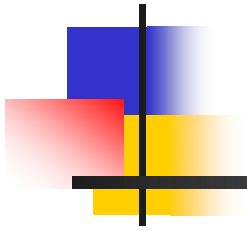
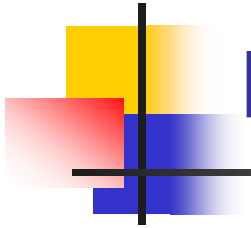


Introduction to XQuery

Jaana Holvikivi

Metropolia

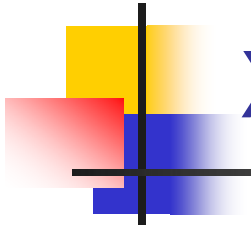


Recommendation and uses

- XQuery 1.0 became a W3C Recommendation January 23, 2007.
- XQuery is to XML what SQL is to database tables.
- XQuery is designed to query XML data - not just XML files, but anything that can appear as XML, including databases.
- XQuery 1.0 and XPath 2.0 share the same data model and support the same functions and operators.

Example: students.xml

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="students.xsl"?>
<students>
<student id="8080">
      <name>Mary Lamb</name>
      <age>22</age>
     <major>Electronics</major>
     <major>Embedded systems</major>
<results>
<result course="XML" grade="3"/>
<result course="Web programming" grade="4"/>
<result course="Advanced Java" grade="5"/>
</results>
</student>
<student id="7766">
      <name>Carl Wolff</name>
      <age>32</age>
     <major>Systems engineering</major>
<results>
<result course="XML" grade="1"/>
<result course="Web programming" grade="4"/>
<result course="Laplace transformations" grade="2"/>
</results>
</student>
</students>
```



XQuery example

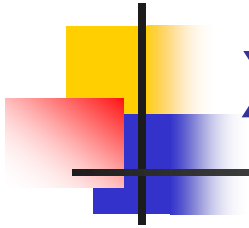
Find all grades that are higher than 3!

student.xquery -file:

```
doc("students.xml")/students/student/results/result
    [@grade > 3]
```

Note: not XML syntax!

Result not well-formed XML



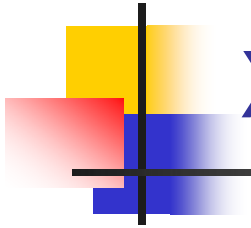
XQuery examples

Find all grades that are higher than 3 and print them in well-formed XML!

student.xquery -file:

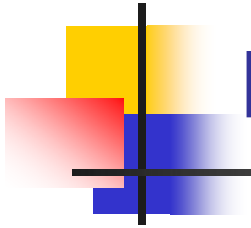
```
<grades>
{doc("students.xml")/students/student/results/result [@grade > 3]
}
</grades>
```

Tip: work outside in to have all brackets and commas in place



XQuery Basic Syntax Rules

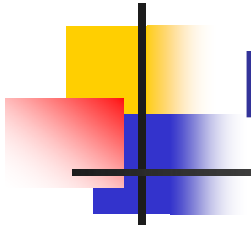
- XQuery is case-sensitive
- XQuery elements, attributes, and variables must be valid XML names
- An XQuery string value can be in single or double quotes
- An XQuery variable is defined with a $ followed by a name, e.g. $bookstore
- XQuery comments are delimited by (: and :), e.g. (: XQuery Comment :)



FLWOR

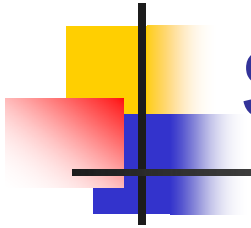
- doc("books.xml")/bookstore/book[price>30]/title
  - select all the title elements under the book elements that are under the bookstore element that have a price element with a value that is higher than 30.
- The FLWOR expression:

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



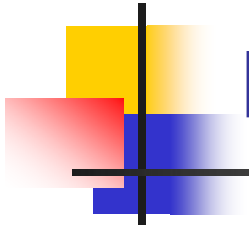
FLWOR clauses

- FLWOR is an acronym for "For, Let, Where, Order by, Return".
- The for clause selects all book elements under the bookstore element into a variable called $x.
- let clauses bind single variables
- The where clause selects only book elements with a price element with a value greater than 30.
- The order by clause defines the sort-order. Will be sort by the title element.
- The return clause specifies what should be returned. Here it returns the title elements.



Selecting and Filtering Elements

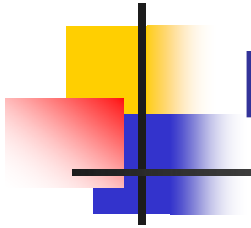
- for $x in doc("books.xml")/bookstore/book
  where $x/price>30
  order by $x/title
  return $x/title



For ... in

- for $x in (1, 2, 3, 4, 5) return <item>{$x} </item>

- yields
  <item>1</item>
  <item>2</item>
  <item>3</item>
  <item>4</item>
  <item>5</item>

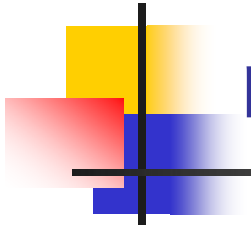


Finding multiple majors

```
for $s in fn:doc("students.xml")//student
let $m := $s/major
where fn:count($m) ge 2
order by $s/@id
return <double> { $s/name/text() } </double>
```

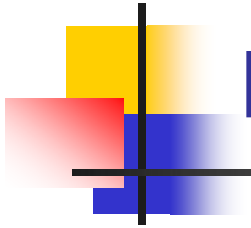
filtering with *where*

sorting using order *by*



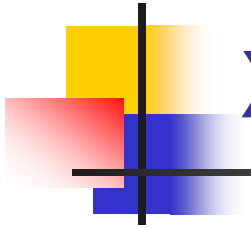
majors fixed

```
<doubles>
{
      for $s in fn:doc("students.xml")//student
      let $m := $s/major
      where fn:count($m) ge 2
      order by $s/@id
      return <double> { $s/name/text() } </double>
}
</doubles>
```



Predicates in XPath

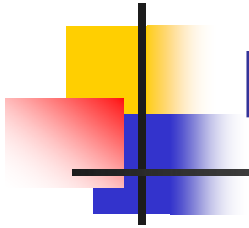
- Predicates are used to find a specific node or a node that contains a specific value.
- Predicates are always embedded in square brackets.
- //Section[@security = "confidential"]
- //Section[@security = "public"] [@version = "final"]
- /bookstore/book[price>35.00]/title
- /bookstore/book[last()-1]



XQuery Built-in Functions

The URI of the XQuery function namespace is http://www.w3.org/2005/02/xpath-functions

- The default prefix for the function namespace is fn:
- Functions are often called with the fn: prefix, such as fn:string()
- Since fn: is the default prefix of the namespace, the function names do not need to be prefixed when called.
- doc() function specifies the xml document that you want to query



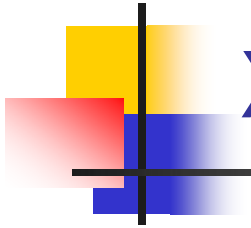
Examples of Function Calls

- A call to a function can appear where an expression may appear. Look at the cases below:

1: In an element
<name>{uppercase($booktitle)}</name>

2: In the predicate of a path expression
doc("books.xml")/bookstore/book[substring(title,1,5) ='Harry']

3: In a let clause
let $name := (substring($booktitle,1,4))



XQuery Comparisons

In XQuery there are two ways of comparing values.

1. General comparisons: =, !=, <, <=, >, >=

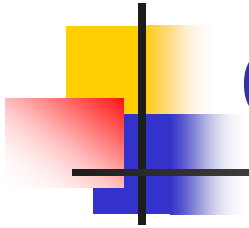
   $bookstore//book/@q > 10

   expression returns true if any q attributes have a value greater than 10

2. Value comparisons: eq, ne, lt, le, gt, ge

   $bookstore//book/@q gt 10

   expression returns true if there is only one q attribute returned by the expression, and its value is greater than 10. If more than one q is returned, an error occurs

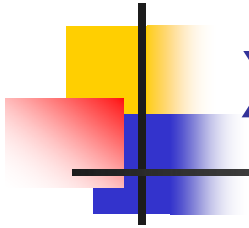


Conditional Expressions

- 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>

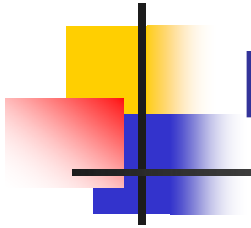
"if-then-else" syntax:

- parentheses around the if expression are required.
- else is required, but it can be just else ()



Xquery to create elements

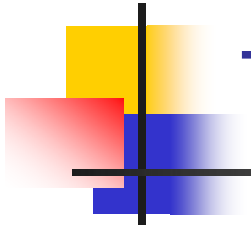
```
element card {
  namespace {"http://businesscard.org"},
  element name {text { "John Doe" }},
  element title {text { "CEO, Widget Inc." }},
  element email {text { "john.doe@widget.com" }},
  element phone {text { " (202) 555-1414 " }},
  element logo {
     attribute uri { "widget.gif" }
}}
```



Result

Instance document:

```
<card xmlns="http://businesscard.org">
<name>John Doe</name>
<title>CEO, Widget Inc.</title>
<email>john.doe@widget.com</email>
<phone>(202) 555-1414</phone>
<logo uri="widget.gif"/>
</card>
```



The prolog and namespaces

- Version declaration
  xquery version "1.0";

declare boundary-space preserve;

declare boundary-space strip;

declare default element namespace *URI;*

declare namespace xs=
  "http://www.w3.org/2001/XMLSchema";
  (not needed because implicitly defined)

import schema at *URI;*

declare variable $seven as xs:integer :=7*;*