Query Languages for XML

- XPath
- XQuery
- XSLT (not being covered today!)

(Slides courtesy Wenfei Fan, Univ Edinburgh and Bell Labs)

Common Querying Tasks

- Filter, select XML values
  - Navigation, selection, extraction
- Merge, integrate values from multiple XML sources
  - Joins, aggregation
- Transform XML values from one schema to another
  - XML construction

Query Languages

- XPath
  - Common language for navigation, selection, extraction
  - Used in XSLT, XQuery, XML Schema, . . .
- XQuery 1.0: XML ⇒ XML
  - Strongly-typed query language
  - Large-scale database access
  - Safety/correctness of operations on data
- XSLT: XML ⇒ XML, HTML, Text
  - Loosely-typed scripting language
  - Format XML in HTML for display in browser
  - Highly tolerant of variability/errors in data

XML data: Running example

XML input: www.a.b/bib.xml

```
<book   year="1996">
    <title>  HTML </title>
    <author>  <last> Lee </last>  <first> T. </first></author>
    <author>  <last> Smith</last> <first>C.</first></author>
    <publisher>  Addison-Wesley </publisher>
    <price>  59.99 </price>
</book>
<book   year="2003">
    <title>  WMD </title>
    <author>  <last> Bush</last>  <first> G.</first></author>
    <publisher> white house </publisher>
</book>
```

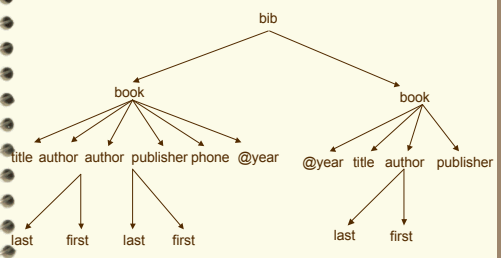

DTD

```
<!ELEMENT     bib       (book*) >
<!ELEMENT     book      (title, (author+  | editor+),
                          publisher?,  price?)  >
<!ATTLIST     book      year    CDATA      #required >
<!ELEMENT     author    (last, first)>
<!ELEMENT     editor    (last, first, affiliation)>

<!ELEMENT     publisher  (#PCDATA) >
....
```


Data model

Node-labeled, ordered tree



XPath

W3C standard: www.w3.org/TR/xpath

- Navigating an XML tree and finding parts of the tree (node selection and value extraction)

  Given an XML tree T and a context node n, an XPath query Q returns
  - the set of nodes reachable via Q from the node n in T – if Q is a unary query
  - truth value indicating whether Q is true at n in T – if Q is a Boolean query.
- Implementations: XALAN, SAXON, Berkeley DB XML, Monet XML – freeware, which you can play with
- A major element of XSLT, XQuery and XML Schema

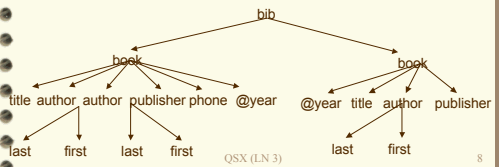
XPath constructs

XPath query Q:

- Tree traversal: downward, upward, sideways
- Relational/Boolean expressions: qualifiers (predicates)
- Functions: aggregation (e.g., count), string functions

//author[last="Bush"]

//book[author/last="Bush"]/title | //book[author/last="Blair"]/title



Downward traversal

Syntax:

Q ::= . | l | @l | Q/Q | Q | Q | //Q | /Q | Q[q]

q ::= Q | Q op c | q and q | q or q | not(q)

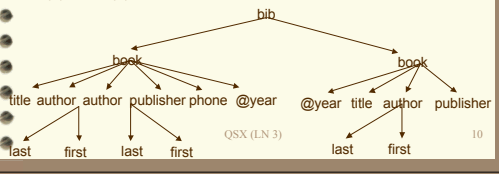
- .: self, the current node
- l: either a tag (label) or *: wildcard that matches any label
- @l: attribute
- /, |: concatenation (child), union
- //: descendants or self, "recursion"
- [q]: qualifier (filter, predicate)
  - op: =, !=, <=, <, >, >=, >
  - c: constant
  - and, or, not(): conjunction, disjunction, negation

Existential semantics: /bib/book[author/last="Bush"]

Examples:

- parent/child: /bib/book
- ancestor//descendant: bib//last, //last
- wild card: bib/book/*
- attributes: bib/book/@year
- attributes with wild cards: //book/@*
- union: editor | author

Are book/author and //author "equivalent" at context nodes (1) root, (2) book, (3) author?



Filters (qualifiers)

- //book[price]/title -- titles of books with a price
- //book[@year > 1991]/title -- titles of books published after 1991
- //book[title and author and not(price)]/title

  titles of books with authors, title but no price
- //book[author/last = "Bush"]/title

  titles of books with an author whose last name is Bush
- //book[author or editor]/title

  titles of books with either an author or an editor

Existential semantics:

What is /[//@id]? /[//[not(@id)]]? /[not(//[not(@id))]] ?

Upward traversal

Syntax:

Q ::= . . . | ../Q | ancestor ::Q | ancestor-or-self::Q

- ../: parent
- ancestor, ancestor-or-self: recursion

Example:

- //author[../title = "WMD"]/last

  find the last names of authors of books with the title "WMD"
- ancestor :: book[//last="Bush"]

  find book ancestors with "Bush" as its last descendant

Are the following equivalent to each other (context node: a book)?

../book/author, ./author

Sideways

Syntax:

Q ::= . . . | following-sibling ::Q | preceding-sibling::Q

- following-sibling: the right siblings
- preceding-sibling: the left siblings
- position function (starting from 1): e.g., //author[position( ) < 2]

Example:

- following-sibling :: book [//last="Bush"]

  find the books that are right siblings and are written by Bush
- preceding-sibling :: book[//last="Bush"]

  find the books that are left siblings and are written by Bush

Query Languages for XML

- XPath
- XQuery
- XSLT

XQuery

W3C working draft www.w3.org/TR/xquery

Functional, strongly typed query language: Turing-complete

- XQuery = XPath + …
  - for-let-where-return (FLWR) ~ SQL's SELECT-FROM-WHERE Sort-by
  - XML construction (Transformation)
  - Operators on types (Compile & run-time type tests)
  - + User-defined functions
    - Modularize large queries
    - Process recursive data
  - + Strong typing
    - Enforced statically or dynamically
- Implementation: GALAX, SAXON

  http://www-db.research.bell-labs.com/galax/

  http://www.saxonica.com

FLWR Expressions

For, Let, Where, OrderBy, return

Q1: Find titles and authors of all books published by Addison-Wesley after 1991.

```
<answer>{
   for  $book  in /bib/book
   where $book/@year > 1991  and  $book/publisher='Addison-Wesley'
   return    <book>
                <title>  {$book/title } </title>,
                for  $author  in $book/author   return
                  <author>  {$author } </author>
             </book>
}</answer>
```

- for loop; $x: variable
- where: condition test; selection
- return: evaluate an expression and return its value

join

Find books that cost more at Amazon than at BN

```
<answer>{
   let  $amazon := doc("http://www.amozon.com/books.xml"),
        $bn  :=  doc("http://www.BN.com/books.xml")
   for    $a  in $amozon/books/book,
          $b  in $bn/books/book
   where    $a/isbn  = $b/isbn    and     $a/price > $b/price
   return   <book> {$a/title, $a/price, $b/price }  <book>
}</answer>
```

- let clause
- join: of two documents

Conditional expression

Q2: Find all book titles, and prices where available

```
<answer>{
   for  $book  in /bib/book
   return   <book>
               <title>  {$book/title } </title>,
               { if   $book[price]
                 then  <price> {$book/price } </price>
                 else  ( ) }
            </book>
}</answer>
```

Summary and Review

Query languages for XML

- XPath: navigating an XML tree
- XQuery: XML query language

Very powerful (as opposed to relational algebra); however, query processing/optimization is hard – open issue!

For thought:

- Write queries on the school document you created, using XPath, XSLT and XQuery; display the query answers in HTML
- Find some queries in XPath, XSLT and XQuery that are not expressible in SQL, even when relational data is considered (i.e., relational data represented in a canonical form in XML)