Linguaxes de consulta e manipulación

Sumario

Introdución a XQuery

XQuery é unha linguaxe de consulta de datos en formato XML, non só para ficheiros, senón tamén para bases de datos. É a XML o que o SQL é ás bases de datos relacionais.

Como linguaxe de consulta permite atopar e extraer elementos e atributos dende ficheiros ou BBDD XML. Un exemplo de consulta en XQuery podería ser a seguinte:

"Selecciona todos os rexistros cun prezo menor de 10? na miña colección de CD almacenada nun documento XML chamado cd_catalog.xml".

XQuery 1.0 e XPath 2.0 comparten o mesmo modelo de datos, funcións e operadores. De feito, XQuery converteuse nunha recomendación do W3C dende o ano 2007 e, por tanto, é compatible con outras tecnoloxías XML como os espazos de nome, XSLT, XPath, e XML Schema.

O uso de XQuery está amplamente estendido. Algunhas aplicacións concretas desta tecnoloxía son as seguintes:

- Extracción de información para usar nos servizos web.
- Xeración de informes, en especial, extractos de documentos a partir de orixinais.
- Transformación de XML en XHTML, conxuntamente con XSLT.
- Busca de información de interese nos documentos web.

Selección de nodos

XQuery emprega **funcións** para extraer datos dos documentos XML. Por exemplo, a función doc() úsase para abrir un ficheiro XML:

```
doc("books.xml")
```

As **expresións XPath** úsanse para navegar polos elementos do documento XML. Por exemplo, o seguinte ficheiro contén datos sobre libros:

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<bookstore>

<book category="COOKING">
  <title lang="en">Everyday Italian</title>
  <author>Giada De Laurentiis</author>
  <year>2005</year>
```

```
  <price>30.00</price>
</book>

<book category="CHILDREN">
  <title lang="en">Harry Potter</title>
  <author>J K. Rowling</author>
  <year>2005</year>
  <price>29.99</price>
</book>

<book category="WEB">
  <title lang="en">XQuery Kick Start</title>
  <author>James McGovern</author>
  <author>Per Bothner</author>
  <author>Kurt Cagle</author>
  <author>James Linn</author>
  <author>Vaidyanathan Nagarajan</author>
  <year>2003</year>
  <price>49.99</price>
</book>

<book category="WEB">
  <title lang="en">Learning XML</title>
  <author>Erik T. Ray</author>
  <year>2003</year>
  <price>39.95</price>
</book>

</bookstore>
```

A seguinte expresión XPath úsase para seleccionar todos os elementos title:

```
doc("books.xml")/bookstore/book/title
```

(/bookstore selecciona o elemento bookstore; /book selecciona todos os elementos book fillos do elemento bookstore e /title selecciona todos os elementos title fillos de cada elemento book)

A consulta anterior devolverá o seguinte:

```
<title lang="en">Everyday Italian</title>
<title lang="en">Harry Potter</title>
<title lang="en">XQuery Kick Start</title>
<title lang="en">Learning XML</title>
```

XQuery usa **predicados** para limitar os resultados da consulta. No seguinte exemplo selecciónanse todos os elementos book baixo o elemento bookstore que teñen un prezo menor que 30:

```
doc("books.xml")/bookstore/book[price<30]
```

O resultado da consulta XQuery anterior será o seguinte:

```
<book category="CHILDREN">
  <title lang="en">Harry Potter</title>
  <author>J K. Rowling</author>
  <year>2005</year>
  <price>29.99</price>
</book>
```

Expresións FLWOR

Tamén se poden seleccionar nodos mediante expresións FLWOR. Por exemplo, a seguinte expresión XPath selecciona todos os elementos title baixo os elementos book que están baixo o elemento bookstore e que teñen un prezo maior que 30:

```
doc("books.xml")/bookstore/book[price>30]/title
```

A seguinte expresión FLWOR fará exactamente o mesmo que a expresión XPath anterior:

```
for $x in doc("books.xml")/bookstore/book
```

```
where $x/price>30
return $x/title
```

O resultado será:

```
<title lang="en">XQuery Kick Start</title>
<title lang="en">Learning XML</title>
```

As expresións FLWOR son máis potentes que as expresións XPath. Por exemplo, podemos ordenar o resultado:

```
for $x in doc("books.xml")/bookstore/book
where $x/price>30
order by $x/title
return $x/title
```

FLWOR é o acrónimo de FLWOR "For, Let, Where, Order by, Return":

- A cláusula **for** permite gardar todos os elemento book baixo o elemento bookstore nunha variable chamada $x.
- A cláusula **where** selecciona só os elementos book cun elemento prezo maior que 30.
- A cláusula **order by** especifica como se ten que ordenar o resultado. Neste caso ordenarase polo elemento title.
- A cláusula **return** especifica que se ten que devolver. Neste caso o resultado será o elemento title.

O resultado da expresión XQuery anterior será:

```
<title lang="en">Learning XML</title>
<title lang="en">XQuery Kick Start</title>
```

For

A cláusula **for** permite almacenar nunha variable cada elemento devolto pola expresión **in**. É unha estrutura de programación repetitiva ou iterativa. Pode haber múltiples cláusulas for na mesma expresión FLWOR.

Para iterar un número específico de veces hai que usar a palabra reservada to:

```
for $x in (1 to 5)
return <test>{$x}</test>
```

O resultado será o seguinte:

```
<test>1</test>
<test>2</test>
<test>3</test>
<test>4</test>
<test>5</test>
```

A palabra reservada **at** úsase para almacenar o número de iteración:

```
for $x at $i in doc("books.xml")/bookstore/book/title
return <book>{$i}. {data($x)}</book>
```

O resultado:

```
<book>1. Everyday Italian</book>
<book>2. Harry Potter</book>
<book>3. XQuery Kick Start</book>
<book>4. Learning XML</book>
```

Tamén está permitido máis dunha expresión in na cláusula for. Neste caso úsanse comas para separar cada unha:

```
for $x in (10,20), $y in (100,200)
return <test>x={$x} and y={$y}</test>
```

O resultado:

```
<test>x=10 and y=100</test>
```

```
<test>x=10 and y=200</test>
<test>x=20 and y=100</test>
<test>x=20 and y=200</test>
```

Let

A cláusula **let** permite declarar variables e evita repetir a mesma expresión varias veces.

```
let $x := (1 to 5)
return <test>{$x}</test>
```

O resultado:

```
<test>1 2 3 4 5</test>
```

Where

A cláusula **where** úsase para especificar un ou máis criterios para o resultado:

```
where $x/price>30 and $x/price<100
```

Order by

A cláusula **order by** úsase para especificar como se ordena o resultado. No seguinte exemplo ordenamos por categoría e título:

```
for $x in doc("books.xml")/bookstore/book
order by $x/@category, $x/title
return $x/title
```

O resultado:

```
<title lang="en">Harry Potter</title>
<title lang="en">Everyday Italian</title>
<title lang="en">Learning XML</title>
<title lang="en">XQuery Kick Start</title>
```

Return

A cláusula **return** especifica que é o que se quere devolver:

```
for $x in doc("books.xml")/bookstore/book
return $x/title
```

O resultado:

```
<title lang="en">Everyday Italian</title>
<title lang="en">Harry Potter</title>
<title lang="en">XQuery Kick Start</title>
<title lang="en">Learning XML</title>
```

XQuery e HTML

A seguinte expresión FLWOR de XQuery seleccionará todos os elementos title baixo elementos book que estean baixo elementos bookstore e devolverá os títulos ordenados alfabeticamente:

```
for $x in doc("books.xml")/bookstore/book/title
order by $x
return $x
```

Se quixéramos representalos mediante unha lista en HTML poderíamos engadir as etiquetas <ul> e <li> tal e como se amosa no seguinte exemplo:

```
<ul>
{
```

```
for $x in doc("books.xml")/bookstore/book/title
order by $x
return <li>{$x}</li>
}
</ul>
```

O resultado será o seguinte:

```
<ul>
<li><title lang="en">Everyday Italian</title></li>
<li><title lang="en">Harry Potter</title></li>
<li><title lang="en">Learning XML</title></li>
<li><title lang="en">XQuery Kick Start</title></li>
</ul>
```

Se o que queremos é eliminar o elemento title e mostrar só os datos faríamos o seguinte:

```
<ul>
{
for $x in doc("books.xml")/bookstore/book/title
order by $x
return <li>{data($x)}</li>
}
</ul>
```

E o resultado será unha lista HTML:

```
<ul>
<li>Everyday Italian</li>
<li>Harry Potter</li>
<li>Learning XML</li>
<li>XQuery Kick Start</li>
</ul>
```

Engadir atributos ao HTML

Tamén é posible engadir atributos de elementos XML como atributos nunha lista HTML. No seguinte exemplo, úsase o atributo category como un atributo class da lista HTML:

```
<html>
<body>

<h1>Bookstore</h1>

<ul>
{
for $x in doc("books.xml")/bookstore/book
order by $x/title
return <li class="{data($x/@category)}">{data($x/title)}</li>
}
</ul>

</body>
</html>
```

O resultado será o seguinte:

```
<html>
<body>
<h1>Bookstore</h1>

<ul>
<li class="COOKING">Everyday Italian</li>
<li class="CHILDREN">Harry Potter</li>
<li class="WEB">Learning XML</li>
<li class="WEB">XQuery Kick Start</li>
</ul>

</body>
```

```
</html>
```

Terminoloxía

XQuery usa a terminoloxía de XPath. Por tanto, os conceptos de nodo fillo, ancestros, etc. son iguais que en XPath.

Sintaxe

Regras sintácticas de XQuery

As regras básicas de sintaxe son as seguintes:

- Distínguese entre maiúsculas de minúsculas.
- Os elementos, atributos e variables deben ser nomes XML válidos.
- As cadeas poden ir entre comiñas simples ou dobres.
- As variables están definidas mediante o símbolo $ seguida polo nome, ie: $bookstore
- Os comentarios están delimitados por (: e :), por exemplo: (: Comentario XQuery :)

Expresións condicionais

Pódense usar expresións condicionais do tipo "If-Then-Else". Por exemplo:

```
for $x in doc("books.xml")/bookstore/book
return if ($x/@category="CHILDREN")
then <child>{data($x/title)}</child>
else <adult>{data($x/title)}</adult>
```

É obrigatorio usar parénteses na expresión if. A palabra reservada else é obrigatoria pero pode estar baleira. Isto indicarémolo mediante else ().

O resultado do exemplo de enriba será:

```
<adult>Everyday Italian</adult>
<child>Harry Potter</child>
<adult>Learning XML</adult>
<adult>XQuery Kick Start</adult>
```

Comparacións

Hai dúas formas de comparar valores:

1. **Comparacións xerais**: =, !=, <, <=, >, >=
2. **Comparacións de valor**: eq, ne, lt, le, gt, ge

Vexamos a diferenza cun exemplo. A seguinte expresión devolve verdadeiro se algún atributo q ten un valor maior que 10:

```
$bookstore//book/@q > 10
```

A seguinte expresión devolve verdadeiro se hai só un atributo q devolto pola expresión XPath e o seu valor e maior que 10. Se se devolve máis dun atributo q prodúcese un erro:

```
$bookstore//book/@q gt 10
```

Funcións

XQuery 1.0, XPath 2.0, e XSLT 2.0 comparten a mesma biblioteca de funcións predefinidas (máis de 100). Hai funcións para manexar cadeas, números, datas e tempo, manipulación de nodos e secuencias, valores boolean, etc. Tamén é posible definir funcións propias que se axusten ás nosas necesidades.

O URI do espazo de nomes das funcións XQuery é http://www.w3.org/2005/02/xpath-functions. O prefixo por defecto é **fn:**.

Para chamar a unha función úsase o prefixo fn: por exemplo, fn:string(). Con todo, xa que fn: é o prefixo por defecto do espazo de nomes os nomes das funcións non precisan o prefixo cando se chaman..

Pódese consultar a referencia completa das funcións predefinidas de XQuery.

Vexamos algúns exemplos:

Exemplo 1:

```
<name>{uppercase($booktitle)}</name>
```

Exemplo 2:

```
doc("books.xml")/bookstore/book[substring(title,1,5)='Harry']
```

Exemplo 3:

```
let $name := (substring($booktitle,1,4))
```

Funcións definidas polo usuario

Para definir unha función empregamos a seguinte sintaxe:

```
declare function prefix:function_name($parameter AS datatype)
AS returnDatatype
{
 ...código da función...
}
```

Cando se declara unha función hai que ter en conta o seguinte:

- Hai que usar a palabra reservada **declare**.
- O nome da función debe ter un prefixo.
- Os tipos de datos dos parámetros son os mesmos que os definidos nos esquemas XML.
- O corpo da función debe ir entre chaves.

Vexamos un exemplo:

```
declare function local:minPrice($p as xs:decimal?,$d as xs:decimal?)
AS xs:decimal?
{
let $disc := ($p * $d) div 100
return ($p - $disc)
}
```

Pódese chamar á función anterior do seguinte xeito:

```
<minPrice>{local:minPrice($book/price,$book/discount)}</minPrice>
```

Ejemplos de ficheros XML

Archivo:Bib.xml

Referencias

Os contidos deste apartado están baseados no material sobre XQuery do sitio web http://www.w3schools.com.