XQuery

Objectives

- To learn XML query language

Lecture outline

- What is XQuery?
- Functions, path expressions, and predicates
- XQuery Basic Syntax Rules
- XQuery Comparisons
- XQuery FLWOR Expressions
- FLWOR Explained Indetail
- XQuery built-in functions
- XQuery user-defined functions

- What is XQuery?

- XQuery is ***the*** language for querying XML data
- XQuery for XML is like SQL for databases
- XQuery is built on XPath expressions
- XQuery is defined by the W3C
- XQuery is supported by all the major database engines (IBM, Oracle, Microsoft, etc.)
- XQuery will become a W3C standard - and developers can be sure that the code will work among different products
- XQuery 1.0 and XPath 2.0 share the same data model and support the same functions and operators.

- Example document

```
<?xml version="1.0" encoding="ISO-8859-1"?>
    <bookstore>
        <book category="COOKING">
            <title lang="en">Everyday Italian</title>
            <author>Giada De Laurentiis</author>
            <year>2005</year>
            <price>30.00</price>
        </book>
        <book category="CHILDREN">
            <title lang="en">Harry Potter</title>
            <author>J K. Rowling</author>
            <year>2005</year>
            <price>29.99</price>
        </book>
        <book category="WEB">
             <title lang="en">XQuery Kick Start</title>
             <author>James McGovern</author>
            <author>Per Bothner</author>
             <author>Kurt Cagle</author>
            <author>James Linn</author>
            <author>Vaidyanathan Nagarajan</author>
            <year>2003</year>
            <price>49.99</price>
        </book>
        <book category="WEB">
            <title lang="en">Learning XML</title>
            <author>Erik T. Ray</author>
             <year>2003</year>
            <price>39.95</price>
        </book>
    </bookstore>
```

- Functions, path expressions, and predicates ...

- Functions:
  - XQuery uses functions to extract data from XML documents.
  - The doc() function is used to open the "books.xml" file:
    - doc("books.xml")
- Path expressions:
  - XQuery uses path expressions to navigate through elements in an XML document
  - The following path expression is used to select all the title elements in the "books.xml" file:
    - doc("books.xml")/bookstore/book/title
      - Output:

```
<title lang="en">Everyday Italian</title>
<title lang="en">Harry Potter</title>
<title lang="en">XQuery Kick Start</title>
<title lang="en">Learning XML</title>
```

... - Functions, path expressions, and predicates

- Predicates
  - XQuery uses predicates to limit the extracted data from XML documents
  - The following predicate is used to select all the book elements under the bookstore element that have a price element with a value that is less than 30:
    - doc("books.xml")/bookstore/book[price<30]
    - Output

```
<book category="CHILDREN">
        <title lang="en">Harry Potter</title>
        <author>J K. Rowling</author>
        <year>2005</year>
        <price>29.99</price>
</book>
```

- 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 :)

- XQuery Comparisons

- In XQuery there are two ways of comparing values.
  - General comparisons: =, !=, <, <=, >, >=
  - Value comparisons: eq, ne, lt, le, gt, ge
- The difference is shown below:
  - `$bookstore//book/@q > 10`
    - The expression above returns true if any q attributes have values greater than 10.
  - `$bookstore//book/@q gt 10`
    - The expression above 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.

- XQuery FLWOR Expressions

- The syntax of Flower expression looks like the combination of SQL and path expression
- The following path expression will select all the title elements under the book elements that is under the bookstore element that have a price element with a value that is higher than 30.

```
doc("books.xml")/bookstore/book[price>30]/title
```

- The following FLWOR expression will select exactly the same as the path expression above

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

  - Output

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

-- FLWOR briefly explained

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

- 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.
  - The **where** clause selects only book elements with a price element with a value greater than 30.
  - The **order by** sorts the results according to the specified element
  - The **return** clause specifies what should be returned. Here it returns the title elements

-- XQuery Conditional Expressions

- "If-Then-Else" expressions are allowed in XQuery.

```
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>
```

- **Notes on the "if-then-else" syntax:** parentheses around the if expression are required. else is required, but it can be just else ().

- Output

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

-- Adding Elements and Attributes to the Result

```
for $x in doc("books.xml")/bookstore/book/title
order by $x
return $x
```

- The XQuery expression above will include both the title element and the lang attribute in the result, like this
- <u>Output</u>

```
<title lang="en">Everyday Italian</title>
<title lang="en">Harry Potter</title>
<title lang="en">Learning XML</title>
<title lang="en">XQuery Kick Start</title>
```

-- Add Attributes to HTML Elements ...

```
<html>
<body>
<h1>Bookstore</h1>
<ul>
{
    for $x in doc("books.xml")/bookstore/book
    order by $x/title
    return  <li class="{data($x/@category)}">{data($x/title)}</li>
}
</ul>
</body>
</html>
```

... -- Add Attributes to HTML Elements

<u>Output</u>

```
<html>

<body>

<h1>Bookstore</h1>

<ul>
        <li class="COOKING">Everyday Italian</li>
        <li class="CHILDREN">Harry Potter</li>
        <li class="WEB">Learning XML</li>
        <li class="WEB">XQuery Kick Start</li>
</ul>

</body>

</html>
```

-- FLWOR Explained Indetail

- **for** - (optional) binds a variable to each item returned by the in expression
- **let** - (optional)
- **where** - (optional) specifies a criteria
- **order by** - (optional) specifies the sort-order of the result
- **return** - specifies what to return in the result

--- The for clause ...

- The for clause binds a variable to each item returned by the in expression. The for clause results in iteration. There can be multiple for clauses in the same FLWOR expression
- To loop a specific number of times in a for clause, you may use the **to** keyword
- Example:

```
for $x in (1 to 5)
return <test>{$x}</test>
```

- Result

```
<test>1</test>
<test>2</test>
<test>3</test>
<test>4</test>
<test>5</test>
```

… --- The for clause

- It is also allowed with more than one in expression in the for clause. Use comma to separate each in expression:

- Example:

```
for $x in (10,20), $y in (100,200)
return <test>x={$x} and y={$y}</test>
```

- Results

```
<test>x=10 and y=100</test>
<test>x=10 and y=200</test>
<test>x=20 and y=100</test>
<test>x=20 and y=200</test>
```

--- The **at** keyword

- The **at** keyword can be used to count the iteration:

```
for $x at $i in doc("books.xml")/bookstore/book/title
return <book>{$i}. {data($x)}</book
```

- Result

```
<book>1. Everyday Italian</book>
 <book>2. Harry Potter</book>
<book>3. XQuery Kick Start</book>
<book>4. Learning XML</book>
```

--- The let Clause

- The let clause allows variable assignments and it avoids repeating the same expression many times. The let clause does not result in iteration

- Example:

```
let $x := (1 to 5)
return <test>{$x}</test>
```

- Result:

```
<test>1 2 3 4 5</test>
```

--- The where Clause

- The where clause is used to specify one or more criteria for the result:

- Example:

```
where $x/price>30 and $x/price<100
```

--- The order by Clause

- The order by clause is used to specify the sort order of the result. Here we want to order the result by category and title:

- Example:

```
for $x in doc("books.xml")/bookstore/book
order by $x/@category, $x/title
return $x/title
```

- Result

```
<title lang="en">Harry Potter</title>
<title lang="en">Everyday Italian</title>
<title lang="en">Learning XML</title>
<title lang="en">XQuery Kick Start</title>
```

--- The return Clause

- The return clause specifies what is to be returned.

- Example:

```
for $x in doc("books.xml")/bookstore/book
return $x/title
```

- Result:

```
<title lang="en">Everyday Italian</title>
<title lang="en">Harry Potter</title>
<title lang="en">XQuery Kick Start</title>
<title lang="en">Learning XML</title>
```

- XQuery Built-in Functions

- XQuery includes over 100 built-in functions. There are functions for string values, numeric values, date and time comparison, node and QName manipulation, sequence manipulation, Boolean values, and more. You can also define your own functions in XQuery.
- A call to a function can appear where an expression may appear.
- Example 1: in an element:

```
<name>{uppercase($booktitle)}</name>
```

- Example 2: In the predicate of a path expression

```
doc("books.xml")/bookstore/book[substring(title,1,5='Harry')]
```

- Example 3: In a let clause

```
let $name := (substring($booktitle,1,4))
```

- XQuery User-Defined Functions ...

```
declare function prefix:function_name($parameter AS datatype)
AS  returnDatatype
{
    (: ... function code here... :)
};
```

- Use the declare function keyword
- The name of the function must be prefixed
- The data type of the parameters are mostly the same as the data types defined in XML Schema
- The body of the function must be surrounded by curly braces

... - XQuery User-Defined Functions ...

```
declare function local:minPrice( $price as xs:decimal?,
                                  $discount as xs:decimal?)
AS xs:decimal?
{
    let $disc := ($price * $discount) div 100
    return ($price - $disc)
};
```

- (: Below is an example of how to call the function above :)

```
<minPrice>{local:minPrice($book/price,$book/discount)}</minPrice>
```

- References

- W3School Xquery Tutorial
  - http://www.w3schools.com/xquery/default.asp
- MSXML 4.0 SDK
- Several online presentations

- Reading list

- W3School Xquery Tutorial
  - http://www.w3schools.com/xquery/default.asp
- The following paper which is posted in the WebCT.
  - D. Chamberlin, *XQuery: An XML query language*, IBM Systems Journal, Vol 41, No 4, 2002.

END