



XUNTA DE GALICIA  
CONSELLERÍA DE CULTURA,  
EDUCACIÓN E UNIVERSIDADE



IES de Teis  
Avda. de Galicia, 101  
36216 – Vigo  
886 12 04 64  
ies.teis@edu.xunta.es



Unión Europea-  
NextGenerationEU

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

Anexo I de la UD6 “*Almacenamiento de información*”:

## Manejo de XML con gestores de bases de datos relacionales: SQL Server



## 1. Índice

1	Introducción.....	3
2	Herramientas.....	3
3	Manejo de XML con SQL Server.....	5
3.1	Consulta de tablas con resultado en formato XML en SQL Server.....	5
3.1.1	FOR XML AUTO.....	5
3.1.2	FOR XML RAW.....	6
3.1.3	FOR XML PATH.....	6
3.1.4	FOR XML y utilización de espacios de nombres.....	12
3.2	Lectura de documentos XML con OPENXML.....	13
3.2.1	Lectura con OPENXML centrada en elementos.....	16
3.2.1.1	Uso de un patrón ColPattern.....	17
3.2.1.2	Obtención de una tabla con una única columna de tipo XML.....	18
3.3	Inserción de datos en una tabla relacional a partir de la lectura de un documento XML.....	19
3.3.1	Almacenamiento de documentos XML directamente en una columna de tipo XML de una tabla relacional.....	21
3.3.1.1	Creación de un XML SCHEMA COLLECTION.....	25
4	RECURSOS.....	27



## 1 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](#).

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.

## 2 Herramientas

Vamos a trabajar con [SQL Server Express](#).

Se proporciona una [máquina virtual con Windows 10](#) en la que se han instalado:

- Microsoft SQL Server 2019 (RTM)- 15.0.2000.5 (X64)
- SQL Server Management Studio 18: Una herramienta gráfica que permite realizar consultas y hacer cambios en las bases de datos de forma gráfica

Para acceder a la máquina virtual:

- Usuario: **wadmin**
- Contraseña: **abc123**.

Se ha utilizado una **base de datos de ejemplo** proporcionada por [Microsoft: AdventureWorksLT2016.Bak](#). 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.

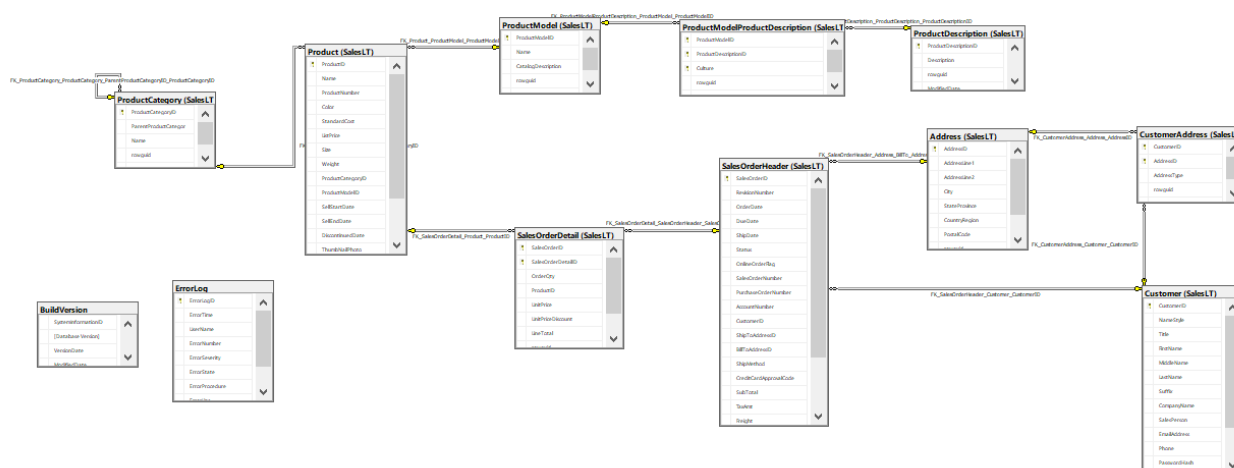


Figura 1: Diagrama de la base de datos de ejemplo AdventureWorksLT2016. bak disponible [aquí](#).

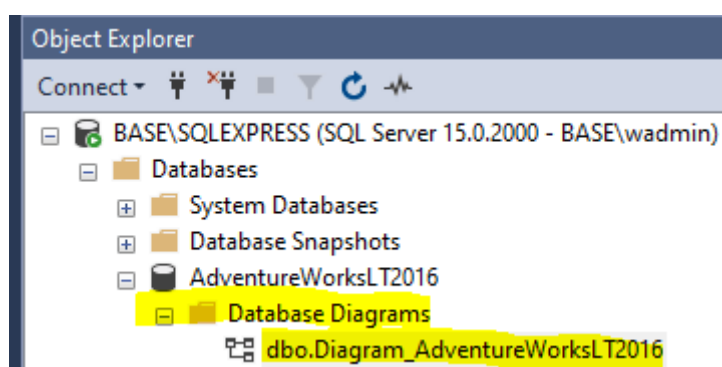


Figura 2: El diagrama también estará disponible en el explorador de SQL Server Management Studio donde se podrá tener una visión con más grado de detalle y mejor visión.



## 3 Manejo de XML con 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

### 3.1 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]
```

CustomerID	NameStyle	Title	FirstName	MiddleName	LastName	Suffix	CompanyName	SalesPerson	EmailAddress	Phone	PasswordHash	PasswordSalt
1	0	Mr.	Orlando	N.	Gee	NULL	A Bike Store	adventure-works/pamela0	orlando0@adventure-works.com	245-555-0173	L/Rwxyz4w7RWmEgXX+/A7cXaePEPcp+KwQH2JL7w=	1KqY9s4=
2	0	Mr.	Keith	NULL	Harris	NULL	Progressive Sports	adventure-works/david8	keith0@adventure-works.com	170-555-0127	YPdRdvqeAhy6wyxEsFdhBDNxxkCXn+CPgbvJlkmw=	fs1Z3hY=
3	0	Ms.	Donna	F.	Cameras	NULL	Advanced Bike Components	adventure-works/jillian0	donna0@adventure-works.com	279-555-0130	LNoK27abGQo48gGue3EBV/UhYSToV0/s87dCRV7uJk=	YTNH5Rw=
4	0	Ms.	Janet	M.	Gates	NULL	Modular Cycle Systems	adventure-works/jillian0	janet1@adventure-works.com	710-555-0173	BlzTp5NbUW1Uit+L5cWFR7MF6nB2a8WpmGaQpJLOJA=	nm7D5e4=
5	0	Mr.	Lucy	NULL	Harrington	NULL	Metropolitan Sports Supply	adventure-works/shu0	lucy0@adventure-works.com	828-555-0186	KUqY15wsX3PG8TS5G5ddp6LFFVdd3CoRhZMAP0+R4=	cnFKU4w=
6	0	Ms.	Rosmarie	J.	Carroll	NULL	Aerobic Exercise Company	adventure-works/linda3	rosmarie0@adventure-works.com	244-555-0112	OKT0eiczCdlzymHH0tyJKQICfCILSooSZ8dQ2Y34VM=	ihWF50M=

Figura 3

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 modos<sup>1</sup>:

- AUTO
- RAW
- PATH

#### 3.1.1 FOR XML AUTO

```
SELECT *
```

<sup>1</sup> 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](#)



FROM [AdventureWorksLT2016].[SalesLT].[Customer]  
FOR XML AUTO

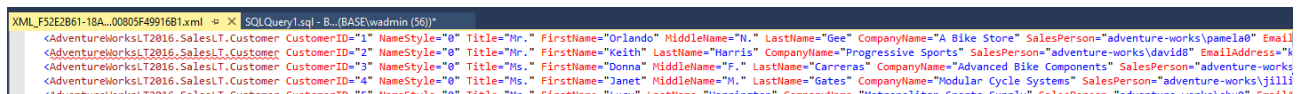


Figura 4: El resultado incluye las columnas como atributos de un elemento con el nombre de la base\_de datos.esquema.nombre\_tabla.

### 3.1.2 FOR XML RAW

SELECT \*  
FROM [AdventureWorksLT2016].[SalesLT].[Customer]  
FOR XML RAW

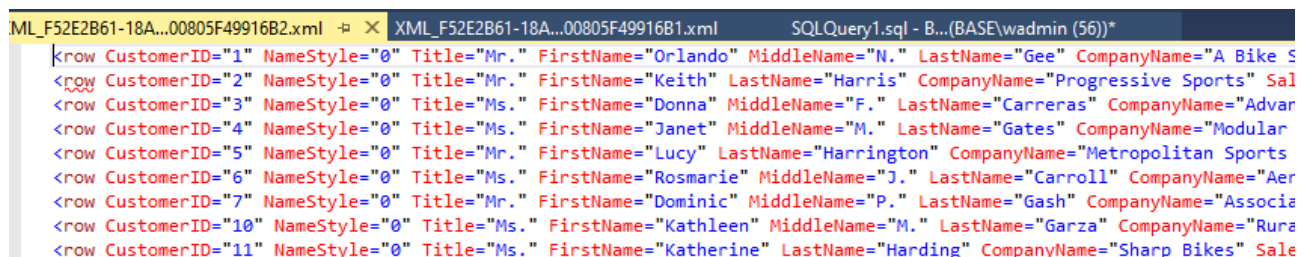


Figura 5: El modo RAW genera un único elemento <row> por cada fila del conjunto de filas que devuelve la instrucción SELECT y las columnas como atributos

### 3.1.3 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 6: 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 7





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 8

<sup>2</sup> 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.



XUNTA DE GALICIA  
CONSELLERÍA DE CULTURA,  
EDUCACIÓN E UNIVERSIDADE



IES de Teis  
Avda. de Galicia, 101  
36216 – Vigo  
886 12 04 64  
ies.teis@edu.xunta.es



Unión Europea-  
NextGenerationEU

**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:**

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



### 3.1.4 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 10: Resultado de la inclusión de los espacios de nombres



## 3.2 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**.

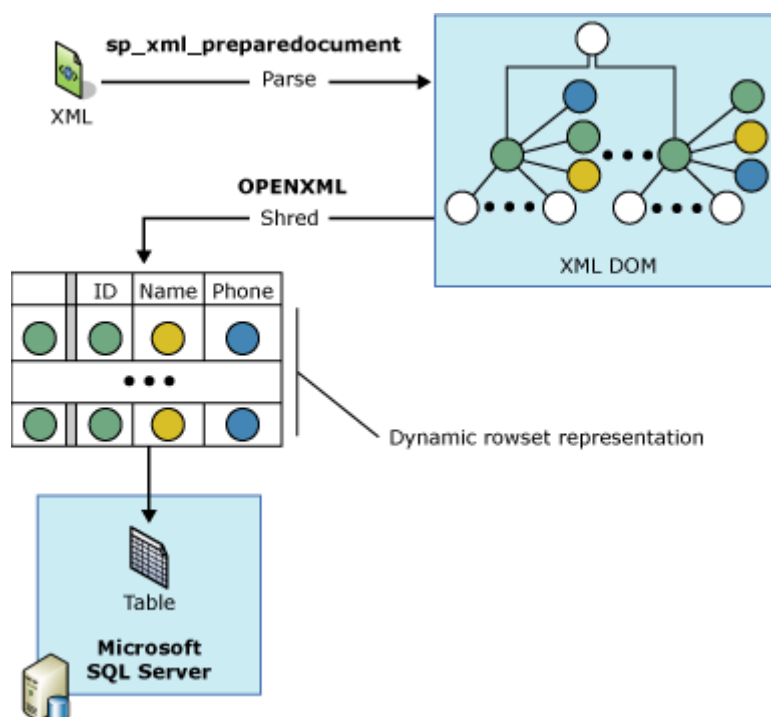


Figura 11: [Fuente](#)

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)



XUNTA DE GALICIA  
CONSELLERÍA DE CULTURA,  
EDUCACIÓN E UNIVERSIDADE



IES de Teis  
Avda. de Galicia, 101  
36216 – Vigo  
886 12 04 64  
ies.teis@edu.xunta.es



Unión Europea-  
NextGenerationEU

Veamos un ejemplo completo. Usaremos un documento `C:\Users\wadmin\Desktop\Ejemplos_UD06\addresses.xml` situado en la carpeta `Ejemplos_UD06` del escritorio de la máquina virtual.

#### **<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

C:\Users\wadmin\Desktop\Ejemplos\_UD06\lectura1.sql:

```
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
```



Results		Messages
(No column name)		
1	<Addresses><Address id="9"><AddressLine1>Avenida...	
	id	
1	9	
2	11	
3	5	

Figura 12: Resultado de ejecución de lectura1.sql

### 3.2.1 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 C:\Users\wadmin\Desktop\Ejemplos\_UD06\lectura2.sql:

```
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 13: Resultado de ejecutar lectura2.sql

### 3.2.1.1 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 C:\Users\wadmin\Desktop\Ejemplos\_UD06\lectura3.sql:

```
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 14: 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**

### 3.2.1.2 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 C:\Users\wadmin\Desktop\Ejemplos\_UD06\lectura4.sql:

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

### 3.3 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 C:\Users\wadmin\Desktop\Ejemplos\_UD06\addresses.xml

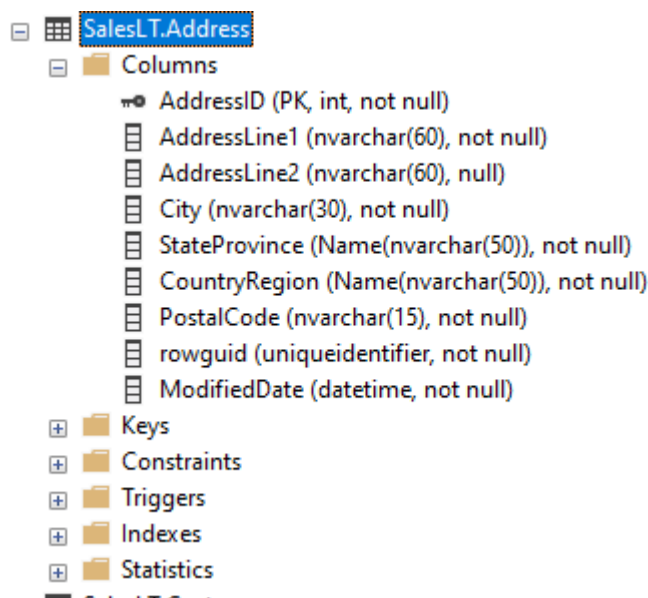


Figura 16: En el explorador, podemos desplegar la carpeta de Columnas para ver los nombres y tipos de las columnas de la tabla SalesLT.Address



Para ello, utilizaremos el script C:\Users\wadmin\Desktop\Ejemplos\_UD06\insercion1.sql:

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

### 3.3.1 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.

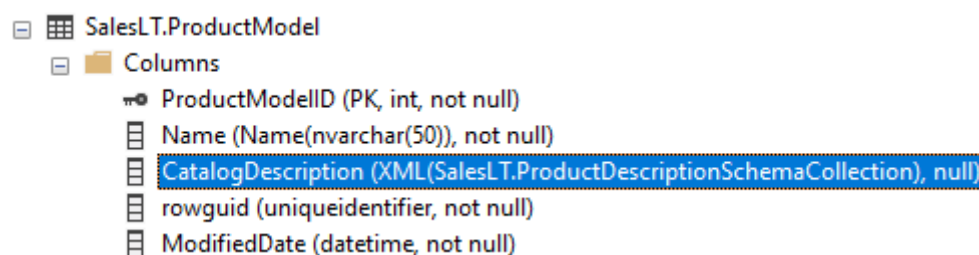


Figura 18: Se observa que la columna CatalogDescription es de tipo XML y además que debe validarse con un esquema que se guarda en una estructura llamada XML SCHEMA COLLECTION: SalesLT.ProductDescriptionSchemaCollection

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	<a href="#">&lt;?xml-stylesheet href="ProductDescription.xsl" type="text/xsl"?&gt;&lt;p 1:Produ</a>
2	23	Mountain-500	<a href="#">&lt;?xml-stylesheet href="ProductDescription.xsl" type="text/xsl"?&gt;&lt;p 1:Produ</a>
3	25	Road-150	<a href="#">&lt;?xml-stylesheet href="ProductDescription.xsl" type="text/xsl"?&gt;&lt;p 1:Produ</a>
4	28	Road-450	<a href="#">&lt;?xml-stylesheet href="ProductDescription.xsl" type="text/xsl"?&gt;&lt;p 1:Produ</a>
5	34	Touring-1000	<a href="#">&lt;?xml-stylesheet href="ProductDescription.xsl" type="text/xsl"?&gt;&lt;p 1:Produ</a>
6	35	Touring-2000	<a href="#">&lt;?xml-stylesheet href="ProductDescription.xsl" type="text/xsl"?&gt;&lt;p 1:Produ</a>

Figura 19: Si se hace clic sobre el texto en azul se podrá ver que se trata de un archivo de XSLT

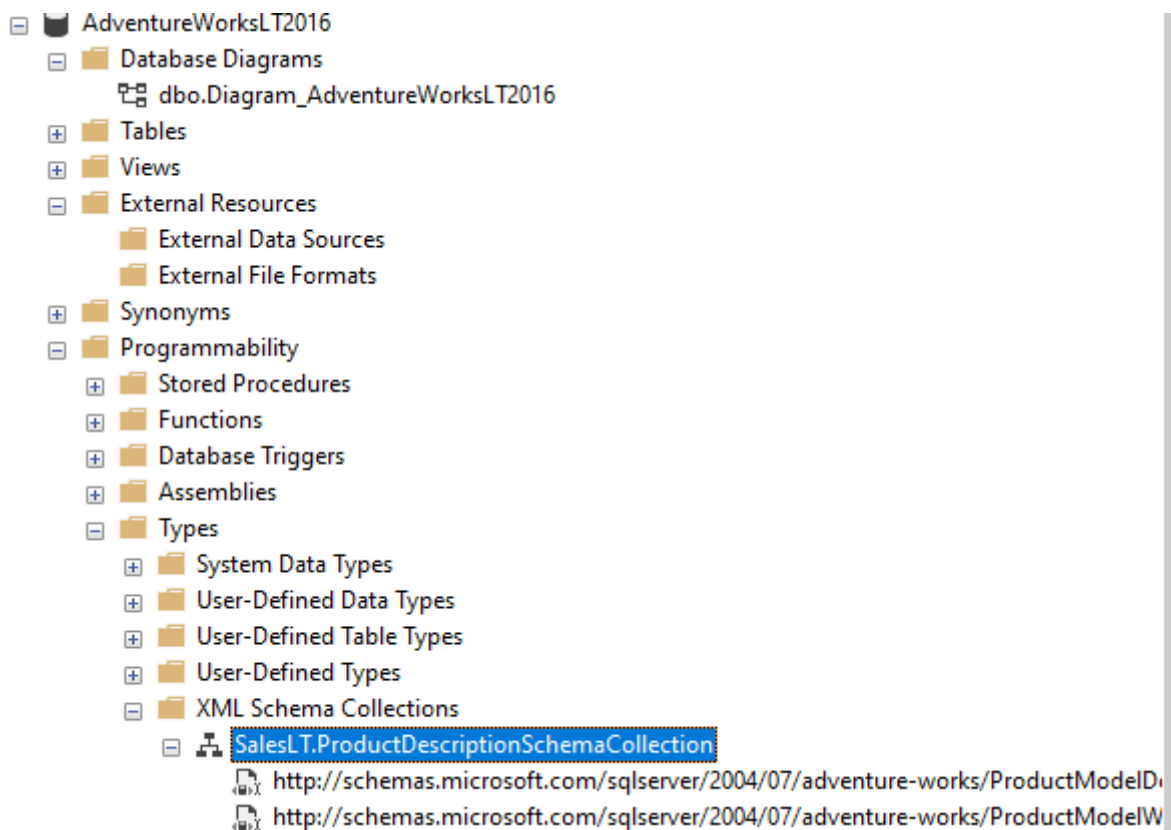


Figura 20: Es posible encontrar el esquema de validación en el explorador gráfico de la base de datos AdventureWorksLT2016 > Programmability > Types > XML Schema Collections

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 C:\Users\wadmin\



Desktop\Ejemplos\_UD06 \insercion2.sql 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
```

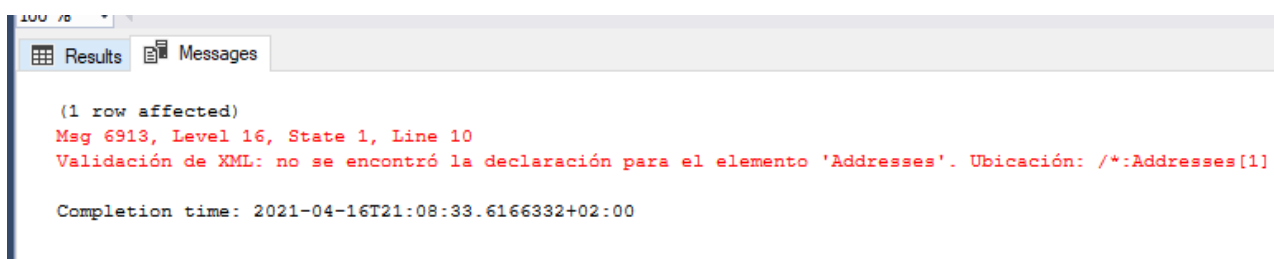


Figura 21: No permite su inserción porque el documento XML no valida con el esquema definido para esa columna



**XUNTA DE GALICIA**  
CONSELLERÍA DE CULTURA,  
EDUCACIÓN E UNIVERSIDADE



**IES de Teis**  
Avda. de Galicia, 101  
36216 – Vigo  
886 12 04 64  
ies.teis@edu.xunta.es



Unión Europea-  
NextGenerationEU

En cambio, si intentamos insertar un documento que sí cumple el esquema, como C:\Users\wadmin\Desktop\Ejemplos\_UD06\CatalogDescription1.xml, habremos logrado su inserción con C:\Users\wadmin\Desktop\Ejemplos\_UD06\insercion3.sql:

```
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%'
```





	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 22: 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
Infracción 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 23: Ejemplo de error que se puede dar si se ejecuta exactamente el mismo script insercion3.sql sin cambiar name.

### 3.3.1.1 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**



**XUNTA DE GALICIA**  
CONSELLERÍA DE CULTURA,  
EDUCACIÓN E UNIVERSIDADE



**IES de Teis**  
Avda. de Galicia, 101  
36216 – Vigo  
886 12 04 64  
ies.teis@edu.xunta.es



Unión Europea-  
NextGenerationEU

## 4 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>