.microsoft.com/sqlserver/2004/07/adventure-works/ProductModelManuInstructions"
  xmlns      ="https://schemas.microsoft.com/sqlserver/2004/07/adventure-works/ProductModelManuInstructions"
  elementFormDefault="qualified"
  attributeFormDefault="unqualified"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema" >

  <xsd:complexType name="StepType" mixed="true" >
    <xsd:choice minOccurs="0" maxOccurs="unbounded" >
      <xsd:element name="tool" type="xsd:string" />
      <xsd:element name="material" type="xsd:string" />
      <xsd:element name="blueprint" type="xsd:string" />
      <xsd:element name="specs" type="xsd:string" />
      <xsd:element name="diag" type="xsd:string" />
    </xsd:choice>
  </xsd:complexType>

  <xsd:element name="root">
    <xsd:complexType mixed="true">
      <xsd:sequence>
        <xsd:element name="Location" minOccurs="1" maxOccurs="unbounded">
          <xsd:complexType mixed="true">
            <xsd:sequence>
              <xsd:element name="step" type="StepType" minOccurs="1"
maxOccurs="unbounded" />
            </xsd:sequence>
            <xsd:attribute name="LocationID" type="xsd:integer"
use="required"/>
            <xsd:attribute name="SetupHours" type="xsd:decimal"
use="optional"/>
            <xsd:attribute name="MachineHours" type="xsd:decimal"
use="optional"/>
            <xsd:attribute name="LaborHours" type="xsd:decimal"
use="optional"/>
            <xsd:attribute name="LotSize" type="xsd:decimal"
use="optional"/>
          </xsd:complexType>
        </xsd:element>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
</xsd:schema>' ;
GO

```

Vemos que se incluye rodeado por comillas simples el texto que forma parte del esquema y que la codificación utilizada es UTF-16

RECURSOS

- <https://docs.microsoft.com/es-es/sql/relational-databases/xml/for-xml-sql-server?view=sql-server-ver15>
- <https://docs.microsoft.com/es-es/sql/relational-databases/xml/openxml-sql-server?view=sql-server-ver15>
- <https://docs.microsoft.com/es-es/sql/relational-databases/import-export/examples-of-bulk-import-and-export-of-xml-documents-sql-server?view=sql-server-ver15>
- <https://docs.microsoft.com/es-es/sql/relational-databases/xml/columns-with-a-name?view=sql-server-ver15>