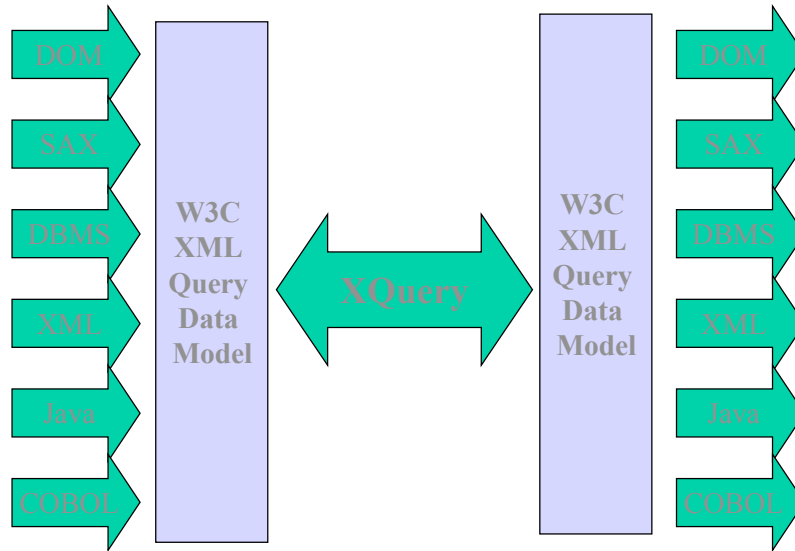


# XQuery – The W3C XML Query Language

Jonathan Robie, Software AG  
Don Chamberlin, IBM Research  
Daniela Florescu, INRIA  
(with some changes and adaptations by Ralf Möller, TUHH)

- Ⓔ Expressive power
  - Major functionality of XML-QL, XQL, SQL, OQL - query the many kinds of data XML contains!
  - Use-case driven approach
- Ⓔ Can be implemented in many environments
  - Traditional databases, XML repositories, XML programming libraries, etc.
  - Queries may combine data from many sources
- Ⓔ Minimalism and good design
  - Small, easy to understand, clean semantics
  - “A quilt, not a camel”

2



3

## The XQuery Language

- ⊞ XQuery is a functional language
  - A query is an expression
  - Expressions can be combined flexibly
- ⊞ Structure of a query
  - Namespace declarations (optional)
  - Function definitions (optional)
  - The query expression – often composed of many expressions

5

- ⊞ Path expressions: `/a//b[c = 5]`
- ⊞ FLWR expressions: `FOR ... LET ... WHERE ... RETURN`
- ⊞ Element constructors: `<a> ... </a>`
- ⊞ Variables and constants: `$x, 5`
- ⊞ Operators and function calls: `x + y, -z, foo(x, y)`
- ⊞ Conditional expressions: `IF ... THEN ... ELSE`
- ⊞ Quantifiers: `EVERY var IN expr SATISFIES expr`
- ⊞ Sorted expressions: `ORDER BY (expr ASCENDING , ... )`
- ⊞ Preliminary proposal for `INSERT, REPLACE, DELETE`

6

```
<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>
```

7

# Element constructors look like the XML they construct

```
<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>
```

8

```

<bib>
  <book year="1994">
    <title>TCP/IP Illustrate
    <author>
      <last>Stevens</last>
      <first>W.</first>
    </author>
    <publisher>Addison-V
    <price> 65.95</price>
  </book>

```

# XQuery uses the abbreviated syntax  
# of XPath for path expressions

document("bib.xml")

/bib/book/author

//author[last="Stevens" and first="W."]

document("bib.xml")//author

9

```

# Range expressions
/bib/book/author[1 TO 2]

# BEFORE and AFTER
//book[ author[last="Stevens"] BEFORE author[last="Abiteboul"] ]

# Namespaces
NAMESPACE rev = "www.reviews.com"
//rev:rating

# Attributes
//publisher/@name

```

10

- FOR - LET - WHERE - ORDER BY - RETURN
- Similar to SQL's SELECT - FROM - WHERE

```
FOR $book IN document("bib.xml")//book
WHERE $book/publisher = "Addison-Wesley"
RETURN
  <book>
    {
      $book/title,
      $book/author
    }
  </book>
```

11

- FOR iterates on a sequence, binds a variable to each node
- LET binds a variable to a sequence as a whole

```
FOR $book IN document("bib.xml")//book
LET $a := $book/author
WHERE contains($book/publisher, "Addison-Wesley")
RETURN
  <book>
    {
      $book/title,
      <count> Number of authors: { count($a) } </count>
    }
  </book>
```

12

```
FOR $book IN document("www.bib.com/bib.xml")//book,  
    $quote IN document("www.bookstore.com/quotes.xml")//listing  
WHERE $book/isbn = $quote/isbn  
ORDER BY (title)  
RETURN  
  <book>  
    { $book/title }  
    { $quote/price }  
  </book>
```

13

```
FOR $book IN document("bib.xml")//book  
ORDER BY (title)  
RETURN  
  <book>  
    { $book/title }  
    {  
      FOR $review IN document("reviews.xml")//review  
      WHERE $book/isbn = $review/isbn  
      RETURN $review/rating  
    }  
  </book>
```

14

```
<bibliography>
{
  FOR $book IN document("bib.xml")//book
  ORDER BY (author, title)
  RETURN
    <book>
      {
        $book/author,
        $book/title
      }
    </book>
}
</bibliography>
```

15

```
<bibliography>
  Expression
</bibliography>
```

16

```
<bibliography>
{
  FOR $book IN Expression
  RETURN
    Expression
}
</bibliography>
```

17

```
<bibliography>
{
  FOR $book IN Expression
  ORDER BY (Expression, Expression, ...)
  RETURN
    <book>
      {
        Expression,
        Expression
      }
    </book>
}
</bibliography>
```

18

```
<bibliography>
{
  FOR $book IN document("bib.xml")//book
  ORDER BY (author, title)
  RETURN
    <book>
      {
        $book/author,
        $book/title
      }
    </book>
}
</bibliography>
```

19

### ☒ Built-in functions

- max(), min(), sum(), count(), avg()
- distinct(), empty(), contains()
- the normative set has not yet been fixed

### ☒ User-defined functions

- Defined in XQuery syntax
- May be recursive
- May be typed

### ☒ Extensibility mechanisms planned

20

```
FUNCTION depth(ELEMENT $e) RETURNS integer
{
  -- An empty element has depth 1
  -- Otherwise, add 1 to max depth of children
  IF empty($e/*)
    THEN 1
    ELSE max(depth($e/*)) + 1
}

depth(document("partlist.xml"))
```

21

### W3C XML Schema simple types

- string "Hello"
- boolean true, false
- integer 47, -369
- float -2.57, 3.805E-2

### Type constructor functions

- date("2000-06-25")

### Operators and functions to be defined...

22

## Data Transformations

```
<?xml version="1.0"?>
<bib>
  <book>
    <title> Harold and the Purple Crayon </title>
    <author>
      <lastname> Johnson </lastname>
      <firstname> Crockett </firstname>
    </author>
    <pubinfo>
      <publisher> Harper and Row </publisher>
      <price> 4.76 </price>
      <year> 1995 </year>
    </pubinfo>
  </book>
</bib>
```

```

<?xml v
<bib>
  <book
    <title
      <author
        <name>
          <last> Johnson </last>
          <first> Crockett </first>
        </name>
      </author>
    <pub
      <p
        <title> Harold and the Purple Crayon </title>
      <p
        <title> Harold and the Circus </title>
      <y
        <title> Harold's ABCs </title>
      </pul
        <title> Harold's Fairy Tale </title>
      </book
    </bib>
  . . .
</booksByAuthor>

```

25

```

<?xml version="1.0"?>
<bib>
  . FOR $a IN distinct(document("powerpoint/bib.xml")//book/author)
  LET $b :=
    FOR $c IN document("powerpoint/bib.xml")//book[author = $a]
    ORDER BY title
    RETURN { $c }
  ORDER BY(author/last, author/first)
  RETURN
    <book>
      { $a }
      { $b/title}
    </book>
</bib>

```

26

## Updates (preliminary)

### ⊞ INSERT

```
FOR $e IN /emp
  INSERT <n_skills> { count($e/skill) } </n_skills>
  BEFORE $e/skill[1]
```

### ⊞ REPLACE

```
FOR $e IN /emp
  WHERE $e/empno = "1234"
  REPLACE $e/job
  WITH <job> "Broom Tester" </job>
```

## ⓔ DELETE

```
FOR $e IN /emp/[job = "Programmer"],
  $s IN $e/skill
WHERE $s/rating < 4
  OR $s/cert_date < date("1995-01-01")
DELETE $s
```

29

- ⓔ No distributed update - single data source
- ⓔ No updates on views

30

## Summary

- ⊞ The W3C XML Query Language
- ⊞ Many DOM+XPath+XSLT applications can now be implemented in just one language
- ⊞ Expressive, concise, easy to learn
- ⊞ Implementable, optimizable
- ⊞ Data integration for multiple sources
- ⊞ Several current implementations
- ⊞ Preliminary update proposal

- Ⓜ W3C XQuery  
<http://www.w3.org/TR/xquery.html>
- Ⓜ W3C XML Query Use Cases  
<http://www.w3.org/TR/xmlquery-use-cases.html>
- Ⓜ W3C XML Query Requirements  
<http://www.w3.org/TR/xmlquery-req.html>
- Ⓜ W3C XML Query Data Model  
<http://www.w3.org/TR/query-datamodel.html>
- Ⓜ W3C XML Query Algebra  
<http://www.w3.org/TR/query-algebra.html>