XQuery

A typed language for XML

Jérôme Siméon

Bell Laboratories
XML is data on the Web

- Used as an exchange format between applications
  - XML & e-commerce: cXML, mpXML, etc.
  - XML & biology: BIOXML, GeneXML, etc.
  - XML & the news: NML
- Stored in back-end systems
  - Tamino, SQL/Server, etc.
- Published to user’s interfaces
  - XHTML, VoiceXML, etc.
Example of a cXML document

<OrderRequestHeader orderID="D01234" orderDate="1999-03-12"
   requestedDeliveryDate="1999-03-24">
  <Total><Money currency="USD">12.34</Money></Total>
  <ShipTo>
    <Address>
      <Name xml:lang="en">Lucent Technologies</Name>
      <PostalAddress name="foo">
        <DeliverTo>Joe Smith</DeliverTo>
        <DeliverTo>Mailstop M-543</DeliverTo>
        <Street>123 Anystreet</Street>
        <City>Sunnyvale</City>
        <State>CA</State>
      </PostalAddress>
    </Address>
  </ShipTo>
  ...

Building XML applications

- Using XML Interfaces
  - Document Object Model (DOM)
  - Simple API for XML (SAX)
  - Java API for XML Parsing (JAXP)

- Using Programming languages
  - Java
  - C++
  - Perl

- Using dedicated XML languages
  - XPath for path expressions
  - XSLT for structural transformations
Example of a DOM Nodefilter in Java

Class NamedAnchorFilter implements NodeFilter
{
    short acceptNode(Node n) {
        if (n.getNodeType()==Node.ELEMENT_NODE) {
            Element e = (Element)n;
            if (! e.getNodeName().equals("A"))
                return FILTER_SKIP;
            if (e.getAttributeNode("NAME") != null)
                return FILTER_ACCEPT;
        }
        return FILTER_SKIP;
    }
    return FILTER_SKIP;
}
What is XQuery?

- A query language for XML
  - Aimed at supporting data-intensive applications
- A W3C activity (XML Query Working Group)
  - Industry investment
  - Standardization effort
  - Coordination with XML Schema, XPath, XSLT, etc.
- A preliminary proposal
  - First working draft last February
  - Incomplete implementations (XML DevCon’2001)
  - Subject to changes
  - Receptive to feedbacks
- Hype, but also some interesting technical contributions
Example of a query in XQuery

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>
Why XQuery?

- More logical/physical independance
  - Physical XML documents (e.g., in a file)
  - Virtual XML documents (e.g., view over a RDBMS)
  - Streaming documents (e.g., stock quotes)
- More declarativity
  - Concise syntax for complex data manipulations
  - Subject to optimization
- More types
  - Data values (int, float, string, date, etc.)
  - Detecting errors
    - e.g., a book does not have a titel but a title
  - Infering schemas
    - e.g., my program does generate VoiceXML
Outline

- Status of XQuery
- XML Query Data Model
- XQuery features
- XQuery typing with the XML Algebra
- XQuery architecture and Galax Prototype
W3C XML Query Working Group

http://www.w3.org/XML/Query/

- Composed of 32 companies
- Work started 18 months ago
- Chartered to produce a *data model*, an *algebra*, and *two syntaxes*
  - User-level syntax
  - XML syntax
- Collaboration with other groups
  - XML Schema, XSLT
  - XML Core (Information Set), I18N, SQL (ISO)
Current documents

- Requirements for XQuery
- Use cases (9)
  - Querying relational databases
  - Querying hierarchy and order in documents
  - Querying text in XML documents
  - Querying recursive documents
  - etc.
- Specifications
  - XML Query Data Model (June 2000)
  - XML Query Algebra (Dec 2000)
  - XML Query Syntax (XQuery - Feb 2001)
XML Query Data Model in a Nutshell

- Different kinds of nodes (9)
  - Document, Element, Attribute, Value, Namespace, PI, Comment, etc.

- Accessors
  - `name(e)`, `nodes(e)`, etc.

- Constructors
  - `elemNode qname attributes children`

- Typed values (int, string, etc.)

- Each node has an identity and a (optional) parent
XQuery Design

- functional
  - no side-effect
- compositional
  - made of basic expressions
  - expressions can be composed arbitrarily
- allows full recursion
  - user defined functions without restriction
- "declarative"
  - high-level FLWR block
- answers use cases
  - expressive
- formal semantics
  - by means of the XML Query algebra
XQuery expressions

- Constants
- Variables
- XPath expressions
- operators and functions
- FLWR (FOR LET WHERE RETURN) expressions
- SORT BY
- conditional
- ∃ ∀
- Aggregation
- Element/Attribute constructors
Values and variables

► Atomic values

1
"John Smith"
true

► Variables

$a   $mybook

► Elements and attributes:

<book>
   231
</book>
Operators and functions

- Arithmetic operations
  \[ 1+3 \]

- Comparison operations
  \[(1+3) > 2\]

- Boolean operations
  \[\text{not}(((1+3) > 2) \text{ or } ((a > 2) \text{ and } (a < 6)))\]

- Collection operations (Union, Except, Intersect)
- Data model functions (name, value, etc)
List the titles of books published by Morgan Kaufmann in 1998.

document("bib.xml")//book[publisher = "Morgan Kaufmann"
    AND $year = "1998" ]/title
List the titles of books published by Morgan Kaufmann in 1998.

FOR $b$ IN document("bib.xml")//book
WHERE $b/publisher = "Morgan Kaufmann"
AND $b/year = "1998"
RETURN $b/title
Joins in XQuery

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>
Sorting in XQuery

List the titles and years of all books published by Addison-Wesley after 1991, in alphabetic order.

```xquery
<bib>
  FOR $b IN document("www.bn.com/bib.xml")//book
      [publisher = "Addison-Wesley" AND @year > "1991"]
  RETURN
      <book>
        $b/@year,
        $b/title
      </book> SORTBY (title)
</bib>
```
Quantification in XQuery

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
Global order in XQuery

In Report1, what happened between the first Incision and the second Incision?

FOR $proc IN
document("report1.xml")//section[section.title="Procedure"],
  $bet IN
  $proc//* AFTER ($proc//incision)[1]
  BEFORE ($proc//incision)[2]
RETURN $bet
Semantics of XQuery

- Semantics by mapping to the XML Query Algebra
  - "minimal" set of operations to support XQuery
  - Static and Dynamic semantics for this set
  - Nothing implicit!

- Dynamic semantics:
  - Given input values for documents and functions
  - What is the output document for each query expression

- Static semantics:
  - Given input types for documents and functions
  - Infer the type of the output for each query expressions
XML Algebra Core Operations


atomic constant
variable
attribute
element
sequence or union
empty sequence
conditional
iteration
local binding
function application
error
binary operator
unary operator
match

CaseRules ::= case Var : Type do Exp CaseRules
          | else Exp
Types

t ::= y  type variable
   | Integer | Float
   | String  | Boolean
   | ()      | empty sequence
   | 0       | empty choice
   | @Wild[t]| attribute
   | Wild[t] | element
   | t1 , t2 | sequence
   | t1 | t2 | choice
   | t {m, n}| repetition
Typing path expressions

type Bib =
    bib [ Book{0,*} ]
type Book =
    book [
        @year [ Integer ],
        @isbn [ String ],
        title [ String ],
        author [ String ]{1,*}
    ]

let bib0 : Bib = document("algebrabib.xml")

bib0/book/author
: author [ String ]{0,*}
More typing

► Constants

39.95  :  Float

"Lord of the Rings"  :  String

► Element creation

price [ 39.95 ]  :  price [ Float ]

► Sequences

Even more typing

- Let expressions
  
  ```
  let p = 39.95 in price [ p ]
  : price [ Float ]
  ```

- Typing conditionals
  
  ```
  let p = 39.95 do
  if p > 100
  then expensivebook[ p ]
  else cheapbook[ p ]
  : expensivebook [ Float ] | cheapbook [ Float ]
  ```
Query expansion

bib0/book

for v1 in bib0 do
  for v2 in nodes(v1) do
    match v2
      case v3 : book[AnyType] do v3
    else ()
  : Book{0,*}
Typing iteration

nodes(book0)

: @year[Integer], @isbn[String], title[String], (author[String]){1,*}

for v1 in nodes(book0) do
  match v1
    case v2 : title[AnyType] do v2
    else silly[]

  : silly[], silly[], title[String], (silly[])\{1,*\}

▶ Each components of the input forest is typed separately.
Galax Overview

- Implements W3C Working drafts:
  - XML Query Data Model (almost complete)
  - XML Query Algebra (almost complete)
  - XQuery (minimal)
- In its second round of implementation
  - Written in Objective Caml and C
  - Lightweight, performant, portable (now Linux & Solaris)
- Designed to support various storage modules
  - Native XML Query data model (now)
  - Module for DataBlitz (soon)
  - Module for native XML storage?
Once there was a query...
The query met a Schema
Going well, together they created a new Schema

XML Algebra
Expression
XML Algebra Parser
XML Algebra
Internal Structure
XML Schema
Description XML Schema Parser XML Schema
Internal Structure
Type Checker XML Schema
of Query Result
Static Error
for non
well-typed
queries!

XML Algebra
Expression XML Algebra Parser
XML Algebra
Internal Structure
XML Schema
Description XML Schema Parser XML Schema
Internal Structure
Type Checker XML Schema
of Query Result
Static Error
for non
well-typed
queries!

Bell Laboratories University of Tokyo, April 2001
The query quickly forgot about the schema...
...and feeling safe, created a document
In the real world, the query compiles
The query type checks

XQuery Expression → XQuery Parser → XQuery Internal Structure

XML Algebra Expression → XML Algebra Parser → XML Algebra Internal Structure

XML Schema Description → XML Schema Parser → XML Schema Internal Structure

XML Document → XML Parser → XML Data Model Internal Structure

XQuery Compiler

Type Checker

Query Processor

XML Schema of Query Result → Type Checker

Static Error for non well-typed queries!

XML Document Query Result
And finally the query evaluates
Typing Use Case

- Detect errors before running queries on 10Gb data
  - Float arithmetics over dates?
  - Mistakes in element names: result always empty
  - Query does not generate expected schema
    - E.g., Some Lucent products generating VoiceXML
- XQuery must work with and without a Schema
  - Keep relational schema information in use case R
  - Generating some HTML for any well-formed XML
What’s in the Demo

- Support for XML Algebra:
  - Static typing
  - Query evaluation

- Examples:
  - From XML Algebra document
  - Use cases: XMP and R (relational)

What’s not in the Demo

- Support for XQuery:
  - Compilation to the XML Algebra
Some Influential Related Work

- XML Query languages
  - XML-QL (ATT)
  - Lorel (Stanford)
  - YATL (INRIA)
  - Quilt (Software AG, IBM)

- Typed ‘query’ languages
  - OQL
  - Kleisli (UPenn)
  - XDuce (Upenn)
Conclusion

▶ XQuery is on tracks
  ▶ Concrete progress
  ▶ Concrete contributions
  ▶ On-going implementations
▶ Still very young
▶ Feedback is essential at this stage
▶ Stay tuned!
More about Galax

- Working towards full W3C compliance
- Complete set of use cases
- Optimization
- Physical storage
  - Datablitz (Lucent Main memory Database)
  - ODBC
  - Native XML storage

Contact: Jérôme Siméon
mailto:simeon@research.bell-labs.com
http://www-db.research.bell-labs.com/user/simeon/