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

```xml
<book year="1996">
  <title>HTML</title>
  <author>Lee <last>T. <first></author>
  <author>Smith <last>C. <first></author>
  <publisher>Addison-Wesley</publisher>
  <price>59.99</price>
</book>

<book year="2003">
  <title>WMD</title>
  <author>Bush <last>G. <first></author>
  <publisher>White House</publisher>
</book>
```

DTD

```xml
<!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, MonetXML – 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

Examples:
- $a$ / $b$ / $c$ = // $b$ / $c$ – titles of books with a price
- $a$ / $b$ / $c$ = // $b$ / $c$ – titles of books published after 1991
- $a$ / $b$ / $c$ = // $b$ / $c$ – titles of books published after 1991
- $a$ / $b$ / $c$ = // $b$ / $c$ – titles of books published after 1991
- $a$ / $b$ / $c$ = // $b$ / $c$ – titles of books published after 1991
- $a$ / $b$ / $c$ = // $b$ / $c$ – titles of books published after 1991
- $a$ / $b$ / $c$ = // $b$ / $c$ – titles of books published after 1991
- $a$ / $b$ / $c$ = // $b$ / $c$ – titles of books published after 1991
- $a$ / $b$ / $c$ = // $b$ / $c$ – titles of books published after 1991

Downward traversal

Syntax:

\[
Q := . | l | @l | Q/Q | Q/Q | Q/L | Q/L | Q[L] \\
Q := Q | Q \text{ op } c | c \text{ and } q \text{ or } q | \text{not}(q)
\]

- : self, the current node
- $l$: either a tag (label) or *: wildcard that matches any label
- @: attribute
- $L$: concatenation (child), union
- $Q$: qualifier (filter, predicate)
  - op: =, !=, <=, <, >, >=, >
  - c: constant
  - and, or, not(): conjunction, disjunction, negation

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

Upward traversal

Syntax:

\[
Q := . . . | .Q | ancestor :: Q | ancestor-or-self :: Q
\]

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

Example:

\[
(author [.title = "WMD"] last) \\
\text{find the last names of authors of books with the title "WMD"}
\]

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

\[
..book/author, /author
\]
## Query Languages for XML

### XPath
- Functional, strongly typed query language: Turing-complete
- XPath = XQuery +...
- For-let-where-return (FLWR) ~ SQL
  - 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

### XQuery
- W3C working draft [www.w3.org/TR/xquery](http://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

### FLWR Expressions
- For, Let, Where, OrderBy, return
- Q1: Find titles and authors of all books published by Addison-Wesley after 1991.
  ```xquery```
  ```
  let $amazon := doc("http://www.amazon.com/books.xml"),
  $bn := doc("http://www.bn.com/books.xml")
  for $a in $amazon/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>
  ```

### Conditional expression
- Q2: Find all book titles, and prices where available
  ```xquery```
  ```
  for $book in bib/book
  return <book>
  <title>{$book/title } </title>,
  {if $book[price] then <price> {$book/price } </price>
  else () }
  </book>
  ```
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)