

XQuery

Una consulta XQuery és una expressió que llegeix dades d'un o més documents XML i retorna com a resultat una altra seqüència de dades en XML. XQuery conté a XPath, tota expressió de consulta en XPath és vàlida i retorna el mateix resultat en XQuery.

XQuery permet:

- Seleccionar informació basada en un criteri específic.
- Cercar informació en un document o conjunt de documents.
- Unir dades des de múltiples documents o col·lecció de documents.
- Organitzar, agrupar i resumir dades.
- Transformar i reestructurar dades XML en un altre vocabulari o estructura.
- Fer càlculs aritmètics sobre nombres i dates.
- Manipular cadenes de caràcters a format de text.

1. Norma FLWOR

En XQuery les consultes segueixen la norma FLWOR (llegit com flower en anglès), correspon a la sigles de For, Let, Where, Order i Return. Permet a diferència de XPath manipular, transformar i organitzar els resultats de les consultes. La sintaxi general d'una estructura FLWOR és aquesta:

for <variable> in <expressió XPath>

let <variables vinculades>

where <condició XPath>

order by <expressió>

return <expressió de sortida>

En aquests apunts utilitzarem l'exemple books.xml i bookscat.xml

books.xml

```
<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>

```

bookscat.xml

```

<?xml version="1.0" encoding="ISO-8859-1"?>
<bookstore>
  <category name="COOKING">
    <book>
      <title lang="en">Everyday Italian</title>
      <author>Giada De Laurentiis</author>
      <year>2005</year>
      <price>30.00</price>
    </book>
  </category>
  <category name="CHILDREN">
    <book>
      <title lang="en">Harry Potter</title>
      <author>J K. Rowling</author>
      <year>2005</year>
      <price>29.99</price>
    </book>
  </category>
  <category name="WEB">
    <book>
      <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>
      <title lang="en">Learning XML</title>
      <author>Erik T. Ray</author>
      <year>2003</year>
      <price>39.95</price>
    </book>
  </category>
</bookstore>

```

- **For:** S'utilitza per seleccionar nodes i emmagatzemar-los en una variable, similar a la clàusula from de SQL. Dins del for escrivim una expressió XPath que seleccionarà els nodes. Si s'especifica més d'una variable en el for actua com producte cartesià. Les variables comencen amb \$.

Les consultes XQuery han de portar obligatòriament una ordre Return, on s'indica el què es vol que retorni la consulta.

XQuery

XPath

for \$b in doc("books.xml")/bookstore/book return \$b	/bookstore/book
---	-----------------

Resultat



XQuery

XPath

for \$b in doc("books.xml")/bookstore/book return \$b/title	/bookstore/book/title
---	-----------------------

Resultat

```
<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>
```

- **Let:** Permet que s'assignin valors resultants d'expressions XPath a variables per simplificar la representació. Es poden posar vàries línies let una per cada variable o separar les variables per comes.

Dues variables

```
for $b in doc("books.xml")/bookstore/book
let $tit:=$b/title, $y:=$b/year
order by $b/year
return <tit_year>{data($tit)} -- {data($y)}</tit_year>
```

Resultat

```
<tit_year>XQuery Kick Start -- 2003</tit_year>
<tit_year>Learning XML -- 2003</tit_year>
<tit_year>Everyday Italian -- 2005</tit_year>
<tit_year>Harry Potter -- 2005</tit_year>
```

Sense for

Amb for

<pre>let \$b := doc("books.xml")/bookstore/book let \$tit:= \$b/title return <titles>{\$tit}</titles></pre>	<pre>for \$b in doc("books.xml")/bookstore/book let \$tit:= \$b/title return <titles>{\$tit}</titles></pre>
---	---

Resultat

Sense for

Amb for

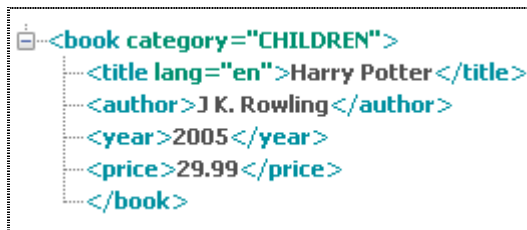
<pre><titles> <title lang="en">Everyday Italian</title> <title lang="en">Harry Potter</title> <title lang="en">XQuery Kick Start</title> <title lang="en">Learning XML</title> </titles></pre>	<pre><titles> <title lang="en">Everyday Italian</title> </titles> <titles> <title lang="en">Harry Potter</title>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

Fixeu-vos que no és el mateix un resultat que l'altre en l'exemple anterior:

- Si s'agafa la opció sense for, només amb let, el què fa és agafar tot el resultat que retorna l'expressió XPath que s'assigna a la variable \$tit i després es retorna tota sencera, dins d'un element titles
 - Si s'agafa l'opció amb for, per cada element title es retorna un element titles
- **Where:** Filtra els elements, eliminant tots els valors que no compleixen les condicions donades.

```
for $b in doc("books.xml")/bookstore/book
where $b/price<30
return $b
```

Resultat



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

- **Order by:** Ordena les dades segons el criteri donat.

```
for $b in doc("books.xml")/bookstore/book
order by $b/year descending
return <book>{data($b/year)}--{data($b/title)}</book>
```

Resultat



```
<book>2005--Everyday Italian</book>
<book>2005--Harry Potter</book>
<book>2003--XQuery Kick Start</book>
<book>2003--Learning XML</book>
```

- **Return:** Construeix el resultat de la consulta en XML, s'hi poden afegir etiquetes XML a la sortida. Si s'afegeixen etiquetes, les dades a visualitzar s'han de tancar entre {}. A més a més, en el return es poden afegir condicionals utilitzant **if-then-else** i així tenir més versatilitat a la sortida.
- S'ha de tenir en compte que la clàusula else és obligatòria i ha d'aparèixer sempre en l'expressió condicional, es deu a què tota expressió XQuery ha de retornar un valor. Si no s'hagués de tornar cap valor en no complir-se la clàusula if, retornem una seqüència buida amb else().
- Utilitzarem la funció data() per extreure el contingut en text dels elements i per extreure el contingut dels atributs.

If

```
for $b in doc("books.xml")/bookstore/book
return if($b/year=2005)
then <year2005>{data($b/title)}</year2005>
else()
```

Quan es vol donar el resultat només a la part de l'if, també s'ha de posar l'else encara que estigui buit. És a dir, else().

Resultat

```
<year2005>Everyday Italian</year2005>
<year2005>Harry Potter</year2005>
```

If else

```
for $b in
doc("books.xml")/bookstore/book
return if($b/year=2005)
then <year2005>{data($b/title)}</year2005>
else <year2003>{data($b/title)}</year2003>
```

Resultat

```
<year2005>Everyday Italian</year2005>
<year2005>Harry Potter</year2005>
<year2003>XQuery Kick Start</year2003>
<year2003>Learning XML</year2003>
```

If else if

```
for $b in
doc("books.xml")/bookstore/book
return if($b/year=2005)
then <year2005>{data($b/title)}</year2005>
else if($b/year=2003)
then <year2003>{data($b/title)}</year2003>
else ()
```

Resultat

```
<year2005>Everyday Italian</year2005>
<year2005>Harry Potter</year2005>
<year2003>XQuery Kick Start</year2003>
<year2003>Learning XML</year2003>
```

Atributs sense data()

Retorna els atributs amb l'etiqueta

```
let $b :=
doc( "books.xml" )/bookstore/book
let $cat:=$b/@category
return $cat
```

Atributs amb data()

Retorna les dades de dins dels atributs

```
let $b :=
doc( "books.xml" )/bookstore/book
let $cat:=data($b/@category)
return $cat
```

Resultat

```
category="COOKING"
category="CHILDREN"
category="WEB"
category="WEB"
```

```
● xs:untypedAtomic = COOKING
● xs:untypedAtomic = CHILDREN
● xs:untypedAtomic = WEB
● xs:untypedAtomic = WEB
```

Elements sense data()

Retorna els elements amb les etiquetes

```
let $b :=
doc( "books.xml" )/bookstore/book
return $b
```

Elements amb data()

Retorna les dades de dins dels elements

```
let $b :=
doc( "books.xml" )/bookstore/book/
return data($b)
```

Resultat

```
<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>
```

```
● xs:untypedAtomic = Everyday Italian
● xs:untypedAtomic = Harry Potter
● xs:untypedAtomic = XQuery Kick Start
● xs:untypedAtomic = Learning XML
```

Restringir nodes al for i where

Agafa primer només els llibres de l'any 2003 i al where agafa els que tenen la categoria WEB i en retorna els seus autors.

```
for $b in
doc( "books.xml" )/bookstore/book[year="2003" ]
let $cat:=$b/@category
let $aut:=$b/author
where $cat="WEB"
return $aut
```

Restringir nodes al for sense where

Agafa directament només els llibres de l'any 2003 i els que tenen la categoria WEB i en retorna els seus autors.

```
for $b in
doc( "books.xml" )/bookstore/book[year="2003" and @category="WEB" ]
let $aut:=$b/author
return $aut
```

Resultat (dels dos exemples anteriors)

```
<author>James McGovern</author>
<author>Per Bothner</author>
<author>Kurt Cagle</author>
<author>James Linn</author>
<author>Vaidyanathan Nagarajan</author>
<author>Erik T. Ray</author>
```

2. Operacions i funcions més comunes en XQuery

Les operacions i funcions suportats per XQuery pràcticament són les mateixes que les suportades per XPath. Suporta operadors i funcions matemàtiques, de cadenes, pel tractament d'expressions regulars, comparacions de dates i hores, manipulació de nodes XML, manipulació de seqüències, comprovació i conversió de tipus i lògica booleana. Els operadors i funcions més comuns es mostren en la següent taula.

Matemàtiques	+, -, *, div (en comptes de /), idiv (divisió entera), mod
Comparació	=, !=, <, >, <=, >=, not()
Seqüència	union(), intersect, except
Arrodoniment	floor(), ceiling(), round()
Agrupació	count(), min(), max(), avg(), SUM()
Cadenes	concat(), string-length(), starts-with(), ends-with(), substring(), upper-case(), lower-case(), string()

Genèriques	distinct-values()	elimina nodes duplicats
	empty()	retorna cert si l'expressió dins del parèntes està buida
	exists()	retorna cert si una seqüència conté com a mínim un element
Comentaris	(: Això és un comentari :)	

starts-with

```
for $tit in doc("books.xml")/bookstore/book/title
where starts-with($tit,"X")
return $tit
```

Resultat

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

Retorn de dades amb concat

Retorn de dades sense concat

<pre>for \$b in doc("books.xml")/bookstore/book return <book>{concat(data(\$b/year), ' -- ',data(\$b/title))}</book></pre>	<pre>for \$b in doc("books.xml")/bookstore/book return <book>{data(\$b/year)} -- {data(\$b/title)}</book></pre>
--	---

Resultat (és el mateix per als dos casos anteriors)

```
<book>2005 -- Everyday Italian</book>
<book>2005 -- Harry Potter</book>
<book>2003 -- XQuery Kick Start</book>
<book>2003 -- Learning XML</book>
```

Càlculs amb distinct-values o sense distinct-values

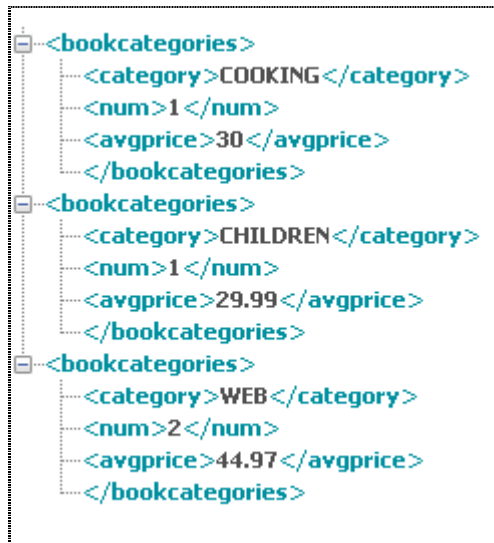
Si tenim uns elements dels què en volem comptar els que hi ha segons un atribut o un altre element, si no està agrupat dins d'aquell altre element o atribut s'ha de fer servir distinct-values perquè ens ho agrupi per aquell element o atribut.

Aquest és el cas del següent exemple (Fer càlculs al return amb books.xml), en l'exemple posterior s'ha canviat el fitxer books.xml de manera que s'agrupessin els llibres dins d'elements category, i que cada llibre que fós d'una categoria anés dins de l'element category corresponent. En aquest cas no cal fer servir distinct-values, perquè cada categoria hi surt només un cop dins del fitxer.

- **Càlculs amb distinct-values (books.xml)**

```
for $cat in
distinct-values(doc("books.xml")/bookstore/book/@category)
let $b:=doc("books.xml")/bookstore/book[@category=$cat]
let $pri:=$b/price
let $numb:=count($b)
return <bookcategories><category>{data($cat)}</category>
<num>{$numb}</num><avgprice>{avg($pri)}</avgprice></bookcategories>
```

Resultat

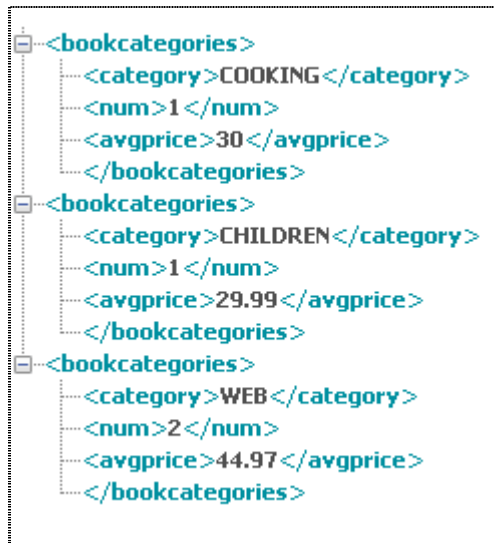


```
<bookcategories>
  <category>COOKING</category>
  <num>1</num>
  <avgprice>30</avgprice>
</bookcategories>
<bookcategories>
  <category>CHILDREN</category>
  <num>1</num>
  <avgprice>29.99</avgprice>
</bookcategories>
<bookcategories>
  <category>WEB</category>
  <num>2</num>
  <avgprice>44.97</avgprice>
</bookcategories>
```

- **Càlculs sense distinct-values (bookscat.xml)**

```
for $cat in
doc("bookscat.xml")/bookstore/category
let $b:=$cat/book
let $pri:=$b/price
let $numb:=count($b)
return <bookcategories><category>{data($cat/@name)}</category>
<num>{$numb}</num><avgprice>{avg($pri)}</avgprice></bookcategories>
```

Resultat



Els dos exemples anteriors donen el mateix resultat.

3. Altes, modificació, reanomenament i esborrat de nodes en documents XML

• Inserció de node

```
insert node  
  
<book category="CHILDREN">  
  <title lang="en">New Book</title>  
  <author>New Author</author>  
  <year>2013</year>  
  <price>19.99</price>  
</book>  
  
before doc("books.xml")/bookstore/book[1]
```

Es pot utilitzar per a inserir:

```
before/after ....  
as first/as last into ....
```

• Modificació de valor d'un node

```
replace value of node doc("books.xml")/bookstore/book[2]/title  
with "Romeo"
```

- **Modificació de tot un node**

```
replace node doc("books.xml")/bookstore/book[3]/title with  
<TITLE>test</TITLE>
```

- **Reanomenar**

- 1 node

```
rename node doc("books.xml")/bookstore/book[1] as "BOOK"
```

- Varis nodes

```
for $b in doc("books.xml")/bookstore/book  
return rename node $b  
as "BOOK"
```

- Canviar a majúscules tots els nodes

```
for $x in doc("books.xml")//(*|@*)  
return  
rename node $x  
as upper-case(name($x))
```

- **Eliminació de nodes**

```
delete node doc("books.xml")/bookstore/book[@category =  
'CHILDREN']
```