# Introduction to XQuery

Dr James Cummings

Oxford Text Archive

University of Oxford

ahds literature, languages and linguistics

# What is XQuery?

- Officially "XML Query", but everyone calls it XQuery
- It is a domain-specific method for accessing and manipulating XML
- It is meant for querying XML
- It is built upon XPath
- It is like SQL but for XML
- It is a W3C recommendation


ahds literature, languages and linguistics

# XQuery Expressions

- path expressions (return a nodeset)
- element constructors (return a new element)
- FLWOR expressions (like SQL 'SELECT')
- list expressions (operate on lists or nodesets)
- conditional expressions (if then else)
- qualified expressions (boolean operations over nodesets)
- datatype expressions (test datatypes of values)


ahds literature, languages and linguistics

# Path Expressions

- This is the XPath part of XQuery:

```
//p/foreign[@lang='lat']

//foreign[@lang='lat']/text()

document('test.xml')//p

collection('/db/PC')//person//surname
```

# Element Constructor

- May contain literal text and/or variables:

```
<latin>o tempora o mores</latin>

<latin>{$s}</latin>

<ul><li>item one is {$one}</li>
<li>item two doesn't exist</li></ul>
```

ahds literature, languages and linguistics

# FLWOR Expression

- For - Let - Where - Order – Return
  - **for** defines a cursor over an XPath selection
  - **let** defines a name for the contents of an XPath
  - **where** selects from the nodes as in SQL
  - **order** sorts the results as in SQL
  - **return** specifies the XML fragments to be constructed
- Curly braces are used for grouping, and define the scope of the **for** clause
- This is one of the most common forms of XQuery, and is often used for the equivalent of SQL joins


ahds literature, languages and linguistics

# FLWOR Expression Example

- **For** every <text> element in the database of XML documents
- **Let** the variable $lats point to any <foreign> child (with 'lang' attribute of 'lat') of the <text> element we are currently processing
- **Where** there is more than one Latin phrase ($lat)
- **Order** these by the number phrases
- **Return** a new <latin> element with $lats and that text's id attribute

```
for  $t in //text
let $lats := $t//foreign[@lang='lat']
where count($lats) > 1
order by count($lats)
return
<latin>{$lats}<txt>{$t/@id}</txt> </latin>
```

ahds literature, languages and linguistics

# List Expressions

- XQuery expressions manipulate lists of values, for which many operators are supported:
  - constant lists: (7, 9, <thirteen/>)
  - integer ranges: i to j
  - XPath expressions
  - concatenation
  - set operators: | (or union), intersect, except
  - functions: remove, index-of, count, avg, max, min, sum, distinct-values ...

ahds literature, languages and linguistics

# List Expressions (nodesets)

- When lists are viewed as nodesets:
  - XML nodes are compared on node identity
  - duplicates are removed
  - the order is preserved

ahds literature, languages and linguistics

# Conditional Expressions

- Usually used in user-defined functions:

```
<div>
  {
    IF document("test.xml")//title/text()
     ="XQuery Test"
    THEN <p>This is true.</p>
    ELSE <p>This is false.</p>
  }
</div>
```

ahds literature, languages and linguistics

# Qualified Expressions (some)

- some in satisfies:

```
for $b in document("book.xml")//text
where some $p in $b//p satisfies
  (contains($p,"sailing") AND
contains($p,"windsurfing") )
return
$b/ancestor::teiHeader//title[1]
```

ahds literature, languages and linguistics

# Qualified Expressions (every)

- every in satisfies:

```
for $b in document("book.xml")//text
where every $p in $b//p satisfies
  contains($p,"sailing")
return $b/ancestor::teiHeader//title[1]
```

ahds literature, languages and linguistics

# Datatype Expressions

- XQuery supports all datatypes from XML Schema, both primitive and complex types
- Constant values can be written:
  - as literals (like string, integer, float)
  - as constructor functions (true(), date("2001-06-07"))
  - as explicit casts (cast as xsd:positiveInteger(47))
- Arbitrary XML Schema documents can be imported into an XQuery
- An instance of operator allows runtime validation of any value relative to a datatype or a schema
- A typeswitch operator allows branching based on types


ahds literature, languages and linguistics

# eXist: Looking For Words

- We are going to be using the eXist native XML Database to practice our XQueries. It has some useful text searching capabilities. For example:

**//p &= 'fish dutch'**

- This will find paragraphs containing both the words fish and dutch (in either order), and is rather easier to type than the equivalent xpath:

**//p[contains(.,'fish') and contains(.,'dutch')]**

- In eXist you can also do a proximity search:

**//p[near(.,'fish dutch',20)]**

- as well as stem matching:

**//p &= 'fish*'**


ahds literature, languages and linguistics

# FLWOR Quiz:

- What does the following do and return?

```
(: This is a how you do a comment :)
declare namespace tei="http://www.tei-c.org/ns/1.0";
let $countryList :=//tei:teiCorpus//tei:taxonomy[@id='Country']
for $person in //tei:TEI//tei:person
let $title := $person/ancestor::tei:TEI/descendant::tei:title[1]/text()
let $nationality := $person/tei:nationality/@code
let $forename := $person/tei:persName/tei:foreName
let $surname :=  $person/tei:persName/tei:surname
let $nation := $countryList/tei:category[@id=$nationality]/tei:catDesc/text()
order by $nationality
return
<ul><li>Title: {$title}</li>
<li>Name: {concat($forename,' ',$surname)}</li>
<li>Country: {$nation} ({string($nationality)})</li></ul>
```

ahds literature, languages and linguistics

# Exercises

- If we have time, there are some quick XQuery exercises for you to do
- Knoppix:
  - Knoppix should already be loaded
  - Go back to Firefox and eXist's 'Basic XQuery Interface'
- You should have this XQuery summary


ahds literature, languages and linguistics