XML Standards and Query Languages

Norbert Fuhr

University of Duisburg-Essen

Tutorial Structure

1. XML standards

   - plain XML
   - XML namespaces
   - DTDs and XML schema

2. XML Query Languages

   - Requirements
   - Development
   - XPath and XQuery
   - XML databases

Part I

XML Standards

Content

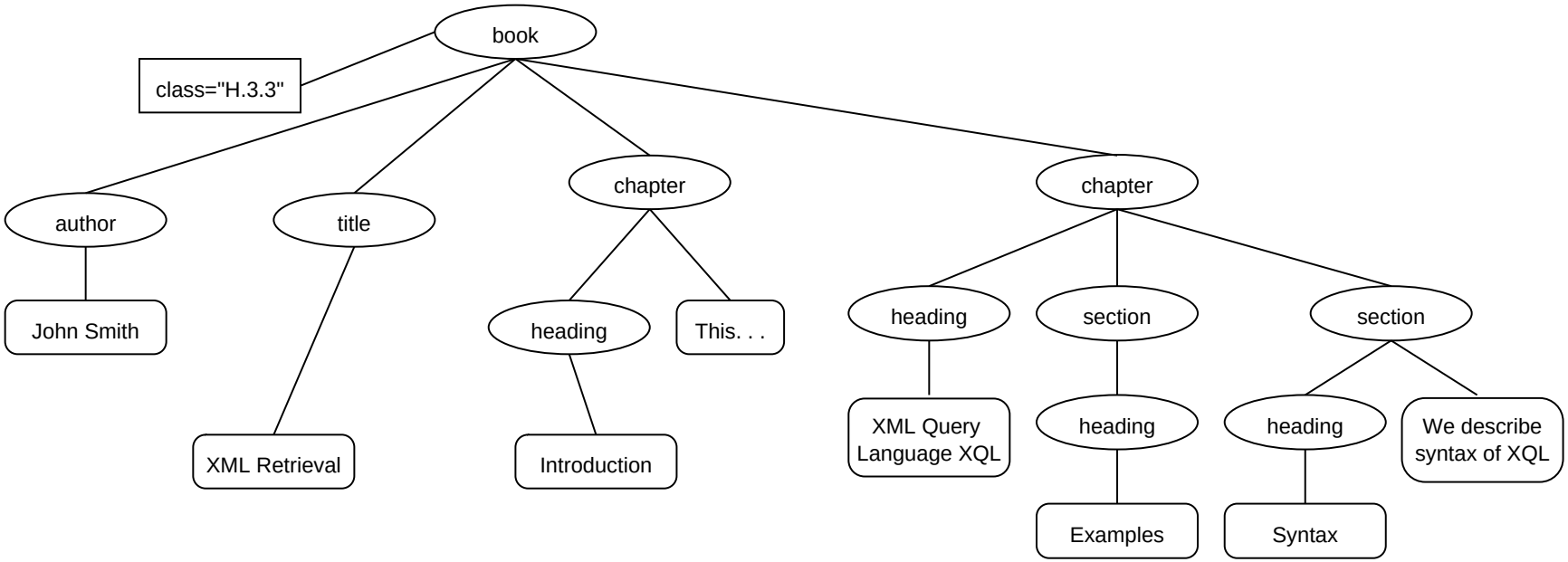
- Introduction
- XML namespaces
- Document Type Definitions (DTDs)
- XML Schema
- Other standards

(For details of the XML standards, see `http://www/w3c.org`)

Introduction: Example XML document

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE book SYSTEM "/services/dtds/book.dtd">
<book class="H.3.3">
  <author>John Smith</author>
  <title>XML Retrieval</title>
  <chapter>
    <heading>Introduction</heading>
    This text explains all about XML and IR.
  </chapter>
  <chapter>
    <heading> XML Query Language XQL </heading>
    <section>
        <heading>Examples</heading>
    </section>
    <section>
        <heading>Syntax</heading>
        Now we describe the XQL syntax.
    </section>
  </chapter>
</book>
```

Tree structure of XML documents



XML properties

- hierarchical structure: nesting of elements
- element: start-tag – content – end tag
  `<tag-name> content </tag-name>`
- tag-name: logical name of element
- content: data or other elements
  (nesting of elements)
  `<author><first>John</first>`
  `<last>Smith</last></author>`
- attributes: assigned to elements
  (specified in start tag)
  pair of (*attribute name*, *attribute value*),
  e.g. `<date format="ISO">2000-05-01</date>`

XML: Basic ideas

- markup of logical structure of documents
  $\rightsquigarrow$ explicit logical structure, can be exploited by appropriate IR methods

- separation of logical structure and layout
  $\rightsquigarrow$ different presentations of one document, depending on output media, user group (language,...)

- support interoperability of Web services and XML-based applications
  $\rightsquigarrow$ standard document format for IR systems

Basic XML standard does not deal with ...

- standardization of element names
  → XML namespaces

- structure of element content
  → XML DTDs

- data types of element content
  → XML schema

I.1 XML namespaces

allow for combination of element names defined independently (in different resources)

```
<?xml version="1.0"?>
 <bk:book xmlns:bk='urn:loc.gov:books'
          xmlns:isbn='urn:ISBN:0-395-36341-6'>
     <bk:title>Cheaper by the Dozen</bk:title>
     <isbn:number>1568491379</isbn:number>
 </bk:book>
```

Example: Dublin Core namespace

```
<oai_dc:dc  xmlns:dc="http://purl.org/dc/elements/1.1/">
  <dc:title>Generic Algebras with Involution of Degree 8m</dc:title>
  <dc:subject>Orthogonal group, Symplectic group,
              invariant field, rational
  </dc:subject>
  <dc:date>2001-02-27</dc:date>
  <dc:format>application/postscript</dc:format>
  <dc:identifier>ftp://ftp.esi.ac.at/pub/esi1001.ps</dc:identifier>
  <dc:source>ESI preprints</dc:source>
  <dc:language>en</dc:language>
</oai_dc:dc>
```

I.2 Document Type Definitions

- well-formed XML: proper nesting of elements
  (e.g. <a> is forbidden

- valid XML: document is well-formed and conforms to *document type definition*

Declaration of DTD

in the document header:

```
<!DOCTYPE name  PUBLIC publicid systemid>

<!DOCTYPE name SYSTEM filename>
```

Example HTML document with public DTD

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
  "http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<title>My Home Page</title>
</head>
<body>
<p>Hello! This is my home page.</p>
</body>
</html>
```

Example XML document with system DTD

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE book SYSTEM  "/services/dtds/book.dtd">
<book class="H.3.3">
  <author>John Smith</author>
  <title>XML Retrieval</title>
  <chapter>
    <heading>Introduction</heading>
    This text explains all about XML and IR.
  </chapter>
  <chapter>
    <heading>
      XML Query Language XQL
    </heading>
    <section>
        <heading>Examples</heading>
    </section>
    <section>
        <heading>Syntax</heading>
        Now we describe the XQL syntax.
    </section>
  </chapter>
</book>
```

DTD for example document

```
<!ELEMENT book (author, title, chapter+)>
<!ELEMENT author (#PCDATA)>
<!ELEMENT title (#PCDATA)>
<!ELEMENT chapter (heading,#PCDATA?,section*)>
<!ELEMENT section (heading,#PCDATA?)>
<!ELEMENT heading (#PCDATA)>
<!ATTLIST book
    class  CDATA #REQUIRED
    crdate CDATA #IMPLIED
    type   (monograph|collection|proceedings) "monograph">
```

DTD Specification

- element definitions
- definition of element attributes
- entity definitions (macros)

DTDs from an IR point of view

- restrict logical structure of documents
  $\rightsquigarrow$ IR methods can be tailored for document type

- element types have a well-defined meaning
  $\rightsquigarrow$ specialized search methods for content of specific elements possible
  e.g. person name, date, classification code

I.3 XML Schema

types of XML applications:

1. structured documents
   text documents with markup of logical structure
   $\rightsquigarrow$ document-centric

2. formatted data
   (e.g. spreadsheets, business forms, databases, . . .)
   XML as exchange format
   $\rightsquigarrow$ data-centric

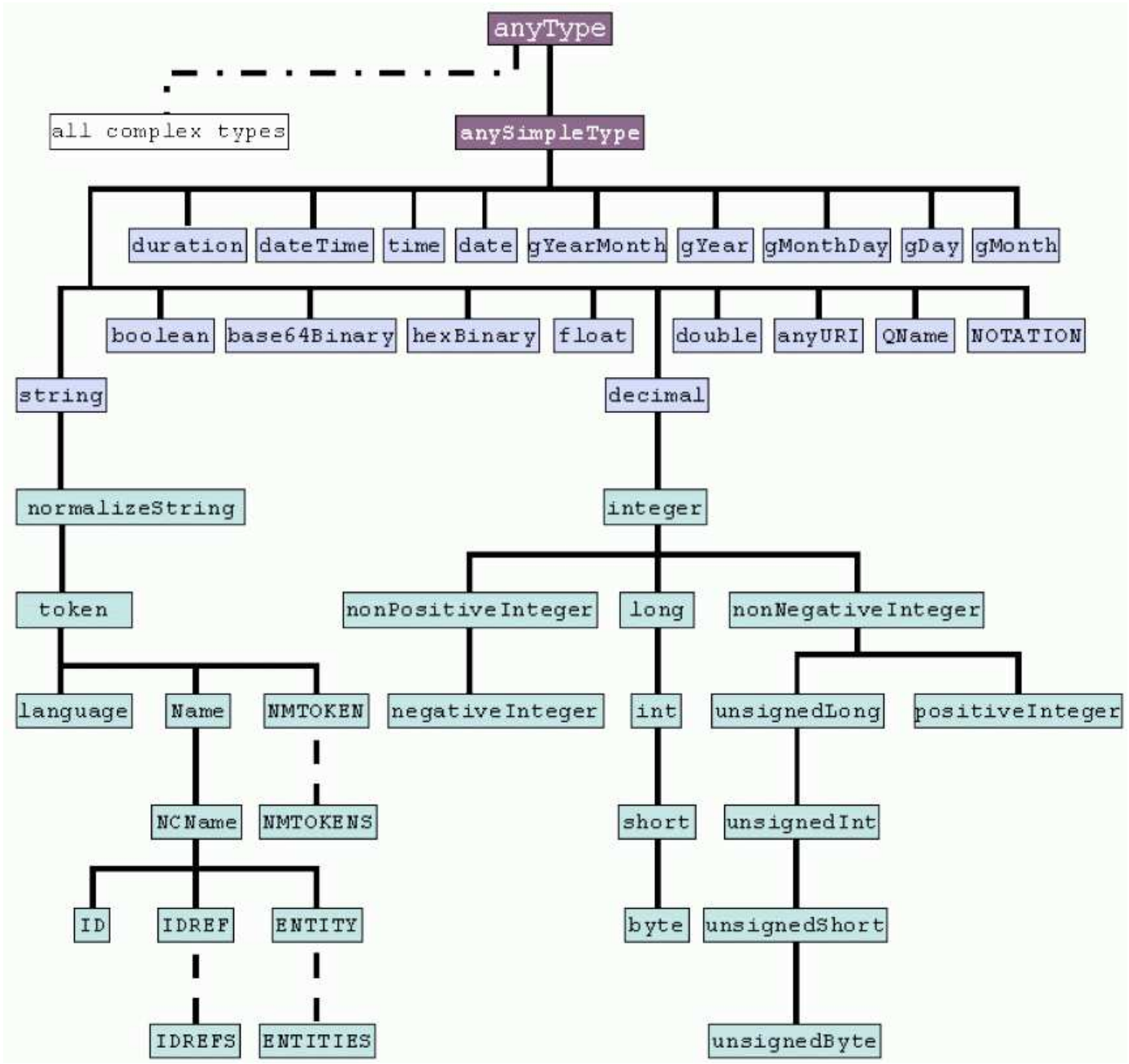
XML Schema

resolves weaknesses of DTDs wrt. formatted data:

- support for data types
  (DTDs: only child elements, PCDATA, mixed content and EMPTY)
- specification of structured data
  (e.g. arrays with lower/upper bounds)
- reuse of data-types
  (explicit support for elements only, but not for attributes)
- extensible type system
  (user-defined types / subtypes)
- support for namespaces
- support for reference mechanism
  (ID, IDREF(S)) supports local references only)
- DTD syntax in XML

Classification of XSD types

- atomic vs. aggregated:
  - atomic: atomic values, content not further interpreted
  - aggregated: lists, union (of different types)
- primitive vs. derived:
  - primitive: independent of other types
  - derived: subtype of other type
- predefined vs. user-defined:
  - XML Schema Part 2 defines 44 types
  - users may define additional types by aggregation or as subtypes of existing types


anyType
all complex types
anySimpleType
duration
dateTime
time
date
gYearMonth
gYear
gMonthDay
gDay
gMonth
boolean
base64Binary
hexBinary
float
double
anyURI
QName
NOTATION
string
decimal
normalizeString
integer
token
nonPositiveInteger
long
nonNegativeInteger
language
Name
NMTOKEN
negativeInteger
int
unsignedLong
positiveInteger
NCName
NMTOKENS
short
unsignedInt
ID
IDREF
ENTITY
byte
unsignedShort
IDREFS
ENTITIES
unsignedByte

Subtyping

- restriction
  - value restriction
    (string length, range, string pattern)
  - cardinality restriction
    (min,max bounds) of arrays

- extension
  (adding elements to a type, like type attributes in object-oriented programming)

- redefinition
  (redefine types of a given schema definition)

XML Schema from an IR point of view

Data types for IR applications:

- language
- thesaurus term / classification code
- geographic location
- . . .

→ most IR data types can not be defined at the syntactic level

→ XML schema can be used for IR data type definition, but type checking is not possible

Other XML Standards

- Style: XSL
- Transformations: XSLT (can be seen as a query language)
- Linking: XLink and XPointer
- Documents: DOM (Document Object Model), MathML, SVG (Scalable Vector Graphics), etc.

Part II

XML Query Languages

Content

The meeting (or clashing) place of databases and IR:

- Requirements
- Query languages history
- XPath locator language
- XQuery query language
- XML Databases

II.1 Requirements (1)

- From Semistructured Data
  - Selection: pattern + filter + constructor
  - Filtering
  - Reduction: pruned elements
  - Restructuring: grouping, sorting, etc.
  - Combine data: joins and semi-joins
  - Vague queries
  - Navigation
  - Aggregation
  - Existential and universal quantifiers
  - Data types
  - Insert, delete, and update operations

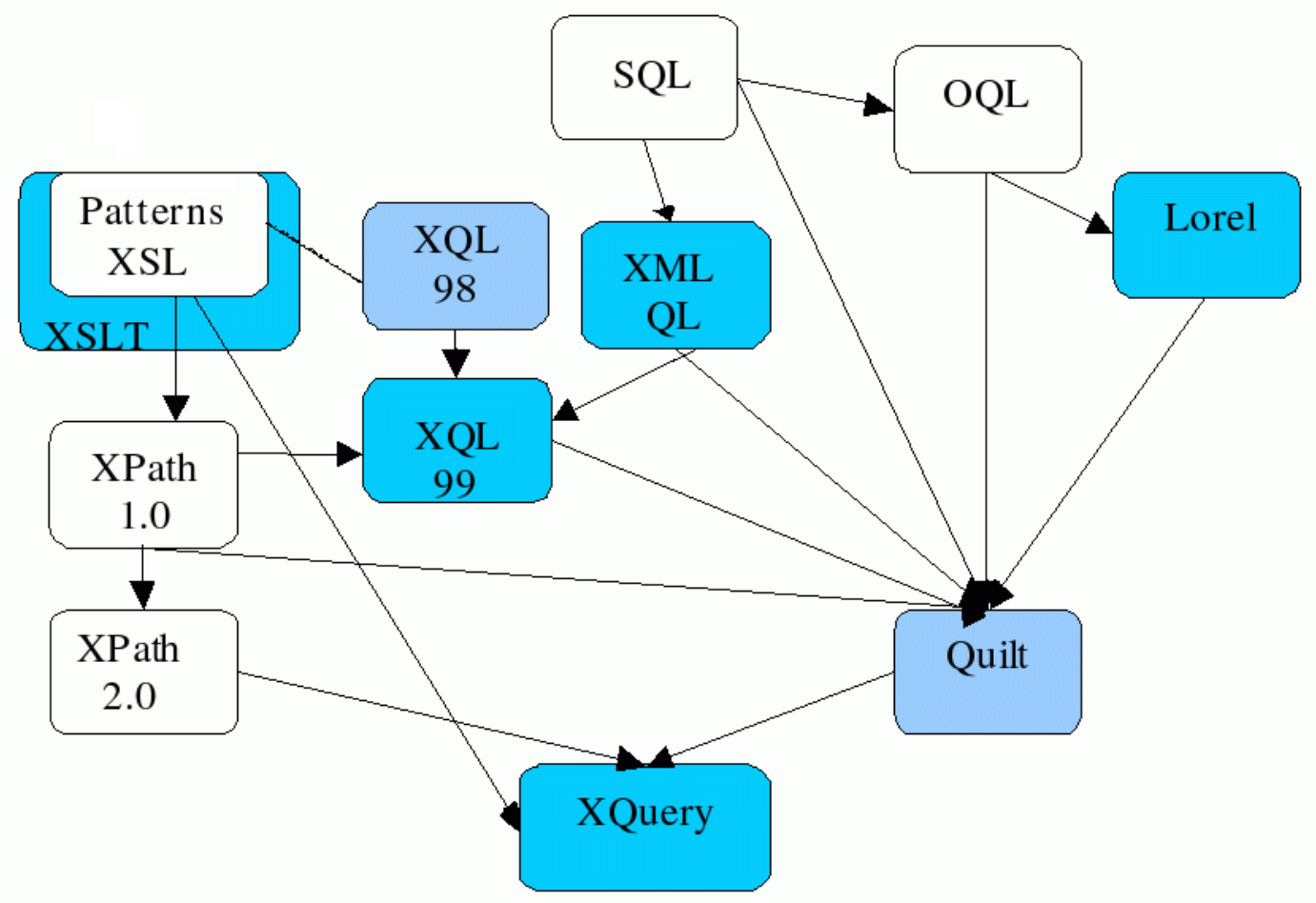
Requirements (2)

- From Information Retrieval
  - Keyword queries: Boolean, context, similarity, etc.
  - Pattern matching
  - Structural queries: inclusion, distance relations, etc.
  - Weighting query terms
  - Ranking

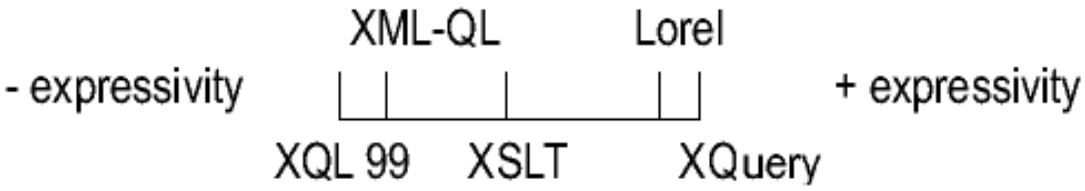
- Others
  - Use of metadata
  - DTD or Xschema awareness
  - Support for XLink and XPointer
  - Set operations on results

II.2 Query Languages History

From [**?**]:



Comparison: Basic Model (1)



	Lorel	XSLT	XML-QL	XQL 99	XQuery
Main functions	Queries of semi-structured data	Transformation of documents	Data queries, transformations, integration of XML data from different sources	Queries within a document and queries on collections of documents	Queries on heterogeneous data sources
Data model	Graph / Tree	Tree (such as XPath 1.0)	Graph	Tree (DOM of XML)	Ordered sequence of nodes (such as XPath 2.0)
Input source & format	XML Documents	XML Document/s + StyleSheet	XML Documents from different sources	XML Document/s	XML Document, XML Fragments, Collections of XML documents
Output information	XML Document (Ordered list of identifiers of the resulting elements)	XML Document (Transformed XML tree), Collections of XML documents *(xsl:document)*	XML Document (XML Fragments)	XML Document (XML Fragments, List of resulting elements)	XML Document, XML Fragment, Collections of XML documents

Comparison: Semistructured Data (2)

		Lorel	XSLT	XML-QL	XQL 99	XQuery
Selection Operation	**Pattern/ Filter/ Constructor**	**select** *constructor* **from** *pattern* **where** *filter*	**<xsl:for-each select=** pattern **>** **<xsl:if match=**filter**>** **<copy-of />** **</xsl:if>** **</xsl:for-each>**	**WHERE** *pattern* **IN** *source*, *filter* **CONSTRUCT** *constructor*	pattern [filter]	**FOR** *patterns* **LET** *bindings* **WHERE** *filter* **RETURN** *constructor*
	Relational Operators	>, >=, <, <=, =, <>, ==	>, >=, <, <=, =, !=	>, >=, <, <=, =, !=	>, >=, <, <=, =, !=	>, >=, <, <=, =, != For nodes: ==, !==
	Boolean Operators	*and, or, not*	*and, or*	No	*and, or*	*AND, OR*
	Nesting queries	Yes	Yes	Yes	Yes	Yes
	Creation of new elements	Yes	Yes	Yes	No	Yes
Filtering of elements preserving hierarchy		No	Yes (using templates)	No	Yes	Yes *(filter)*
Reduction		No	Yes	No	Yes	No
Restructuring operations	**Grouping of results**	Yes *(group by)*	No	No	Only by structure, not by value	Yes
	Skolem Functions	Yes	No	Yes	No	Yes
	Sorting of results	Yes *(order by)*	Partial (xsl:sort[a])	Yes *(ORDER-BY)*	No	Yes *(SORTBY)*
Inter-document links (join), Intra-documents links (semi-join)		Join,Semi-join	Semi-join	Join, semi-join	Semi-join, join	Join, semi-join

Comparison: Semistructured Data (3)

		Lorel	XSLT	XML-QL	XQL 99	XQuery
Use of tag variables		Yes	Yes	Yes	No	Yes
Path expressions		Regular expression operators: *, |, +, ? Qualifiers: >, @	XPath Expressions	Regular expression operators *, |, +, .	Wild card: * Path Operators: /, //	XPath Expressions
Dereferencing of IDREF(S) attributes		Yes (As a subelement using the point notation)	Yes (*id()*)	Yes (By means of a join)	Yes (*id()*)	Yes (Dereference Operator =>)
Set Functions		*min, max, count, sum, avg*	*sum, count*	*min, max, count, sum, avg*	*sum, count*	*min, max, count, sum, avg*
Quantifiers	**Existential**	Yes *(exists)*	Yes (implicit)	Yes (implicit)	Yes (implicit)	Yes *(SOME)*
	Universal	Yes *(for all)*	No	No	Yes *(all)*	Yes *(EVERY)*
Handling of datatypes (XML Schema)		Partial	No (under study)	No	No	Yes
Insertion, delete and update		Yes	Yes	No	No	No

Comparison: IR & others (4)

		Lorel	XSLT	XML-QL	XQL 99	XQuery
Keywords	**A word inside free text**	By means of path expressions	By means of path expressions	By means of path expressions	By means of path expressions	By means of path expressions
	Similarity	No	No	No	No	No
	Context	No	No	No	No	No
	Boolean Operators	Yes	Yes	No	Yes	Yes
Pattern matching		operators: *like, grep, soundex*	String operators and functions	*Like* operator	String operators and functions	String operators and functions
Structural Queries	**Structural Inclusion**	By means of path expressions	By means of path expressions	By means of path expressions	By means of path expressions	By means of path expressions
	Positional Inclusion	Yes	Yes	Yes	Yes	Yes
	Structural proximity	No	No	No	Yes (immediately precedes ";")	Context node
	Structural Order	By means of comparison of positional indexes	Yes (*preceding, preceding-siblings, following, following-siblings*)	By means of comparison of positional indexes	Yes (*before, after*)	Yes (*BEFORE, AFTER*)
Assignation of weighting to the terms of the query		No	No	No	No	No
RDF support		No	No	No	No	No
XLink and Xpointer support		No	No	No	Partial	No (In study)
Operations over sets		Intersection, union, difference	Union, difference	Intersection, union	Intersection, union	Intersection, union, difference

II.3 XPath locator language

restricted XML query language
retrieves complete elements (subtrees) of XML documents

used in

XSLT (Extensible Style Sheet Language Transformations)
for specifying argument of a transformation

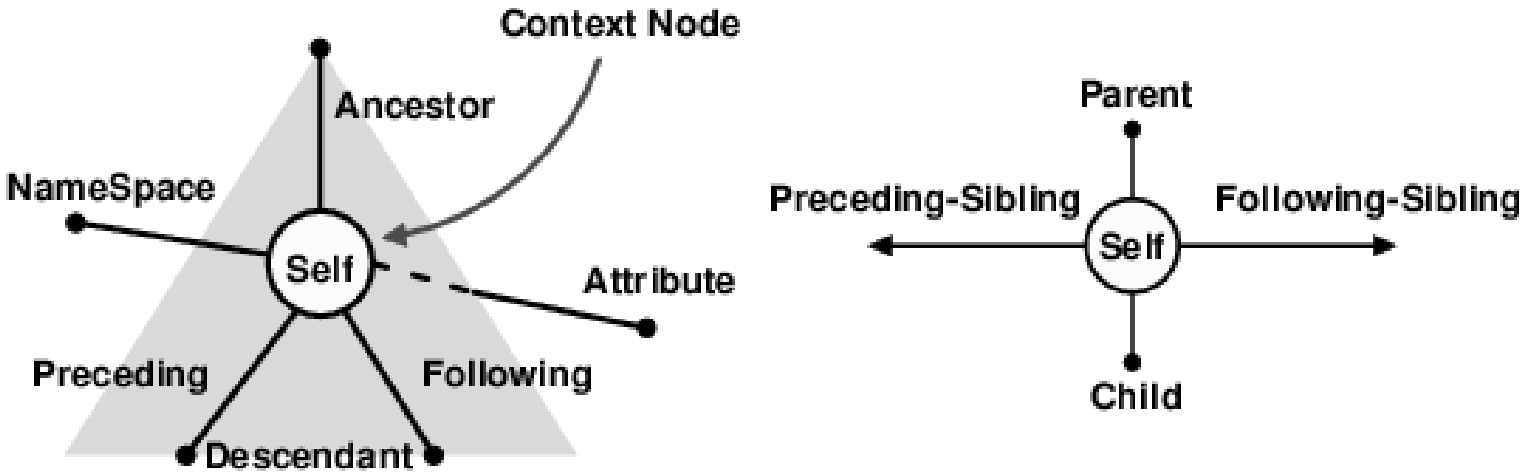
XPointer (XML Pointer)
for defining sources / targets of links

XQuery (XML Query Language)
for selecting elements that are arguments of further operations (value joins, restructuring, aggregation)

Path Expressions

- search for single elements:
  `heading`
- parent-child:
  `chapter/heading`
- ancestor-descendant:
  `chapter//heading`
- document root:
  `/book/*`
- filter wrt. structure:
  `//chapter[heading]`
- filter wrt content:
  `/document[@class="H.3.3" and author="John Smith"]`

Axes



Model: Ordered set of nodes with attributes

Beware: XPath 1.0 model is different from XPath 2.0 model

Axes

Generalization of locator operators

child:: children of the context node

descendant:: descendants of the context node

parent:: parent of the context node

ancestor:: ancestors of the context node

following-sibling:: all the following siblings of the context node

preceding-sibling:: all the preceding siblings of the context node

following:: all nodes in the same document as the context node that are after the context node in document order

preceding:: all nodes in the same document as the context node that are before the context node in document order,

attribute:: attributes of the context node

namespace:: namespace nodes of the context node

self:: just the context node itself

descendant-or-self:: context node and the descendants of the context node

ancestor-or-self:: context node and the ancestors of the context node

XPath axes examples

- `child::para` *para element children of the context node*
- `child::*` *element children of the context node*
- `child::text()` *text node children of the context node*
- `child::node()` *children of the context node, whatever their node type*
- `attribute::name` *name attribute of the context node*
- `attribute::*` *the attributes of the context node*

- `descendant::para` *para element descendants of the context node*

- `ancestor::div` *div ancestors of the context node*

- `ancestor-or-self::div` *div ancestors of the context node and, if the context node is a div element, the context node as well*

- `descendant-or-self::para` *para element descendants of the context node and, if the context node is a para element, the context node as well*

- `self::para` *context node if it is a para element, and otherwise selects nothing*

- `child::chapter/descendant::para` *para element descendants of the chapter element children of the context node*

- `child::*/child::para` *para grandchildren of the context node*

- */ document root (which is always the parent of the document element)*

- `/descendant::para` *para elements in the same document as the context node*

- `/descendant::olist/child::item` *item elements that have an olist parent and that are in the same document as the context node*

II.4 XQuery

XQuery 1.0 and XPath 2.0 Full-Text, November 2005

Weak data model:

- Ordered, labelled forest, with node identity and data types
- Static semantics: type inference, structural hierarchy
- Dynamic semantics: value inference
- Same data model as XPath 2.0
- Pure functional language with impure syntax
  - A query is an expression
  - Expressions can be nested
  - SQL-like basic structure:

```
FOR PathExpression
WHERE AdditionalSelectionCriteria
RETURN ResultConstruction
```

Advantages

- Expressive power
- Easy to learn
- Easy to implement (?)
- Optimizable in many environments (?)
- Related to concepts people already know
- Several current implementations
- The accepted W3C XML Query Language

Expressions

- Element constructors
- Path expressions
- Restructuring
  - FLWOR expressions
  - Conditional expressions
  - Quantified expressions
- Operators and functions
- List constructors
- Expressions that test or modify data-types

Element Constructors

Element constructors look like the XML they construct

```
<book year="1999">
    <title>Modern Information Retrieval</title>
    <author>
      <last>Baeza-Yates</last> <first>R.</first>
    </author>
    <author>
      <last>Ribeiro-Neto</last> <first>B.</first>
    </author>
    <publisher>Addison-Wesley</publisher>
    <price>49.95</price>
 </book>
```

Element Constructors: Examples

Generate an <emp> element that has an "empid" attribute and nested <name> and <job> elements, like:

```
<emp empid = "12345">
   <name>John Smith</name>
   <job>Anthropologist</job>
</emp>
```

Generate an <emp> element that has an "empid" attribute. The value of the attribute and the content of the element are specified by variables that are bound in other parts of the query.

```
<emp empid = {$id}>
   {$name}
   {$job}
</emp>
```

Path Expressions

XQuery uses the abbreviated syntax of XPath for path expressions

```
document("bib.xml")

/bib/book/author

/bib/book//*

//author[last="Knuth" and first="D."]

document("bib.xml")//author
```

Path Expressions: Extensions

```
<-- precedes, follows -->
//book[ author[last="Stevens"] precedes author[last="Abiteboul"] ]

<--  Namespaces -->
 namespace rev = "www.reviews.com"
 //rev:rating
```

Path Expressions: Examples

In the second chapter of the document named "zoo.xml", find the figure(s) with caption "Tree Frogs".

```
document("zoo.xml")//chapter[2]//figure[caption = "Tree Frogs"]
```

Find all the figures in chapters 2 through 5 of the document named "zoo.xml."

```
document("zoo.xml")//chapter[2 TO 5]//figure
```

Find captions of figures that are referenced by `<figref>` elements in the chapter of "zoo.xml" with title "Frogs".

```
document("zoo.xml")//chapter[title = "Frogs"]
//figref/@refid=>fig/caption
```

XQuery Examples (1)

List the names of the second-level managers of all employees whose rating is "Poor".

```
//emp[rating = "Poor"]/@mgr=>emp/@mgr=>emp/name
```

Find all captions of figures and tables in the chapter of "zoo.xml" with title "Monkeys".

```
document("zoo.xml")//chapter[title = "Monkeys"]
//(figure | table)/caption
```

From a document that contains employees and their monthly salaries, extract the annual salary of the employee named "Fred".

```
//emp[name="Fred"]/salary * 12
```

FLWOR Expressions

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>
```

FOR vs. LET

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>
```

Conditional Expressions

```
IF expr THEN expr ELSE expr
FOR $h IN //holding
RETURN
   <holding>
           { $h/title,
            IF ($h/@type = "Journal")
               THEN $h/editor
               ELSE $h/author
           }
   </holding>
```

Sorted Expressions:

```
expr ORDER BY (expr ASCENDING , ... )

FOR $b IN //book
RETURN
    $b ORDER BY(title, author[1]/name)
```

Inner and Outer (Semi) Joins

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

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

Quantifiers

EVERY var IN expr SATISFIES expr

SOME var IN expr SATISFIES expr

```
FOR $b IN //book
WHERE SOME $p IN $b//para SATISFIES
     contains($p, "sailing")
     AND contains($p, "windsurfing")
RETURN $b/title
```

```
FOR $b IN //book
WHERE EVERY $p IN $b//para SATISFIES
     contains($p, "sailing")
     AND contains($p, "windsurfing")
RETURN $b/title
```

FLWOR: Data for Examples

```
<book>
 <title> XML: An Introduction</title>
 <author>Smith </author> <author>Miller</author>
 <publisher>Morgan Kaufmann</publisher>
 <year>1998</year>
 <price>50</price>
</book>
<book>
 <title>XSLT Course</title>
 <author> Jones</author>
 <publisher>Addison Wesley</publisher>
 <year>2000</year>
 <price>40</price>
</book>
```

XQuery Examples (2)

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
```

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>
```

XQuery Examples (3)

List the publishers who have published more than 100 books.

```
<big_publishers>
   {
    FOR $p IN distinct(document("bib.xml")//publisher)
    LET $b := document("bib.xml")//book[publisher = $p]
    WHERE count($b) > 100
    RETURN $p
   }
</big_publishers>
```

XQuery Examples (4)

Invert the structure of the input document so that, instead of each book element containing a sequence of authors, each distinct author element contains a sequence of book-titles.

```
<author_list>
   {
    FOR $a IN distinct(document("bib.xml")//author)
    RETURN
      <author>
         <name> {$a/text()} </name>
         {
          FOR $b IN document("bib.xml")//book[author = $a]
          RETURN $b/title
         }
      </author>
   }
</author_list>
```

XQuery Examples (5)

For each book whose price is greater than the average price, return the title of the book and the amount by which the book's price exceeds the average price.

```
<result>
   {
    LET $a := avg(document("bib.xml")//book/price)
    FOR $b IN document("bib.xml")//book
    WHERE $b/price > $a
    RETURN
       <expensive_book>
          {$b/title}
          <price_difference>
             {$b/price - $a}
          </price_difference>
       </expensive_book>
   }
</result>
```

XQuery Examples (6)

Construct a new element having the same name as the element bound to $e. Transform all the attributes of $e into subelements, and all the subelements of $e into attributes.

```
<{name($e)}>
   {
    FOR $c IN $e/*
    RETURN attribute(name($c), string($c))
   }
   {
    FOR $a IN $e/@*
    RETURN
       <{name($a)}>
          {string($a)}
       </>
   }
</>
```

Conditions on Text

```
Equality:
       //section[title="Procedure"]
Full-text:
      //section[contains(title, "Procedure")]
```

More Full-text support in the future

Last published requirements: May 2003

Full-text Requirements (1)

- Full-Text predicates and SCORE functions independently
- Full-Text predicates use a language subset of SCORE functions
- Allow the user to return and sort-by SCORE (float in 0-1)
- SCORE must not require explicit global corpus statistics
- SCORE algorithm should be provided and can be disabled

Full-text Requirements (2)

- Minimal operations:
  - single-word and phrase search with stopwords
  - suffixes, prefix, infix
  - proximity searching (with order)
  - Boolean operations
  - Word normalization, diacritics
  - Ranking, relevance
- Search over everything, including attributes
- Proximity across markup elements
- Extensible

XQuery 1.0 and XPath 2.0 Full-Text

W3C Working Draft 1 May 2006

Full-Text Operators

- single words, phrases, any/all of a set of words
- logical operators: FTOr, FTAnd, FTMIldNot, FTUnaryNot
- context operators: FTOrder, FTScope, FTDistance, FTWindow, FTTimes, FTContent (start / end / entire content)
- Match options: case, diacritics, stemming, thesaurus, stop-words, language, wild cards

Examples

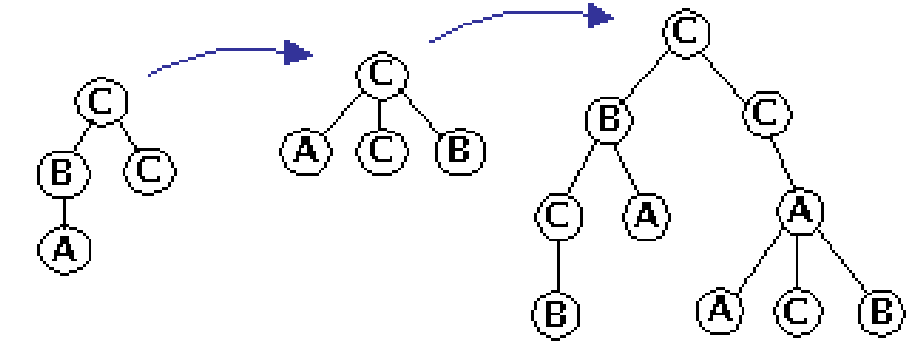
```
/books/book[title ftcontains ("dog" with stemming) && "cat"]/author

for $b score $s
    in /books/book[content ftcontains "web site" && "usability"
                   and .//chapter/title ftcontains "testing"]
return $s

for $b score $s
    in /books/book[content ftcontains "web site" && "usability"]
where $s > 0.5
order by $s descending
return <result>
          <title> {$b//title} </title>
          <score> {$s} </score>
       </result>
```

```
/book[@number="1" and ./title ftcontains {"Expert",
"Reviews"} all]

for $b in /books/book
let score $s := $b/content ftcontains ("web site" weight 0.2)
                                   && ("usability" weight 0.8)
return <result score="{$s}">{$b}</result>

/book[. ftcontains "usability" && "testing"
same paragraph]

/book ftcontains "web" && "site" &&
"usability" distance at most 2 words
```

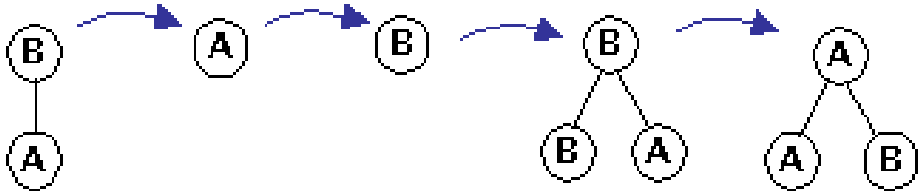
Filters

Filter(expression)

Result is an "ordered forest" that preserves sequence and hierarchy




```
LET $x := /C
```




```
filter(//A | //B)
```

Functions

- Built-in functions: max(), min(), sum(), count(), avg(), distinct(), empty(), contains()
- User-defined functions
- Defined in XQuery syntax
- Can be recursive
- Can be typed
- Extensibility mechanisms planned

Example:

```
define 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"))
```

XQuery Update Facility

W3C Working Draft 8 May 2006

- Insert
- Delete
- Replace
- Rename
- Transform

Examples

```
do insert <year>2005</year>
    after fn:doc("bib.xml")/books/book[1]/publisher

do delete fn:doc("bib.xml")/books/book[1]/author[last()]

do replace fn:doc("bib.xml")/books/book[1]/publisher
with fn:doc("bib.xml")/books/book[2]/publisher

do rename fn:doc("bib.xml")/books/book[1]/author[1]
as "principal-author"

for $e in //employee[skill = "Java"]
return
   transform
      copy $je := $e
      modify do delete $je/salary
      return $je
```