XQuery

1. Introducción

XML se ha convertido en una herramienta de uso cotidiano en los entornos de tratamiento de información y en los entornos de programación.

A medida que se emplea un mayor número de proyectos de complejidad y tamaño crecientes y la cantidad de datos almacenados en XML aumenta, se hace necesaria la aparición de un lenguaje que permita definir de forma rápida y compacta consultas o recorridos complejos sobre colecciones de datos en XML.

Este lenguaje debe ser declarativo, es decir, independientemente de la forma en que se realice el recorrido o donde se encuentren los datos.

Además debe combinar las características de los lenguajes tradicionales de consulta con los lenguajes de recuperación de información. Por lo que se debe permitir una selección de datos de la estructura del documento, la búsqueda de información en texto libre y consultas mixtas de contenidos y estructura. Respecto a las salidas, igual que una consulta de una BD relacional devuelve una relación, una consulta XML devuelve un documento XML.

Como solución a esta necesidad surge el lenguaje XML Query.

2. XQuery

XML Query, comúnmente denominado XQuery y publicado por el W3C (la última recomendación es XQuery 3.1), es un lenguaje funcional que utilizando la notación XML permite realizar consultas en infinidad de tipos de diferentes documentos tales como documentos estructurados, colecciones de documentos, bases de datos, estructuras DOM, catálogos, Servicios Web, aplicaciones, sistemas heredados, etc.

XML Query es un lenguaje de consulta diseñado para escribir consultas sobre colecciones de datos expresadas en XML. Su principal función es extraer información de un conjunto de datos organizados como un árbol de orden n de etiquetas XML. Con esa información extraída se puede construir nuevos documentos XML

Aúna diferentes características de otros lenguajes como XPath (comparten el mismo modelo de datos ya poyo a las misas funciones y operadores), SQL, XQL y XML-QL presentando una sintaxis muy similar a la del SQL. Probablemente se convierta en el lenguaje dominante para consultar datos de la mayoría de las fuentes de datos. Aunque está diseñado para consultar datos XML, puede utilizarse para unir datos de múltiples fuentes de datos (en este sentido es independiente del origen de los datos). A este respeto, es mucho más poderoso que SQL.

XML Query es un lenguaje funcional, lo que significa que, en vez de ejecutar una lista de comandos como un lenguaje procedimental clásico, cada consulta es una expresión que es evaluada y devuelve un resultado, al igual que en SQL. Diversas expresiones pueden combinarse de una manera muy flexible con otras expresiones para crear nuevas expresiones más complejas y de mayor potencia semántica.

Resumiendo, XQuery es a XML lo que SQL es a las tablas de base de datos relacionales y fue diseñado para consulta de datos XML.

3. Cómo aprender XQuery

Nuestro proceso de aprendizaje lo haremos:

- Instalando el software de bases de datos nativas BaseX, para su correcto funcionamiento es necesario que esté instalado como mínimo el JRE7. Una vez instalada, la primera que arranquemos podemos editar las preferencias para lenguaje, fijar nuestra carpeta de trabajo, etc)
- Con los ficheros proporcionados por la profesora, deberemos guardarlos en la carpeta destinada a práctica y que debe ser la que fijamos al crear una nueva base de datos.
- Realizando las consultas que se indican en el fichero de ejercicios de XQuery

4. BaseX

Gestor de Bases de Datos XML. Se centra en el almacenamiento, consulta y visualización de documentos de gran tamaño y colecciones XML y JSON. Una interfaz visual permite a los usuarios explorar de forma interactiva los datos y evaluar las consultas en tiempo real (es decir, con cada pulsación de la tecla). BaseX es independiente de la plataforma y se distribuye bajo la licencia BSD.

5. Consultas en XQuery

Una consulta en XQuery es una expresión que lee una secuencia de datos en XML y devuelve como resultado otra secuencia de datos en XML

Un detalle importante es que, a diferencia de lo que sucede en SQL, en XQuery las expresiones y los valores que devuelven son dependientes del contexto, es decir, los nodos que aparecen en el resultado dependen de los namespaces, de la posición donde aparezca la etiqueta raíz del nodo, etc.

5.1. Reglas sintácticas básicas

- XQuery es “case-sensitive”
- Los XQuery elementos, atributos y variables de be ser los nombres XML válidos
- Un valor XQuery puede estar entre comillas simples o dobles
- Una variable de XQuery se define con un $ seguido de un nombre, por ejemplo $libro
- Los XQuery comentarios están delimitados por **(:** y **:)**, por ejemplo (:comentario: XQuery)

5.2. Estructura general de una consulta XQuery

Una consulta XQuery consta de un Prólogo y una Expresión

5.3. Prólogo

Se utiliza para declaraciones de espacios de nombres, de funciones, variables, etc. Es *optativo*

(: ejemplo de prólogo :) →comentario

Declare namespace mio=http://www.mio.es → espacio de nombres

Declare function mifuncion($param){…} → función

Declare variable $pi := 3.1416 → variable

5.4. Expresión XQuery

Es la consulta propiamente dicha

5.5. Origen de los datos

XQuery utiliza las funciones de entrada para identificar el origen de los datos:

- La función doc(URI) devuelve el nodo del documento, o nodo raíz del documento referenciado por un identificador universal de recursos (URI). Esta es la función más habitual para acceder a la información almacenada en archivos.
- La función collection(URI) devuelve una secuencia de nodos referenciados por una URI, sin necesidad de que exista un nodo documento o nodo raíz. Esta es la función más habitual para acceder a información almacenada en una base de datos que tenga capacidad para crear estructuras de datos XML

5.6. Localizaciones de nodos

Son expresiones que siguen la notación XPath (ejercicios 1 y 2)

5.7. Expresiones FLWOR: combinando y reestructurando nodos

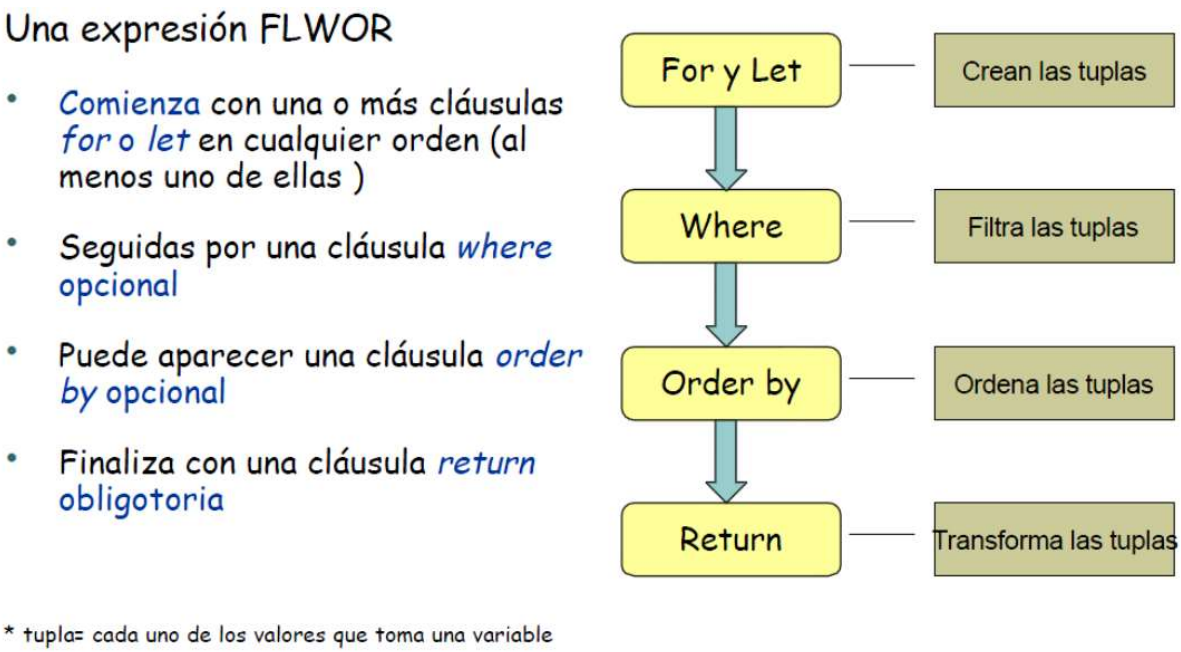
Las consultas en XQuery suelen combinar la información de una o más fuentes de datos reestructurándola para crear un nuevo resultado

Las expresiones FLWOR (siglas de las cláusulas for, let, where, order y return), son una de las características más importantes y típicas de XQuery para conseguirlo. Son similares a SELECT-FROM-WHERE de SQL.

Sin embargo, una expresión FLWOR no está definida en términos de tablas, filsa y columnas; una expresión FLOWR asocia expresiones XPath a las variables de las cláusulas for y let, y utiliza dichas asociaciones para crear nuevos resultados. En XQuery, cuando usamos el término de tupla, nos estamos refiriendo a cada uno de los valores que toma esa asociación.

for	Asocia una (o más) variables a una secuencia de valores, creando una serie de tuplas en el que cada tupla está vinculada a una de las variables	EJEMPLO for $b in doc(libros)/libro let $c:=$b/autor where count ($c)>2 order by $b/titulo return $b/titulo Devuelve los títulos ordenados alfabéticamente de cada uno de los libros que tienen más de dos autores.
let	Asocia una variable al resultado entero de una expresión y devuelve una única tupla, pudiéndose añadir esta nueva asociación a las tupla generadas por una cláusula for si esta existe	
Where	Filtra las tuplas producidas por las cláusulas let y for eliminando todos los valores que no cumplan las condiciones dadas	
order by	Ordena las tuplas según el criterio dado	

return	Construye el resultado de la consulta para cada tupla dada, después de haber sido filtrada por la cláusula where y ordenada por la cláusula order by	



Con estas cláusulas se consigue buena parte de la funcionalidad que diferencia a XQuery de XPath. Entre otras cosas permite construir el documento que será la salida de la consulta

5.8. Diferencias entre for y let

A parte de las diferencias sintácticas (for $x in expresión y let $x:= expresión), se diferencian en los resultados que generan.

La cláusula for asocia una variable con cada nodo que encuentre en la colección de datos, es una secuencia de datos.

La cláusula let, en cambio, asocia una variable con todo el resultado de una expresión.

Consulta	Devuelve
for $d in doc("libros.xml")/bib/libro/titulo return <titulos>{ $d }</titulos>	<titulos> <titulo>TCP/IP Illustrated</titulo> </titulos> <titulos> <titulo>Advan Programming for Unix environment</titulo> </titulos> <titulos> <titulo>Data on the Web</titulo> </titulos> <titulos> <titulo> Economics of Technology for Digital TV</titulo> </titulos>
let $d := doc("libros.xml")/bib/libro/titulo return <titulos>{ $d }</titulos>	<titulos> <titulo>TCP/IP Illustrated</titulo> <titulo>Advan Programming for Unix environment</titulo> <titulo>Data on the Web</titulo> <titulo> Economics of Technology for Digital TV</titulo> </titulos>

5.9. Reglas Generales

Toda consulta escrita en XQuery debe cumplir las siguientes reglas:

- for y let sirven para crear las tuplas con las que trabajará el resto de las cláusulas de la consulta y pueden usarse tantas veces como se desee en una consulta, incluso dentro de otras cláusulas.
- Sin embargo solo pueden declararse una única cláusula where (aunque puede tener condiciones anidadas por and, or, …), una única cláusula order by (aunque es posible especificar varios criterios de ordenación separándolos por comas) y una única cláusula return
- Ninguna de las cláusulas FLWOR es obligatoria en una consulta XQuery

Ejercicios de 3 a 8

5.10. Operadores y funciones principales

El conjunto de funciones y operadores soportado por XQuery es el mismo conjunto de funciones y operadores utilizados en XPath y XSLT.

XQuery soporta operadores y funciones matemáticas, de cadenas, para el tratamiento de expresiones regulares, comparaciones de fechas y horas, manipulación de nodos XML, manipulación de secuencias, comprobación y conversión de tipos y lógica booleana. Además permite definir funciones propias y funciones dependientes del entorno de ejecución del motor XQuery.

- Matemáticos: +, -, *, div, idiv (división entera entre enteros), mod(resto)
- Comparación: =, !=, <, <=, >, >=, not()
- Secuencia: | (unión), intersect, except
- Redondeo: floor(), ceiling(), round()
- Funciones de agrupación: count(), min(), max(), avg(), sum()
- Funciones de cadena: concat(), string-length(), startswith(), ends-with(), substring(), upper-case(), lwer-case(), string()
- Uso general: distinct-values(), empty(), exists()

FunctX XQuery Functions: Hundreds of useful examples

5.11. Constructores de nodos

Si se desea que el resultado de la consulta se aun elemento o atributo, debe incluirse dentro de la cláusula return de la siguiente manera:

- Etiquetas de apertura y cierre del elemento
- Y en medio y entre {} una lista de expresiones que devuelven el contenido del elemento

Ejercicios de 9 a 14

5.12. Uniones

Como ya se ha dicho, for asocia una o más variables a expresiones escritas en XPath, creando una secuencia de tuplas en el que cada tupla está asociada a una de las variables. Es decir, es una iteración.

Hasta ahora solo hemos visto ejemplos sobre como vincular una variable, pero se pueden vincular varias para unir varios documentos (semejante a JOIN de SQL), basta con separarlas con comas

Si en la consulta aparece más de una cláusla for (o más de una variable en una cláusula for), el resultado es el producto cartesiano de dichas variables, es deir, las tuplas generadas cubren todas las posibles combinaciones de los nodos de dichas variables.

(ejercicio15)

5.13. Expresiones constituidas por valores atómicos

Hemos trabajado con expresiones sobre nodos (expresiones XPath), pero XQuery también admite las llamadas expresiones constituidas por valores atómicos. Ejemplos:

let $a := 53 →literal integer

for $a in (1,3,5) → operador, concatena valores para formar una secuencia

let $a := (1 to 3) → devuelve al secuencia, 1,2,3

Etc.

5.14. Iteraciones

La cláusula for puede también utilizarse para repetir un número fijo de veces una acción

(ejercicio 16. Una vez realizada correctamente, observar cómo cambia el resultado si se usa la cláusula let en vez de la cláusula for)

5.15. Variable posicional at

Usada en una cláusula for permite ligar una variable a la posición del elemento en la expresión

(ejercicio 17)

5.16. Expresiones Cuantificadas (Quantified Expressions)

Motivación cuando tienes una lista de elementos y quieres testear para ver si alguno o todos los elementos satisfacen una condición. El resultado será true o false.

Las expresiones cuantificadas tienen un formato similar a una expresión FLWOR con las siguientes dos cambios:

1. Cambiar el for por some o every
2. En vez de where/order/return se utiliza satisfies

Las expresiones cuantificadas cogen una secuencia como entrada y devuelve un booleano true/false como salida.

Ejercicios 18 a 23

5.17. Expresiones condicionales

Además de la cláusula WHERE, XQuery también soporta expresiones condicionales del tipo "if-then-else" con la misma semántica que en los lenguajes de programación más habituales.

La cláusula WHERE de una consulta permite filtrar las tuplas que aparecerán en el resultado, mientras que una expresión condicional nos permite crear una u otra estructura de nodos en el resultado que dependa de los valores de las tuplas filtradas.

La cláusula "else" es obligatoria y debe aparecer siempre en la expresión condicional. El motivo de esto es que toda expresión en XQuery debe devolver un valor. Si no existe ningún valor a devolver al no cumplirse la cláusula "if", devolvemos una secuencia vacía con "else ()";

Ejercicios 24 y 25

6. Obteniendo (X)HTML

También es posible utilizar XQuery para transformar datos XML en otros formatos, como HTML, convirtiéndose XQuery en una alternativa más sencilla y rápida que usar XSLT

7. Fuentes

- http://www.w3.org/TR/xquery/
- http://alarcos.inf-cr.uclm.es/doc/bbddavanzadas/08-09/FUNCIONALIDAD%204.pdf
- http://lenguajes-recuperacion-web.50webs.com/xquery.html
- http://www.xquery.com/tutorials/guided-tour/
- http://www.di.uniovi.es/~labra/cursos/ver06/pres/XMLBD.pdf
- http://www.lsi.us.es/docs/informes/LSI-2005-02.pdf
- Curso «Lenguaje de marcas» impartido por el profesor Oscar Díaz de la Facultad de Informática de la Universidad del País Vasco.
- https://en.wikibooks.org/wiki/XQuery/Quantified_Expressions