Engenharia de Dados e Conhecimento

XQUERY LANGUAGE

universidade de aveiro

W3C Recommendation 21 March 2017

http://www.w3.org/TR/xquery-31/

XQuery 3.1: An XML Query Language

What is XQuery?

- XQuery is a standardized language for combining documents, databases, web pages and almost anything else. It is very widely implemented. It is powerful and easy to learn.

- XQuery is replacing proprietary middleware languages and web application development languages. XQuery is replacing complex Java or C++ programs with a few lines of code.

- XQuery is simpler to work with and easier to maintain than many other alternatives.

- Do more with less.

XQuery?

- XQuery is <u>THE</u> language for querying XML data
  - Like SQL for RDBMS

- XQuery is a language to search, find and extract information from elements and attributes of XML documents.

What for?

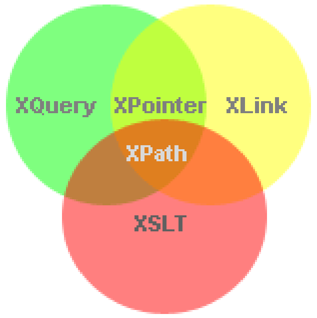
universidade de aveiro

- XQuery can be used to:
  - Extract information from a database for use in a web service.
  - Generate summary reports on data stored in an XML database.
  - Search textual documents on the Web for relevant information and compiling the results.
  - Select and transforming XML data to XHTML to be published on the Web.
  - Pull data from databases to be used for the application integration.
  - Split up an XML document that represents multiple transactions into multiple XML documents.

Context

universidade de aveiro

- XQuery is built on XPath expressions
  - XQuery 3.1 and XPath 3.1 share the same data model and support the same functions and operators.



- XQuery is a W3C Recommendation
  - It's compatible with several W3C standards, such as XML, Namespaces, XSLT, XPath, and XML Schema

XQuery as a Language

- XQuery is a functional language
  - Every query is an expression to be evaluated
  - Expressions can be combined to create new expressions
  - It has:
    - Path expressions
    - Element constructors
    - FLWOR expressions
    - Sorting, conditional and quantified expressions
    - Operators and functions (built-in and user-defined)
    - Use of XML Schema types

The Language - Literals

- Comments

  (: This is a comment :)

- Numeric Literals

  1; -2; +3 are integer (type xs:integer)

  5.36; -6.63; +7.43 are decimal (type xs:decimal)

  2.3e6; -2.3E6 are double (type xs:double)

- String Literals

  ‘This is a string’ (type xs:string)

  “<p>This is a string</p>” (type xs:string)

The Language – Input

- XQuery uses input functions to identify the data to be queried.
  - **doc()** - returns an entire document (document node), identified by a URI.
    - Example: doc(“videos.xml”) – data in a file named “videos.xml”
  - **collection()** - returns a collection (nodes sequence), associated with a URI.
    - Used to identify a database, or dataset, to use in a query.
      - Example: collection(“videos”) – data in a database, or dataset, named “videos”.

The Language - Locating Nodes

- The Xquery’s path expressions are identical to Xpath
- Examples:
  - collection(“videos”)/videos/video – set of ‘video’ elements
  - collection(“videos”)/videos/video/title[1] – set of first titles found
  - (collection(“videos”)/videos/video/title)[1] – the first title found

Namespaces

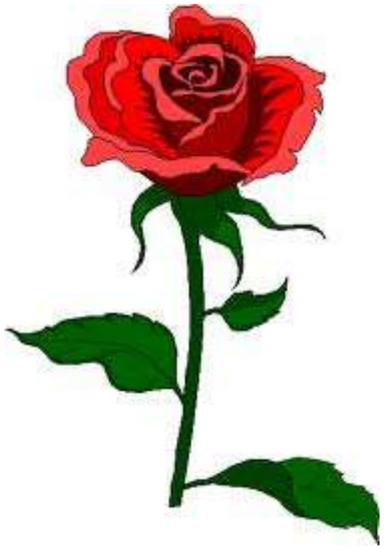
- The use of namespaces:

  declare namespace vi=http://videos.com.pt

  collection(‘videos’)//vi:genre – set of video genres

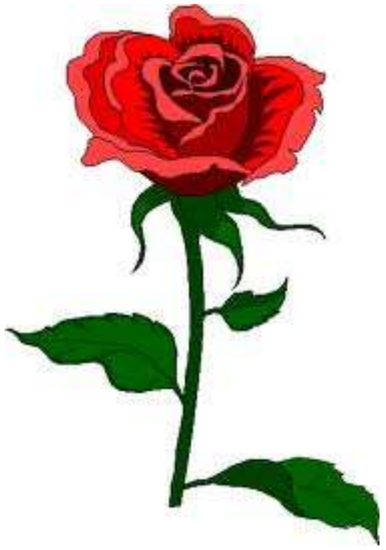
XQuery – FLWOR

- FLWOR expressions are very powerful and common expressions in XQuery.
- They are similar to the SELECT-FROM-WHERE statements in SQL.
- A FLWOR expression is not defined in terms of tables, rows, and columns.
- A FLWOR expression binds variables to values in **for** and **let** clauses, and uses these variable bindings to create new results.
- A combination of variable bindings created by the for and let clauses of a FLWOR expression is called a tuple.



XQuery – FLWOR Structure

- The XQuery FLWOR
  - **For**
    - Each item in an XPath sequence
  - **Let**
    - A new variable have a specified value
  - **Where**
    - A condition expressed in XPath is true
  - **Order By**
    - the value of an XPath expression
  - **Return**
    - A sequence of items



XQuery – For and Let

- Every clause in a FLWOR expression is defined in terms of tuples
- **For** and **Let** clauses create the tuples
- Every FLWOR expression must have at least one **For** or **Let** clause.

XQuery – FLWOR Explained

- **For**: associate one or more variables to expressions, creating several tuples
- **Let**: bind variables to the entire result of an expression, creating one single tuple
- **Where**: filter tuples, retaining only those tuples that satisfy a condition
- **Order By**: sort the tuples in a tuple stream
- **Return**: build the result of the FLWOR expression for a given tuple

XQuery – 1st Example

XQuery:

```
for $i in (100, 200, 300)
return
  <tuple><i>{ $i }</i></tuple>
```

Result:

```
<tuple><i>100</i></tuple>
<tuple><i>200</i></tuple>
<tuple><i>300</i></tuple>
```

XQuery – 2nd Example

XQuery:

```
let $i := (10, 20, 30)
return
  <tuple><i>{ $i }</i></tuple>
```

Result:

```
<tuple><i>10 20 30</i></tuple>
```

XQuery – 3rd Example

XQuery:

```
for $i in (100, 200, 300)
let $j := (10, 20, 30)
return
  <tuple><i>{ $i }</i><j>{ $j }</j></tuple>
```

Result:

```
<tuple><i>100</i><j>10 20 30</j></tuple>
<tuple><i>200</i><j>10 20 30</j></tuple>
<tuple><i>300</i><j>10 20 30</j></tuple>
```

XQuery – 4th Example

XQuery:

```
let $xml:=
  <a>
    <one>number one!</one>
    <two>number two!</two>
  </a>

return $xml//one/text()
```

Result:

```
number one!
```

XML – Example (books.xml)

universidade de aveiro

```
<?xml version="1.0"?>

<bookstore>
  <book category="COOKING">
    <title lang="en">Everyday Italian</title>
    <author>Giada De Laurentiis</author>
    <year>2005</year>
    <price>30.00</price>
    <isbn>1</isbn>
  </book>
...
</bookstore>
```

XQuery – Examples (FLR)

universidade de aveiro

XQuery:

```
for $b in collection(“books")//book
let $c := $b/author
return <book>{ $b/title, <authors>{ count($c) }</authors>}</book>
```

Result:

```
...
<book>
  <title lang="en">Harry Potter</title>
  <authors>1</authors>
</book>
<book>
  <title lang="en">XQuery Kick Start</title>
  <authors>5</authors>
</book>
...
```

XQuery – Examples (FWR)

XQuery:

```
for $x in collection(“books")//book
where $x/price>30
return $x/title
```

Result:

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

XQuery – Examples (FLWR)

XQuery:

```
for $b in collection(“books")//book
let $c := $b/author
where count($c) > 2
return $b/title
```

Result:

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

XQuery – Examples (FOR)

universidade de aveiro

XQuery:

```
<ul>
{
for $b in collection("various")//book
order by $b/author[1]
return <li>{ data($b/title) }</li>
}
</ul>
```

Result:

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

XQuery – Examples (distinct)

XQuery:

```
let $b := collection('various')//book
for $d in distinct-values($b/year)
return <year>{$d}</year>
```

Result:

```
<year>2005</year>
<year>2003</year>
```

XQuery – Examples (if - else)

universidade de aveiro

XQuery:

```
for $x in collection("various")//book
return
if ($x/@category="CHILDREN") then
   <child>{data($x/title)}</child>
else
   <adult>{data($x/title)}</adult>
```

Result:

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

XQuery – Examples (contains)

universidade de aveiro

XQuery:

```
<books-complete-info>
{
  for $x in collection("various")//book
    where $x/isbn>3
          and (contains($x/title, "XQuery")
          or contains($x/title, "XML"))
    order by $x/title
  return <book>{$x/title}</book>
}
</books-complete-info>
```

Result:

```
<books-complete-info>
  <book>
    <title lang="en">Learning XML</title>
  </book>
</books-complete-info>
```

XML – Example (videos.xml)

universidade de aveiro

```
<?xml version="1.0"?>
<result>
  <video_template>
    <title/>
    <genre>
      <choice>action</choice>
      <choice>comedy</choice>
    …
  <actors>
    <actor id="00000015">Anderson, Jeff</actor>
    <actor id="00000030">Bishop, Kevin</actor>
    …
  <videos>
    <video id="id1235AA0">
      <title>The Fugitive</title>
      <genre>action</genre>
      <rating>PG-13</rating>
    …
```

XQuery – Examples (i)

universidade de aveiro

XQuery:

```
<videos>
 {
   let $doc := collection("videos")
   for $v in $doc//video,
       $a in $doc//actors/actor
   where ends-with($a, "Lisa")
         and $v/actorRef = $a/@id
   order by $v/year
   return
     <video year="{$v/year}">
         {$v/title}
     </video>
 }
</videos>
```

Result:

```
<videos>
  <video year="1999">
    <title>Enemy of the State</title>
  </video>
  <video year="1999">
    <title>Clerks</title>
  </video>
</videos>
```

Built-in Functions

universidade de aveiro

- text()
- data()
- count()
- distinct-values()
- contains()
- start-with()
- ends-with()
- …
  - (https://www.w3.org/TR/xpath-functions-31/)