XQuery

XQuery? What is it ?

XQuery is a language for transformation and creation of XML trees

XQuery is able to combine information from multiple documents

XQuery is declarative (nonprocedural)

XQuery is coordinated with XML Schema and it supports simple and complex data types

XQuery has both XML and human-readable syntax

XQuery ver 1.0 is a superset of XPath ver 2.0

XQuery and XSLT overlap

A sample XML document

```
<bib>
   <book year="1994">
      <title>TCP/IP Illustrated</title>
      <author>
         <last>Stevens</last><first>W.</first>
      </author>
      <author>
         <last>Potter</last><first>H.</first>
      </author>
      <author>
         <last>Bond</last><first>J.</first>
      </author>
      <publisher>Addison-Wesley</publisher>
      <price> 65.95</price>
   </book>
   <book year="1992">
      <title>Advanced Programming in the Unix environment
      </title>
      <author>
         <last>Stevens</last><first>W.</first></author>
      <publisher>Addison-Wesley</publisher>
      <price>65.95</price>
   </book>
   ...
```


FLWOR expressions

FLWOR = For-Let-Where-Order by-Return

```
for $b in //book
let $y := $b/title
where $b/publisher = "Addison-Wesley"
order by $y ascending
return $y
```

```
for $a in //book
for $b in //book
let $x := $a/title
let $y :=$b/title
where $x = $y
order by $x ascending
return <a>{$x}.{$y}</a>
```


For and Let bindings

```
for $x in(1,2)
let $y := ("a", "b", "c")
return($x, $y)
```

generates an output:

```
1, a, b, c, 2, a, b, c
```

```
let $x := (1,2)
for $y in ("a", "b", "c")
return($x, $y)
```

generates an output:

```
1, 2, a, 1, 2, b, 1, 2, c
```

```
for $x in (1,2)
for $y in ("a", "b", "c")
return($x, $y)
```

generates an output:

```
1, a, 1, b, 1, c, 2, a, 2, b, 2, c
```


Example 1

List the books published by Addison-Wesley after 1991 and include their year and title

```
for $b in /bib/book
where $b/publisher = "Addison-Wesley" and
      $b/@year > 1991
return
      <book year="{ $b/@year }">
        { $b/title }
      </book>
```

```
<book year="1994">
   <title>TCP/IP Illustrated</title>
</book>
<book year="1992">
   <title>Advanced Programming in the Unix environment</title>
</book>
```

Example 2

Create a flat list of all the title-author pairs, with each pair enclosed in a "result" element

```
for $b in /bib/book
for $t in $b/title
for $a in $b/author
return
   <result>
     { $t }
     { $a }
   </result>
```

```
<result>
   <title>TCP/IP Illustrated</title>
   <author>
      <last>Stevens</last>
      <first>W.</first>
   </author>
</result>
...
```


Example 3

For each book in the bibliography, list the title and authors, grouped inside a "result" element

```
for $b in /bib/book
return
   <result>
     { $b/title }
     { $b/author  }
   </result>
```

```
<result>
   <title>TCP/IP Illustrated</title>
   <author>
         <last>Stevens</last>
         <first>W.</first>
   </author>
</result>
...
```


Example 4

For each author in the bibliography, list the author's name and the titles of all books by that author, grouped inside a "result" element

```
let $a := //author
for $last in distinct-values($a/last)
for $first in distinct-values($a[last=$last]/first)
return
  <result>
    { $last, $first }
    {
      for $b in /bib/book
       where some $ba in $b/author satisfies
             ($ba/last = $last and $ba/first=$first)
       return $b/title
    }
  </result>
```

```
<result>Stevens W.<title>TCP/IP Illustrated</title>
   <title>Advanced Programming in the Unix environment</title>
</result>
...
```


Example 6

List the titles and years of all books published by Addison-Wesley after 1991, in alphabetic order

```
<bib>
  {
    for $b in //book
    where $b/publisher = "Addison-Wesley" and $b/@year > 1991
    order by $b/title
    return
        <book>
            { $b/@year }
            { $b/title }
        </book>
  }
</bib>
```

```
<bib>
   <book year="1992">
      <title>Advanced Programming in the Unix environment</title>
   </book>
   <book year="1994">
      <title>TCP/IP Illustrated</title>
   </book>
</bib>
```


Example 7

For each book that has at least one author, list the title and first two authors, and an empty "et-al" element if the book has additional authors

```
<bib>
  {
    for $b in //book
    where count($b/author) > 0
    return
        <book>
            { $b/title }
            {
                for $a in $b/author[position()<=2]
                return $a
            }
            {
                if (count($b/author) > 2)
                 then <et-al/>
                 else ()
            }
        </book>
  }
</bib>
```


Example 7

For each book that has at least one author, list the title and first two authors, and an empty "et-al" element if the book has additional authors

```
<bib>
   <book>
      <title>TCP/IP Illustrated</title>
      <author>
         <last>Stevens</last>
         <first>W.</first>
      </author>
      <author>
         <last>Potter</last>
         <first>H.</first>
      </author>
      <et-al/>
   </book>
   <book>
      <title>Advanced Programming in the Unix environment</title>
      <author>
         <last>Stevens</last>
         <first>W.</first>
      </author>
   </book>
...
```

Example 8

Find books in which an element contains a string "Unix" somewhere in its content. For each such book, return a title

```
for $b in //book
let $e := $b/*[contains(string(.), "Unix")]
where exists($e)
return
    <book>
        { $b/title }
    </book>
```

```
<book>
  <title>Advanced Programming in the Unix environment</title>
</book>
```


Example 9

For each book with an author, return the book with its title and authors. For each book with an editor, return a reference with the book title and the editor's affiliation

```
<bib>
{
        for $b in //book[author]
        return
            <book>
                { $b/title }
                { $b/author }
            </book>
}
{
        for $b in //book[editor]
        return
          <reference>
            { $b/title }
            {$b/editor/affiliation}
          </reference>
}
</bib>
```


Example 9

For each book with an author, return the book with its title and authors. For each book with an editor, return a reference with the book title and the editor's affiliation

```
<bib>
   …
   <book>
      <title>Advanced Programming in the Unix environment</title>
      <author>
         <last>Stevens</last>
         <first>W.</first>
      </author>
   </book>
…
  <reference>
      <title>Java for kids</title>
      <affiliation>CACM</affiliation>
   </reference>
</bib>
```


References

Elmasri R., Navathe S., Fundamentals of Database Systems, 6th edition, chapter 12 XML: Extensible Markup Language, pp. 420-448

`http://www.uow.edu.au/~jrg/235/HOMEWORK`

9.2 How to use XQuery query language ?