

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

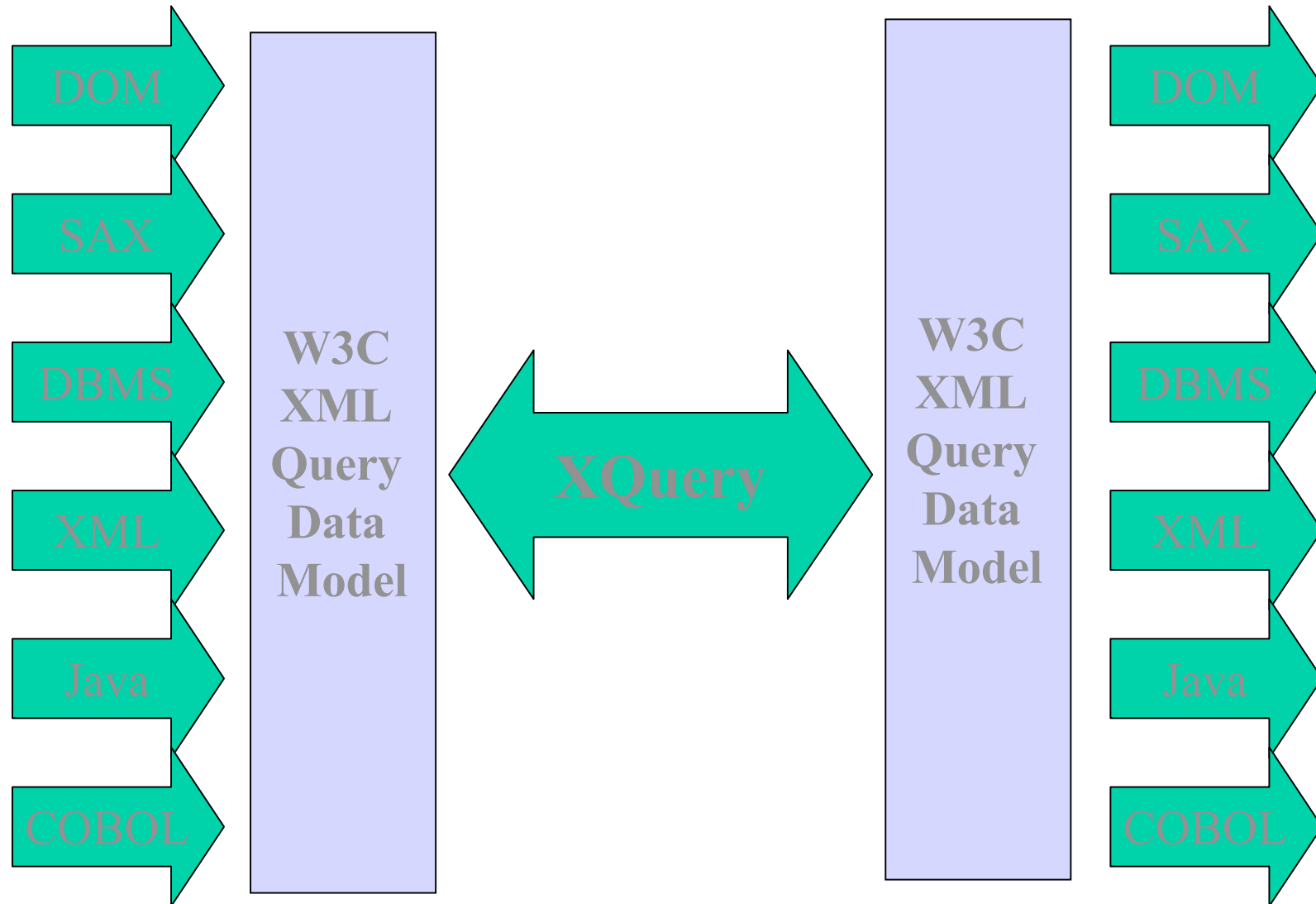
- Major functionality of XML-QL, XQL, SQL, OQL
- query the many kinds of data XML contains!
- Use-case driven approach

⌘ 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”



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

- ⓔ Path expressions: `/a//b[c = 5]`
- ⓔ FLWR expressions: `FOR ... LET ... WHERE ... RETURN`
- ⓔ Element constructors: `<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`

```
<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>
```

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>
```

```
<bib>
```

```
<book year="1994">
```

```
<title>TCP/IP Illustrate
```

```
<author>
```

```
<last>Stevens</last>
```

```
<first>W.</first>
```

```
</author>
```

```
<publisher>Addison-W
```

```
<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
```

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

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

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

- ⊞ 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>
```

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

Comparison Operators

- ⊞ Comparison of base values: eq
- ⊞ Comparison of structured values: equal (deep equal)
- ⊞ Comparison of sequences: =
 - Example:

```
FOR ...  
LET $book1 = ...,  
    $book2 = ...  
WHERE $book1/title = $book2/title  
RETURN ...
```

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

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

<bibliography>

Expression

</bibliography>

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

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

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

☞ 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

```
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"))
```

☞ 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...

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 versio
<bib>
  <book>
    <title> Har
    <author>
      <lastna
      <firstna
    </author>
    <pubinfo>
      <publis
      <price>
      <year>
    </pubinfo:
  </book>
</bib>
```

```
<booksByAuthor>
  <author>
    <name>
      <last> Johnson </last>
      <first> Crockett </first>
    </name>
  </author>
  <title> Harold and the Purple Crayon </title>
  <title> Harold and the Circus </title>
  <title> Harold's ABCs </title>
  <title> Harold's Fairy Tale </title>
  . . .
</booksByAuthor>
```

```
<?xml version="1.0"?>
```

```
<bib>
```

```
  <booksByAuthor>
```

```
  {
```

```
    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
```

```
      { $a }
```

```
      { $b/title }
```

```
</
```

```
  }
```

```
</booksByAuthor>
```

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
```

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

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>