XQUERY

What is XQuery?

- **XQuery** is an XML query language that makes use of XPath to query XML structures

- It also allows for functions to be defined and called, as well as complex querying of data structures using **FLWOR** expressions

- Whereas **XPath** is a way of locating specific elements in an XML tree

In short

- **XQuery** is the language for querying XML data
- **XQuery** for XML is like SQL for databases
- **XQuery** is built on XPath expressions

Symbols

.	Denotes the current node.
..	Denotes the parent of the current node.
/	Denotes the root node, or a separator between steps in a path.
//	Denotes descendants of the current node.
@	Denotes attributes of the current node.
*	Denotes "any" (node with unrestricted name).
[ ]	Brackets enclose a Boolean expression that serves as a predicate for a given step.
[n]	When a predicate consists of an integer, it serves to select the element with the given ordinal number from a list of elements.

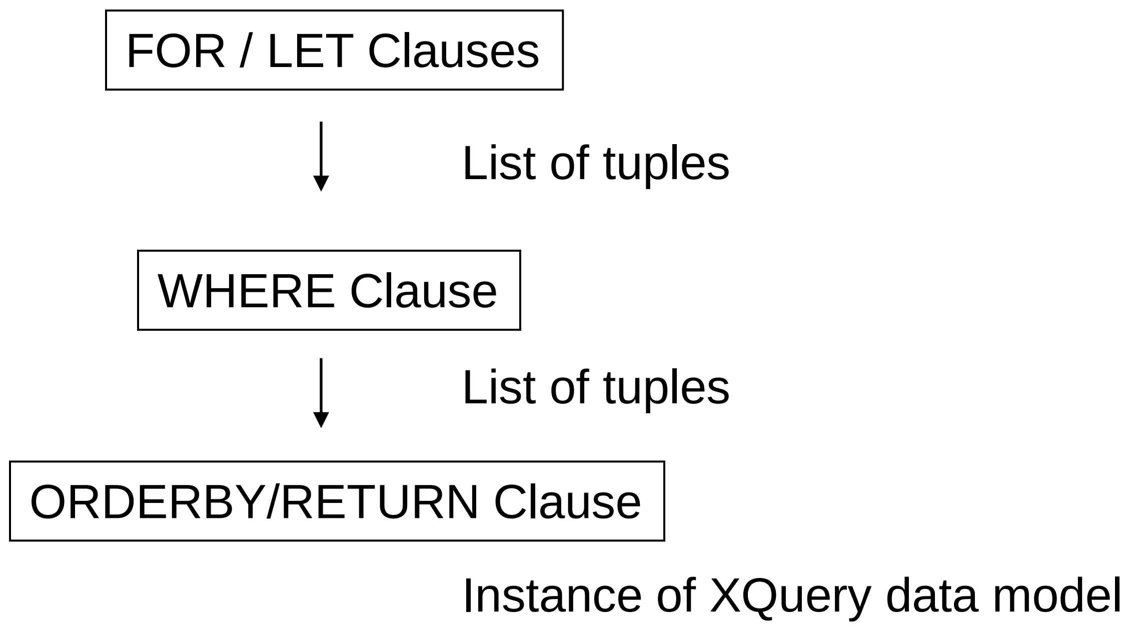
FLWR (“Flower”) Expressions

FOR ... LET... FOR... LET...

WHERE...

RETURN...

FLWOR Expressions



FOR vs. LET

- FOR $x IN list-expr
  - Binds $x in turn to each value in the list expr

- LET $x = list-expr
  - Binds $x to the entire list expr
  - Useful for common sub-expressions and for aggregations

FOR vs. LET: Example

```
FOR $x IN document("bib.xml")/bib/book
RETURN <result> $x </result>
```

Returns:

```
<result> <book>...</book></result>
<result> <book>...</book></result>
<result> <book>...</book></result>
...
```

Notice that result has several elements

```
LET $x IN document("bib.xml")/bib/book
RETURN <result> $x </result>
```

Returns:

```
<result> <book>...</book>
         <book>...</book>
         <book>...</book>
          ...
</result>
```

Notice that result has exactly one element

WHERE

- AND, OR, and NOT usually contain references to bound variables
- Variables bound in FOR clause usually contain scalar predicates

```
$p/color = "Red"
```

- Variables bound in LET clause usually used in list predicates

```
avg($p/price) > 100
```

Operators

- Allows expressions to be constructed using prefix and infix operators
- Standard arithmetic and logical operators
- “=“ “!=“ “<“ “>” “+” “-” “*”
- Many built-in functions

Operators in Expressions

- XQuery allows expressions to be constructed using prefix and infix operators (BEFORE, AFTER
- XQuery contains usual logical and arithmetic operators
- Also operators like UNION, INTERSECT, and EXCEPT

Quantifiers

- Tests for existence of some elements that satisfy a condition
- Also used to test whether all elements in a collection satisfy a condition
- Key words satisfies and contains

XQuery Example 1

Find all book titles published after 1995:

```
FOR $x IN document("bib.xml")/bib/book

WHERE $x/year > 1995

RETURN $x/title
```

Result:

```
<title> abc </title>
<title> def </title>
<title> ghi </title>
```

XQuery Example 2

For each author of a book by Morgan Kaufmann, list all books she published:

```
FOR $a IN distinct(document("bib.xml")
                    /bib/book[publisher="Morgan Kaufmann"]/author)

RETURN <result>

            $a,

            FOR $t IN /bib/book[author=$a]/title

            RETURN $t

          </result>
```

distinct = a function that eliminates duplicates (after converting inputs to atomic values)

Results for Example 2

```
<result>
    <author>Jones</author>
    <title> abc </title>
    <title> def </title>
</result>
<result>
     <author> Smith </author>
     <title> ghi </title>
</result>
```

XQuery Example 3

Find publishers who have more than 100 books published

```
<big_publishers>
    FOR $p IN distinct(document("bib.xml")//publisher)
    LET $b := document("bib.xml")/book[publisher = $p]
    WHERE count($b) > 100
    RETURN $p
</big_publishers>
```

For each publisher p

- Let the list of books published by p be b

Count the # books in b, and return p if b > 100

count = (aggregate) function that returns the number of elements

XQuery Example 4

Find books whose price is larger than average of all books:

```
LET $a=avg(document("bib.xml")/bib/book/price)
FOR $b in document("bib.xml")/bib/book
WHERE $b/price > $a
RETURN $b
```

Collections in XQuery

- Ordered and unordered collections
  - /bib/book/author = an ordered collection
  - Distinct(/bib/book/author) = an unordered collection
- LET $a = /bib/book → $a is a collection
- $b/author → a collection (several authors...)

```
RETURN <result> $b/author </result>
```

Returns:

```
  <result> <author>...</author>
           <author>...</author>
           <author>...</author>
           ...
</result>
```

Collections in XQuery

What about collections in expressions ?

- $b/@price → list of n prices
- $b/@price * 0.7 → list of n numbers
- $b/@price * $b/@quantity → list of n x m numbers ??
- $b/@price * ($b/@quant1 + $b/@quant2) ≠ $b/@price * $b/@quant1 + $b/@price * $b/@quant2 !!

Sorting in XQuery

Find the title of books published with a publisher with their price is descending order and publisher name in ascending order

```
<publisher_list>
   FOR $p IN distinct(document("bib.xml")//publisher)
   RETURN <publisher> <name> $p/text() </name> ,
                  FOR $b IN
document("bib.xml")//book[publisher = $p]
                  RETURN <book>
                                  $b/title ,
                                  $b/@price
                          </book> SORTBY(price
DESCENDING)
              </publisher> SORTBY(name)
</publisher_list>
```

If-Then-Else

```
FOR $h IN //holding
RETURN <holding>
          $h/title,
          IF $h/@type = "Journal"
              THEN $h/editor
          ELSE $h/author
       </holding> SORTBY (title)
```

Existential Quantifiers

A quantified expression determines whether some or all of the items in a sequence meet a particular condition

Find titles of books in which both sailing and windsurfing are mentioned in the same paragraph.

```
FOR $b IN //book
WHERE SOME $p IN $b//para SATISFIES
   contains($p, "sailing")
   AND contains($p, "windsurfing")
RETURN $b/title
```

Universal Quantifiers

if you want to find the title of book in which *every* para contains “sailing” then change the word some to every as follows:

```
FOR $b IN //book
WHERE EVERY $p IN $b//para SATISFIES
   contains($p, "sailing")
RETURN $b/title
```

Group-By in Xquery ??

```
FOR $b IN document("http://www.bn.com")/bib/book,
        $y IN $b/@year
WHERE $b/publisher="Morgan Kaufmann"
RETURN    GROUPBY $y
          WHERE count($b) > 10
```

← with GROUPBY

Equivalent SQL →

```
SELECT year
FROM Bib
WHERE Bib.publisher="Morgan Kaufmann"
GROUPBY year
HAVING count(*) > 10
```

Query

- Example: Return a flat list of supplier names and their part descriptions for the parts that are actually supplied, in alphabetic order.

JOINS in Relation

P (part)

pno	descrip	qnty
1	ABC	100
2	DEF	75
3	GHI	36
4	JKL	2
5	MNO	0

SP (Supplies)

pno	sno	price
2	24	5.00
3	35	6.50
2	14	4.00
4	24	10.00
1	27	2.25

S (supplier)

sno	name	locat
27	IBM	NY
35	MSFT	WSH
8	LSN	JAX
14	AMD	CA
51	AJR	BNA
24	UF	GNV

XML documents

P.XML

```
<parts>
   <p_tuple>
      <p_no>
          1
      </p_no>
      <descrip>
          ABC
      </descrip>
      <qty>
          100
      </qty>
   </p_tuple>
</parts>
```

S.XML

```
<supplier>
   <s_tuple>
      <s_no>
          27
      </s_no>
      <name>
          IBM
      </name>
      <locat>
          NY
      </locat>
   </s_tuple>
</supplier>
```

SP.XML

```
<supplies_part>
   <sp_tuple>
      <p_no>
          2
      </p_no>
      <s_no>
          24
      </s_no>
      <price>
          5.00
      </price>
   </p_tuple>
</supplies_part>
```

JOINS in XQuery

Query: Return list of supplier names and their part descriptions for the parts that are actually supplied, in alphabetic order

```
For $sp in document(“sp.xml”)//sp_tuple,
     $p  in document(“p.xml”)//p_tuple[
                         pno = $sp/pno]
    $s in document(“s.xml”)//s_tuple
                         [sno = $sp/sno]
Return <sp_pair> {
        $s/name, $p/descrip }
        </sp_pair> sortby(sname, descrip)
```

- Example:

- <bib>
- <book year="1994"> <title>TCP/IP Illustrated</title> <author><last>Stevens</last><first>W.</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>

- <book year="1999">
- <title>The Economics of Technology and Content for Digital TV</title> <editor> <last>Gerbarg</last><first>Darcy</first> <affiliation>CITI</affiliation> </editor>
- <publisher>Kluwer Academic Publishers</publisher> <price>129.95</price>
- </book>
- </bib>

- List books published by Addison-Wesley after 1991, including their year and title.

```
<bib> {
for $b in
   document("http://www.bn.com")/bib/book
   where $b/publisher = "Addison-Wesley" and
   $b/@year > 1991
return <book year="{ $b/@year }">
{ $b/title }
</book>
}
</bib>
```

Expected Result

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

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

```
<results> {
for $b in
   document("http://www.bn.com")/bib/book, $t
   in $b/title,
$a in $b/author
return <result>
{ $t }
{ $a } </result>
}
</results>
```

Expected Results

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

- For each book found at both bn.com and amazon.com, list the title of the book and its price from each source.

```
<books-with-prices>
{ for $b in document("www.bn.com/bib.xml")//book,
$a in
   document("www.amazon.com/reviews.xml")//entry
where $b/title = $a/title
return <book-with-prices>
{ $b/title }
<price-amazon>{ $a/price/text() }</price-amazon>
   <price-bn>{ $b/price/text() }</price-bn>
</book-with-prices> }
</books-with-prices>
```

- 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 $b in document("www.bn.com/bib.xml")//book
   where count($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>
```

```
<book>
<title>Data on the Web</title>
<author> <last>Abiteboul</last>
   <first>Serge</first>
</author>
<author> <last>Buneman</last>
   <first>Peter</first>
</author>
<et-al/>
</book>
```

- List the titles and years of all books published by Addison-Wesley after 1991, in alphabetic order.
-
```
<bib>
{ for $b in
   document("www.bn.com/bib.xml")//book
   where $b/publisher = "Addison-Wesley" and
   $b/@year > 1991
return <book>
{ $b/@year }
{ $b/title }
</book>
sort by (title) }
 </bib>
```

Element Constructors

- To generate a new element is to embed the element directly in a query using XML notation.

(Q) Generate an <emp> element that has an "empid" attribute and nested <name> and <job> elements.

```
<emp empid = "12345">
        <name>John Smith</name>
        <job>Anthropologist</job>
</emp>
```

Element Constructors

(Q) Generate an <emp> element that has an "empid" attribute. The value of the attribute and the content of the element are specified by variables that are bound in other parts of the query.

```
<emp empid = {$id}>
        {$name}
        {$job}
</emp>
```

FLWR Expressions

- *(Q) List each publisher and the average price of its books.*

```
FOR $p IN distinct(document("bib.xml")//publisher)
  LET $a := avg(document("bib.xml")//book[publisher =
  $p]/price)
  RETURN
     <publisher>
       <name> {$p/text()} </name>
       <avgprice> {$a} </avgprice>
     </publisher>
```

Sorting

- A sequence can be ordered by means of a SORTBY clause that contains one or more "ordering expressions."

(Q)List all books with price greater than $100, in order by first author; within each group of books with the same first author, list the books in order by title.

```
document("bib.xml")//book[price > 100] SORTBY
(author[1], title)
```

Conditional Expressions

(Q) Make a list of holdings, ordered by title. For journals, include the editor, and for all other holdings, include the author.

```
FOR $h IN //holding
RETURN
   <holding>
     {$h/title,
               IF ($h/@type = "Journal")
               THEN $h/editor
               ELSE $h/author
     }
     </holding>
SORTBY (title)
```

More Examples

- http://www-106.ibm.com/developerworks/xml/library/x-xquery.html