XQuery: An XML Query Language
D. Chamberlin

Pre-Discussion

After the presentation, we will evaluate XQuery. During the presentation, think about consequences of the design decisions on the usability of the language.

1. The story, in brief is...

• XQuery : XML :: SQL : relational tables
• Intended as a 'standard' way to query 'different' XML data sources
• Declarative, Uses expressions to assemble queries and results
• A number of factors influenced XQuery

Outline

1. The story, in brief is...
2. XQuery's design is influenced by:
   1. XML data
   2. Pre-existing standards and languages
   3. The vision behind XQuery, it's intended use
3. XQuery, the language
4. Current affairs

XML vs. Relational Data

• Relational Data
  – Flat Structure
  – Optimized for efficient access and retrieval
• XML
  – Hierarchical Structure
  – Optimized for representing intrinsic relationships of data that make up an XML document
• Implications for query language design
  • Path Expressions
  • FLWOR Expressions
  • Case sensitive,

Other query languages

• Constrained by existing standards (XML Schema, XPath)
• Influenced by other languages
  – XSLT, Quilt etc.
• Each of these were good for a niche area
• XPath
  – Cannot create new XML, introduce variables or namespaces, select part of nodes etc.
• XSLT (styling XML for display formats)
  – Joins, querying etc...can be difficult to do (debatable!)
Other query languages

- XQuery
  - Joins and Sorts
  - Manipulating Sequences of Values
  - User defined functions easy to write (even with recursion)
  - Create temporary results and navigate them
  - Restructure using FLWOR expressions

The vision

- Data Integration
  - Built by people from the db and SQL community
  - XML Schema included (allows validation)
- Designed to support query processing
  - Procedural query processing
    - User defined functions
    - FLWOR expressions
  - Element Constructors for temporary results

Outline

1. The story, in brief is…
2. XQuery’s design is influenced by:
   1. XML data
   2. Pre-existing languages
   3. The vision behind XQuery, it’s intended use
3. XQuery, the language
   1. Data Model
   2. Query Processing Steps
   3. Language Features

Data Model

- Abstract representation for XML docs
- Based on the notion of sequences of nodes or atomic values
- Nodes form hierarchies to represent relationships implicitly
- Document order is important

items.xml

```xml
<item status = "available">
  <itemno>0021</itemno>
  <description>Scooter</description>
  <seller>Vespa</seller>
  <reserve-price>12000</reserve-price>
</item>
```
2. Query Processing Steps

XML DOC → Schema validation → XML Representation → Query → Query Results

3. Language Features

- **Path Expressions**
- **Predicates**
- **FLWOR**
- **Functions**
- **Type System**

Language Features

- **Path Expressions** to navigate through elements in an XML document
  - Based on XPath syntax
  - Series of steps separated by ‘/’
  - Each step returns a sequence of nodes
  - The last sequence is the value of the Expression
  - `doc("items.xml")/child::*/child::item[child::seller = "Vespa"]/child::description`

Language Features

- **Predicates** to filter a sequence of values
  - Often used in the steps of a path expression
  - `item[seller = "Vespa"]`
  - Evaluated on each item of the sequence
  - `seller = "Vespa"` used to select some item nodes and discard others

FLWOR

- Iteration over sequences of values
- **FLWOR**
  - for, let, when, order-by, return
  - Adopted from Quilt
  - For each item that has more than ten bids, generate a popular-item element containing item-description (pg: 606)

FLWR

```xml
for $i in doc("items.xml")/*/item
let $b := doc("bids.xml") /*/bid[itemno = $i/itemno]
where count ($b) > 10
return
<popular-item>
  { $i/description }
</popular-item>
```
Functions

- Functions
  - Libraries as well as User defined
  - Recursion supported
  ```
  define function depth( element $e) 
  returns integer 
  {
      ...
  }
  
  depth(doc("bids.xml"))
  ```

Type System

- Based on the Type System of XML Schema
- Queries need to refer to types sometimes
  - Can use qualified name, generic keywords
- Occurrence indicators
  - Element of type order+
- Dynamic type information: typeswitch
- Typecasting available: cast, treat and assert

Outline

1. The story, in brief is...
2. XQuery’s design is influenced by:
   1. XML data
   2. Pre-existing languages
   3. The vision behind XQuery, it’s intended use
3. XQuery, the language
4. Current affairs

Current affairs

- Supported by all major vendors
  - IBM, Oracle, Microsoft etc
  - XQuery code should work across vendors
- Compatible with several W3C standards
- XQuery 1.0 is still a working draft

Discussion: Evaluating XQuery

Full-class discussion: How would you evaluate XQuery?
- As a query language, how does it compare to SQL?
- What are your general feelings about usability?
- When would you use it?
- When would you prefer something else like XML-QL or XSLT (or even mapping XML to relational and using SQL)?

For and Return

- for and return
  ```
  for $m in (2,3), $n in (5,10)
  return <fact> {m} times {n} is {$m * $n} </fact>
  ```

  ```
  Element constructor
  ```
For and Return

• for and return

```plaintext
for $m$ in (2, 3), $n$ in (5, 10)
return <fact> ($m$) times ($n$) is
{$m * n}$ </fact>
```

RESULT
```plaintext
<fact> 2 times 5 is 10 </fact>
<fact> 2 times 10 is 20 </fact>
<fact> 3 times 5 is 15 </fact>
<fact> 3 times 10 is 30 </fact>
```

Let

• Let

```plaintext
for $i$ in (1 to 3)
let $j := (1 to $i$)

$i = 1, \; j = 1$
$i = 2, \; j = (1, 2)$
$i = 3, \; j = (1, 2, 3)$
```