7. SQL Server y XML

Este capítulo contiene apuntes relacionados con el ambiente de SQL Server y con la manera en que procesa documentos XML.

7.1 XML

Etiquetas (tags) comunes para definir esquemas XML

Etiqueta	Uso
<xsd:annotation> y <xsd:documentation>	Comentarios y descripciones en el documento xml.
<xsd:element>	Especificación de un elemento. Se pueden indicar los siguientes atributos en el elemento: ▪ name- nombre del elemento. ▪ type- nombre de la estructura de un elemento complejo, la cual debe describirse en otra parte del esquema. Para elementos simples, este atributo sirve para especificar su tipo de datos. ▪ minOcurrs- cantidad mínima de instancias del elemento. También se puede usar para indicar elementos multivaluados de un elemento complejo; si se van a permitir nulos, su valor sería 0. ▪ maxOcurrs- cantidad máxima de instancias del elemento. Un valor unbounded indica instancias ilimitadas. También se puede usar para indicar elementos multivaluados de un elemento complejo.
<xsd:complexType>	Especificación de la estructura de un elemento (tipo) complejo o de un atributo compuesto.
<xsd:sequence>	Indicación de secuencia de elementos dentro de un tipo complejo o de un atributo compuesto.
<xsd:key>	Especifica un elemento que corresponde a una clave primaria (PK) en una base de datos relacional. El atributo name indica el nombre de dicho elemento.
<xsd:keyref>	Especifica un elemento que corresponde a una clave externa (FK) en la base de datos. El atributo name indica el nombre de dicho elemento y ref el nombre de la clave primaria referida.
<xsd:unique>	Especifica un elemento que tiene valores únicos en la base de datos. El atributo name indica el nombre de dicho elemento.

<xsd:selector> y <xsd:field>	Son etiquetas asociadas con las tres anteriores y sirven para indicar, por medio del atributo path, el elemento complejo y el campo del mismo, respectivamente, donde se localizan la clave primaria, la clave externa o el elemento único indicados con las tres etiquetas previas.
<xsd:string>, <xsd:byte>, <xsd:float>,<xsd:decimal>, <xsd:integer>, <xsd:date>, etc.	Tipos de datos simples.

7.2 Creación de bases de datos en SQL Server

A continuación se describe el proceso a seguir para crear una base de datos en SQL Server utilizando el SQL Server Management Studio como ambiente de acceso al manejador.

1. Ejecutar el SQL Server Management Studio como administrador.

2. En la ventana: Conectar con el servidor, dar los siguientes valores:
   Tipo de servidor: Motor de base de datos
   Nombre del servidor: localhost o el que aparezca como default
   Autenticación: Autenticación de SQL Server
   Inicio de sesión: sa
   Contraseña: sqladmin
   Clic en el botón <Conectar>.

3. En la ventana: Explorador de objetos (izquierda de la pantalla), elegir Bases de datos, menú breve[1], Nueva base de datos…, en Nombre de la base de datos: dar el nombre de la misma (por ejemplo: *BD_XML*), <Aceptar>, en la ventana del Explorador de objetos expandir la carpeta Bases de datos, debiendo aparecer ahí una carpeta con el nombre de la nueva base de datos.

4. Para crear la estructura y guardar las primeras tuplas de la base de datos: seleccionar la carpeta de la base de datos (*BD_XML*), clic en el botón Nueva consulta (está en la barra de herramientas), sobre la parte derecha de la pantalla aparece una ventana en blanco, ahí ya se pueden dar/copiar las instrucciones para crear las tablas y para agregar sus primeras tuplas.

   Notas:
   a. Las fechas deben darse en el formato: 'AAAA-MM-DD' (por ejemplo: '2017-11-17').
   b. Si al insertar fechas en las tablas marca error, entonces darlas en el formato: 'DD-MM-AAAA' (por ejemplo: '17-11-2017').

[1] El término: menú breve, se refiere al botón derecho del ratón.

7.3 SQL Server - XML

7.3.1 Características generales

- SQL Server brinda soporte nativo tanto para datos relacionales como de xml.
- En SQL Server un documento completo xml se puede almacenar como un valor en una sola columna. Por ejemplo en la tabla:

  Company(Id int primary key, Doc xml (Company_Schema))

  la segunda columna almacenará un documento xml por tupla. El documento xml completo es tratado como un solo objeto/valor y al almacenarlo se validaría con el esquema xml definido por Company_Schema.
- En SQL Server, una misma columna xml puede usarse para contener documentos xml con o sin esquema.

7.3.2 Trabajo inicial con SQL Server - XML

Para ejemplificar el uso de XML con SQL Server se usará la base de datos *BD_XML* (punto 3, sección 7.2) la cual contendrá una tabla cuyas tuplas tendrán un campo para almacenar documentos xml. Para esto hay que hacer lo siguiente:

- Abrir una ventana de Nueva consulta para la base de datos *BD_XML* (punto 4, sección 7.2).
- Crear el esquema xml (*Company_Schema*, para el ejemplo), al cual se ajustará el documento xml, usando la siguiente instrucción:

  ```
  create xml schema collection Company_Schema as
  N'Texto del esquema xml';
  ```

  Entre los apóstrofos se debe copiar todo el texto del esquema xml a usar. La codificación que acepta SQL Server es UTF-16[2], así que si el esquema xml tiene una UTF-8, hay que cambiarla a UTF-16. Una vez creado el esquema ya se puede usar para especificar el tipo de datos de un campo de una tabla.
- Para ver la lista de esquemas xml disponibles en la base de datos se puede dar:

  ```
  select * from sys.xml_schema_collections;
  ```
- Crear la tabla con la siguiente instrucción:

  ```
  create table Company(Id int primary key, Doc xml (Company_Schema));
  ```

  Los documentos que se carguen en la columna *Doc* se validarán usando el esquema asociado.

[2] UTF-16 es una forma de codificación de caracteres Unicode. Es capaz de representar cualquier carácter Unicode utilizando símbolos de longitud variable: 1 o 2 palabras, de 16 bits, por carácter Unicode.

- Después ya se pueden insertar documentos xml con:

```
insert into Company
 select 1, Doc
 from
  (select *
  from openrowset
  (bulk 'ruta\company.xml', single_blob) as Doc) as R(Doc);
```

Donde *ruta* es la trayectoria donde se encuentra el archivo xml.

La función openrowset() permite cargar un documento xml desde un archivo. Después de esto ya se pueden hacer consultas sobre el documento xml.

7.4 XPath con SQL Server

7.4.1 Trabajo con XPath en SQL Server (usando el SQL Server Management Studio)

- Abrir una Nueva consulta para la base de datos.
- Dar las expresiones de XPath. El resultado se muestra en el área de resultados correspondiente. Ejemplo: Obtener el nombre de los empleados:

```
select Doc.query('companyDB//employee/fname')
from Company;
```

- La raíz del árbol es companyDB.
- Los nombres de elementos, atributos, etc., de un documento xml **son case sensitive**.

7.4.2 Operadores comunes de XPath usados en las expresiones

Operador (o función)	Uso
/	Hijo del nodo actual
//	Descendiente del nodo actual
[ ]	Para especificar una ruta desde el nodo actual, pero "conservando el nodo actual"
@	Atributo del nodo actual
.	Contexto actual
..	Contexto del padre
*	Indica cualquier elemento
Lógicos: and, or, not	El habitual
De comparación: >, >=, etc.	El habitual

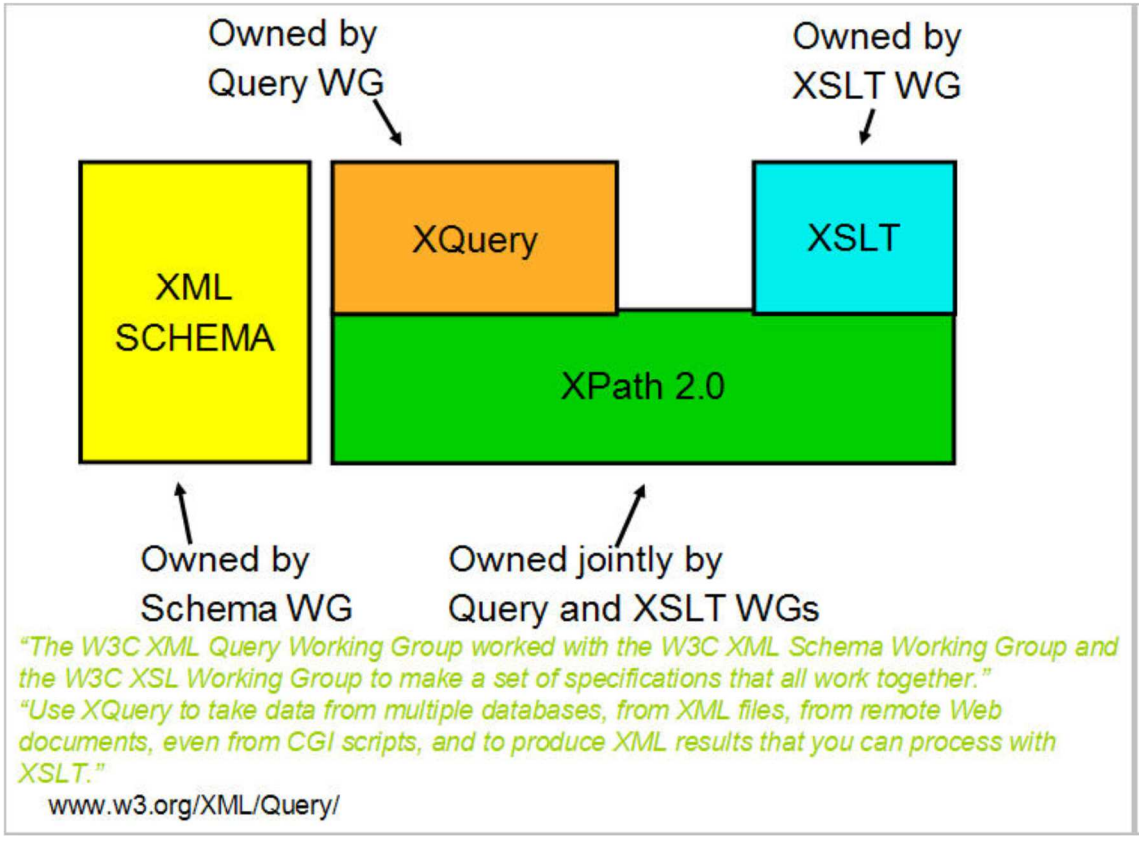
7.4.3 Algunas funciones de XPath

contains(string, string)	Regresa verdadero si el primer parámetro contiene al segundo.
substring(string, pos) substring(string, pos, long)	Regresa la sub-cadena del primer parámetro que inicia en la posición *pos*. Regresa la sub-cadena del primer parámetro que inicia en la posición *pos* con una longitud *long*.
concat(string, string [, …])	Regresa la concatenación de dos o más cadenas.
string-length(string)	Regresa la longitud de la cadena.
distinct-values(element)	Regresa un conjunto de valores distintos.
min(exp), max(exp), count(exp), sum(exp), avg(exp)	Funciones de agregados. *exp* representa una expresión en general, incluyendo una *path expression*.

7.5 XQuery con SQL Server

7.5.1 Características generales

- Es un lenguaje de consultas diseñado para datos XML. Fue desarrollado por la comunidad W3C (World Wide Web Consortium - www.w3c.org) y se ha convertido en un estándar de la industria para consultas XML.



- El cuerpo de una consulta XQuery puede consistir de:
  - Literales y variables
  - Expresiones de trayectoria (XPath expressions)
  - Predicados
  - Comparaciones
  - Expresiones FLWOR

7.5.2 Expresiones FLWOR

- Representan la forma típica de hacer una consulta en XQuery. Tienen la siguiente forma:

  For <variable ligada a nodos individuales (elementos)>
  Let <variable ligada a colecciones completas de nodos (elementos)>
  Where <predicado>
  Order by <variable>
  Return <especificación del resultado de la consulta>

- Características de la expresión FLWOR
  - Puede haber cero o más instancias de la cláusula For. Por medio de una variable se recorre a cada nodo dentro de una secuencia de nodos.
  - También puede haber cero o más instancias de la cláusula Let. Liga la variable a una secuencia completa de nodos (o sea, a un conjunto completo).
  - La cláusula Where es opcional y puede aparecer una vez, máximo. Permite especificar condiciones a satisfacer por los elementos.
  - La cláusula Order by también es opcional y puede aparecer una vez, máximo. Sirve para hacer, principalmente, ordenamientos de elementos, aunque también puede usarse para hacer agrupaciones de elementos.
  - La cláusula Return debe aparecer una sola vez. Puede regresar la información tal como está en el XML o agregar construcciones como se muestra en el ejemplo 1 siguiente.
  - Las variables deben llevar el prefijo $.
  - **En cualquiera de las cláusulas se puede especificar una expresión XPath**.

7.5.3 Trabajo con XQuery en SQL Server (usando el SQL Server Management Studio)

- Abrir una Nueva consulta para la base de datos.
- Dar las expresiones de XQuery. El resultado se muestra en el área de resultados correspondiente. Las {} indican obtener el valor de la(s) variable(s) enmarcada(s). Ejemplo: Listar el nombre y el apellido de los empleados ordenando descendentemente por apellido (el orden default es *ascending*):

```
select Doc.query
('for $x in //employee
order by $x/lname descending
return <resul> {$x/fname,$x/lname} </resul>')
from Company;
        o
select Doc.query
('for $x in //employee
let $y := ($x/fname,$x/lname)
order by $x/lname descending
return <resul> {$y} </resul>')
from Company;
```