Querying and Storing XML

Week 1
Introduction & course overview, XML basics
January 15-18, 2013

Key information

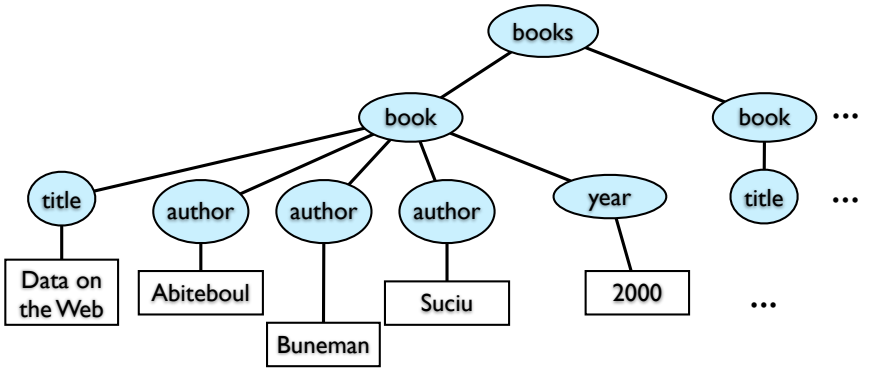
- Instructor: James Cheney, jcheney@inf.ed.ac.uk
- Lectures: 11:10-12:00, Tuesday/Friday
  - LT4, 7 Bristo Square
- Office Hours: (IF 5.29)
- TA: Clare Llewellyn, s1053147@sms.ed.ac.uk
- Webpage:
  - http://www.inf.ed.ac.uk/teaching/courses/qsx/
  - Weekly readings, project ideas/suggested readings

What is XML?

- eXtensible Markup Language [W3C 1998]
- Ask five different people, get five different answers...
  - a self-describing data format?
  - a generalization of HTML?
  - the future/past?
  - best thing since sliced bread/clunky and evil?
  - a metalanguage?
    - http://en.wikipedia.org/wiki/List_of_XML_markup_languages

In a nutshell:

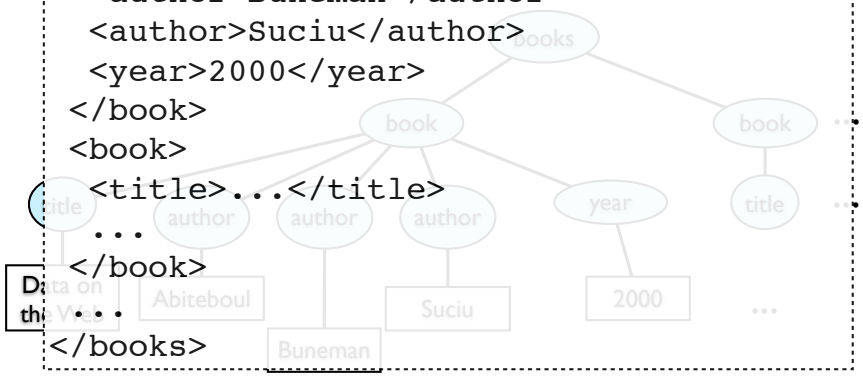
- A (meta)language for *semi-structured data* (trees)



In a nutshell:

- A (meta) language for *semi-structured data* (trees)

```
<?xml version="1.0"?>
<books>
 <book>
  <title>Data on the Web</title>
  <author>Abiteboul</author>
  <author>Buneman</author>
  <author>Suciu</author>
  <year>2000</year>
 </book>
 <book>
  <title>...</title>
  ...
 </book>
 ...
</books>
```



XML for markup/ documents

- SGML
- HTML - hypertext markup language
- TEI - Text markup, language technology
- DocBook - documents -> html, pdf, ...
- SMIL - Multimedia
- SVG - Vector graphics
- MathML - Mathematical formulas

XML for (semi-)structured data

- MusicXML
- NewsML
- iTunes
- DBLP http://dblp.uni-trier.de
- CIA World Factbook
- IMDB http://www.imdb.com/
- XBEL - bookmark files (in your browser)
- KML - geographical annotation (Google Maps)
- XACML - XML Access Control Markup Language

XML as an abstract syntax

- Many systems now use XML as a general-purpose syntax for other programming languages or configuration files...
  - Java servlet config (web.xml)
    - Apache Tomcat, Google App Engine, ...
  - Web Services - WSDL, SOAP, XML-RPC
  - XUL - XML User Interface Language (Mozilla/Firefox)
  - BPEL - Business process execution language
  - Other Web standards:
    - XSLT, XML Schema, XQueryX
    - RDF/XML
    - OWL - Web Ontology Language
    - MMI - Multimodal interaction (phone + car + PC)

XML tools

- Standalone:
  - xsltproc, mxquery, calabash (XProc)
- Most PLs have XML parsers; many have XSLT engines/ libraries also
  - SAX (streaming), DOM (in-memory) interfaces
  - libxml2, expat, libxslt (C)
  - Xerces, Xalan (Java)

- XPath (path expressions) used in many languages
  - JavaScript/JQuery
  - XSLT, XQuery

Native XML databases

- Offer native support for XML data & query languages (not building on existing RDBMS)
  - Galax
  - MarkLogic
  - eXist
  - BaseX
  - among others...

- Suitable for new or lightweight applications
  - but some lack features like transactions, views, updates

XML support in industry

- Most commercial RDBMSs now provide some XML support

  - Oracle 11g - XML DB

  - IBM DB2 pureXML

  - Microsoft SQL Server - XML support since 2005
    - Language Integrated Query (LINQ) targets SQL & XML in .NET programs
- Data publishing, exchange, integration problems are very important
  - big 3 have products for all of these
  - SQL/XML standard for defining XML views of relational data

The wonderful thing about standards...

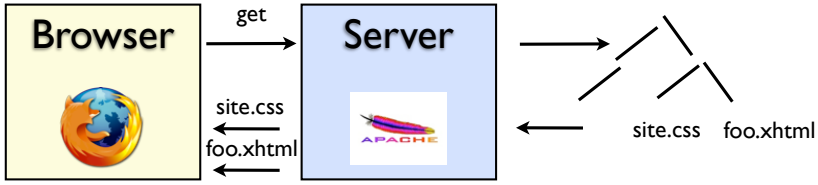
- There are **so many** to choose from!
  - XPath 1.0, 2.0, XSLT 1.0, 2.0, XQuery, XProc
  - RDF, RDFS, OWL 1.0, 2.0, SPARQL 1.0, 1.1, ...
- W3C process moves quickly
  - and is hit-or-miss
  - often driven by nontechnical/industrial issues
- Standards reflect compromises between needs of different communities
  - XML standards often compromise between "data" and "document" views of world
  - and it shows!

Beyond the hype (Google "I hate XML")

- XML "wave" (1999-200?): may have crested
- XML can be and has been (justifiably) criticized
  - bloat, ad hoc features, too many standards
  - wheel reinvention: why not LISP S-Expressions [McCarthy 1960]
- New semistructured formats/syntaxes now in vogue
  - RDF, JSON, YAML, Google Protocol Buffers
  - many XML vendors re-branding as "NoSQL"
- Nevertheless, the basic issues are pretty much the same
  - and XML is definitely not going away
- Our goal: rise above fray, understand essential issues in CS terms

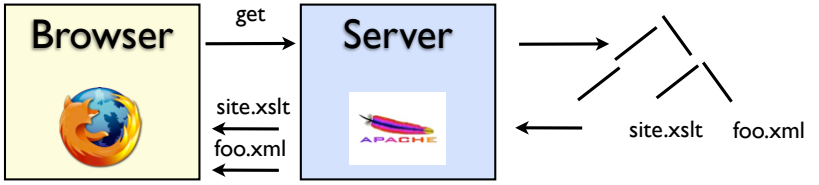
Where is XML used?

Static Web site



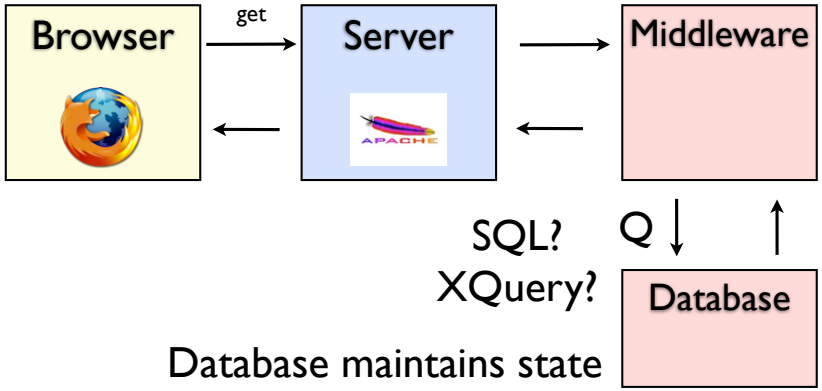
(X)HTML + CSS
(+ SVG + MathML)

Static Web site (XML + XSLT)

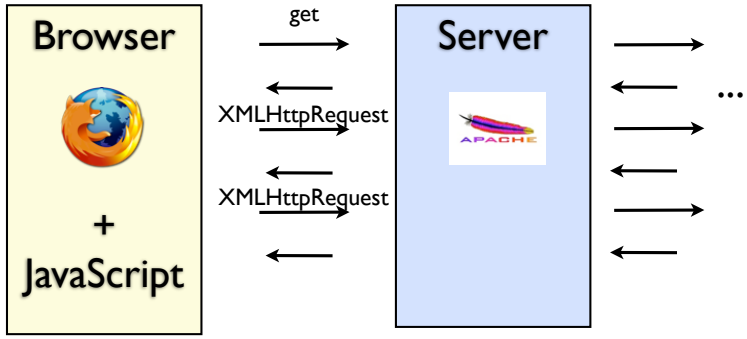


allows better factoring
into data + presentation

Dynamic Web site

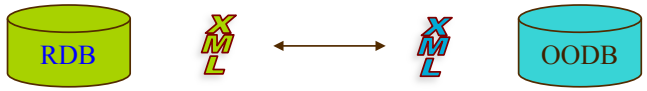


Web 2.0: Asynchronous Java and XML



Data exchange

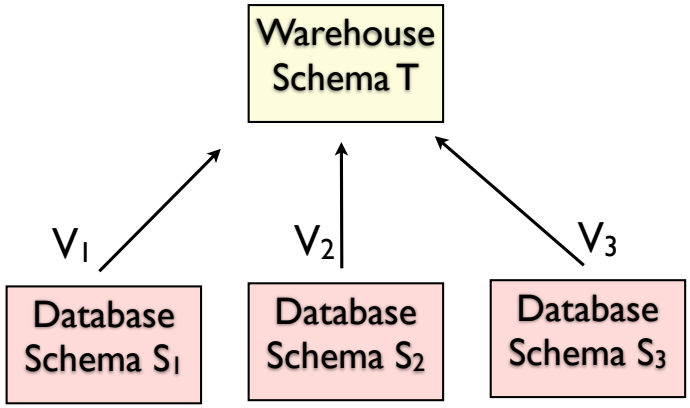
- Massive demand
  - across platforms/DBs
  - across enterprises



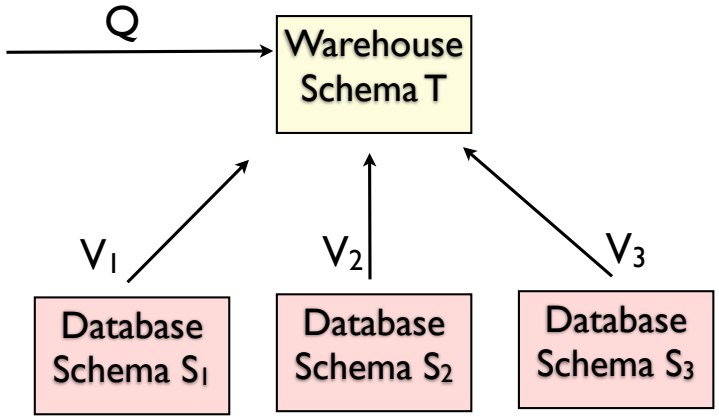
- XML has become the prime standard for data interchange on the Web

Data integration (warehousing)

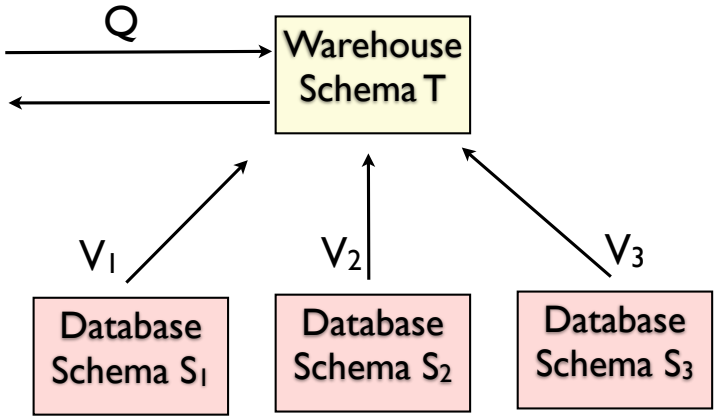
Data integration (warehousing)



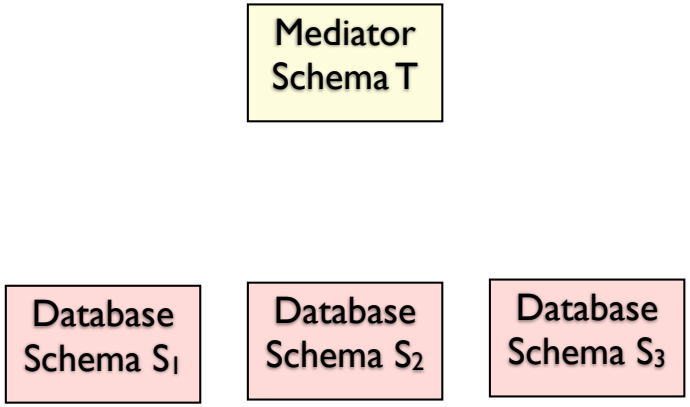
Data integration (warehousing)



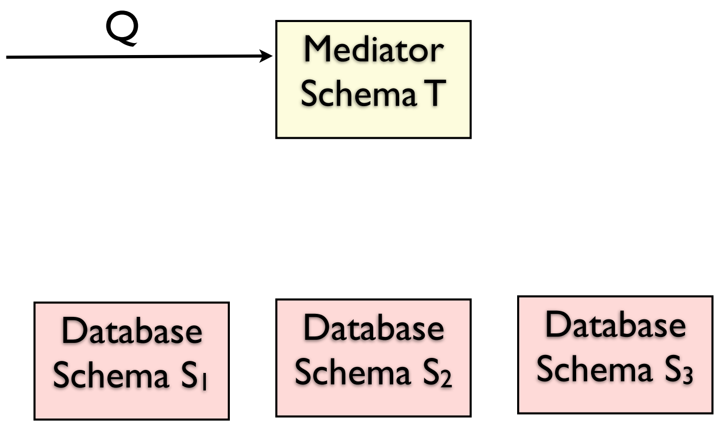
Data integration (warehousing)



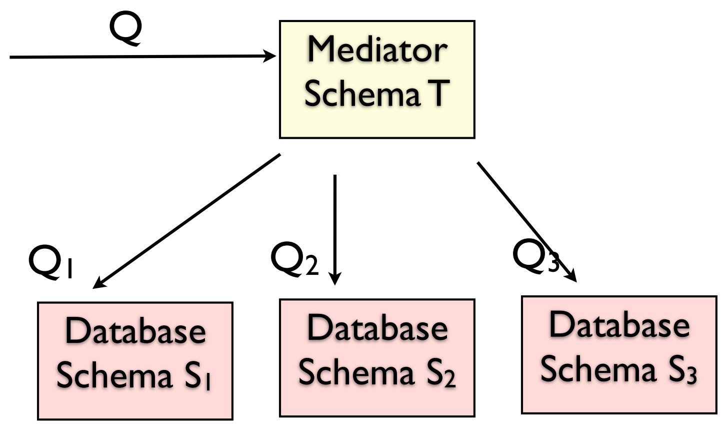
Data integration (mediation)



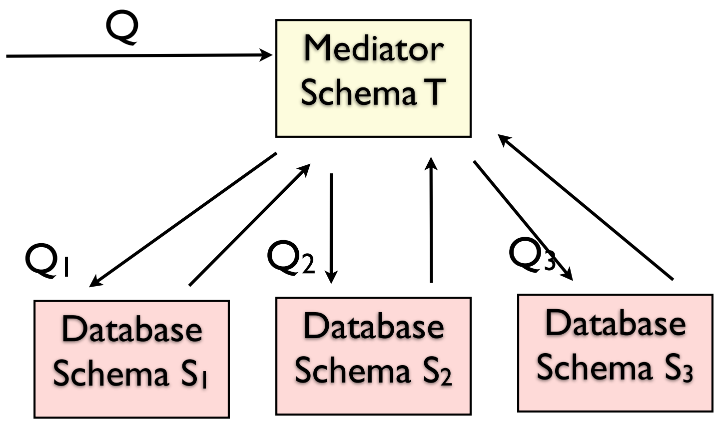
Data integration (mediation)



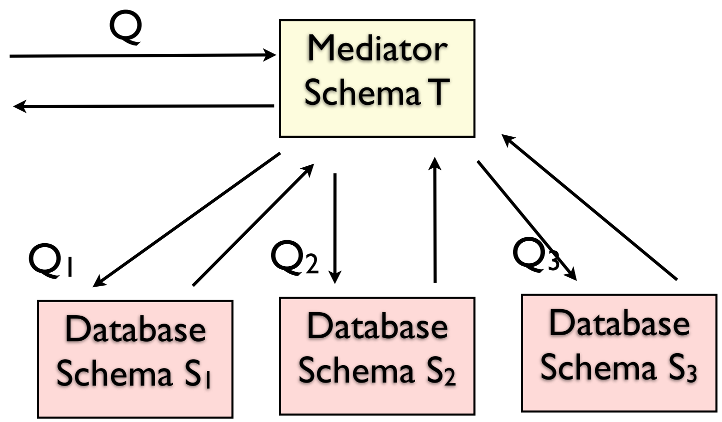
Data integration (mediation)



Data integration (mediation)



Data integration (mediation)



Course overview

What you need for this course

- Prerequisites:
  - *Language Semantics & Implementation* (PL)
  - or *Computers & Intractability* (Algorithms/complexity)
  - or permission
- *Database Systems* or *ADBS* wouldn't hurt
  - good if you have at least heard of SQL
- *Data Integration & Exchange*, *Extreme Computing* or *Natural Language Technologies* may complement

Evaluation

- 35%: Reviews (7 assignments)
  - Due **each Monday at 4pm until week 8**
  - Week 2-3: read/review **all 3 tutorials**
  - Week 4-8: read/review **2 out of 3 papers**
- 50%: Course project report
  - due **Monday, March 25, 4pm**
- 15%: Project presentation
  - during week 10-11 (**after** report due!)

Reviews

- Hand in reviews on **Monday at 4pm before** of the week in which the papers are to be discussed.
  - Online handin, details TBA soon
- Each review should be about **half a page**
  - summary, key ideas, questions you have, flaws you have found, and suggestions for improvement.
- Marked on "+/✓/-" scale.
  - + = 2 pts (outstanding)
  - ✓ = 1 pt (pass)
  - - = 0 pts (incomplete)

Projects

- The project is the main assessed component of the course.
- Projects can take two forms:
  - **Design and development projects**
    - (2-3 students) implement or improve upon an algorithm or system.
  - **Survey/benchmarking projects**
    - (1 student) surveying 5-10 research papers on a particular topic, or using several systems for the same task and comparing them
- **Project report**: 15-30 pages (typically longer for group projects)

Suggested timetable

- Week 3: select project, form groups
  - project ideas: http://www.inf.ed.ac.uk/teaching/courses/qsx/project.html
  - contact me or use qsx-students@inf.ed.ac.uk list
- Week 5: literature review/identify related work
- Week 7: Draft report/preliminary results
- Week 10-11: Final report & presentation
- **It is up to you to ensure that your project stays on track!**

Presentations

- Each group member must participate in presentation
- ~5 min per group member (e.g. 3-person group gets 15 minutes)
- Summarize background
- Present research question
- Summarize progress and next steps

Introductions

- Who you are
- What you want to get out of the course
- ...Start thinking about project groups early
  - Use mailing list qsx-students@inf.ed.ac.uk

What you should get from this course

- Skills/knowledge in demand in industry ($$)
  - **NOT:** Specific language/tool/system/protocol/API
  - you should be able to pick it up on your own.
- Research / team / project experience
- Background useful for working with other XML or Web standards/tools
- Chance to put diverse CS concepts into practice

Syllabus/topics

- Weeks 1-3: Foundations
- Weeks 4-5: Storage & publishing
- Weeks 6-8: Additional topics (updates, static analysis, provenance)
- Week 9: Break
- Week 10-11: Project presentations

Next time

- XML background

Foundations of XML

- XML itself
- Query and transformation languages
  - XPath: a path-based language for navigating / selecting parts of XML trees
  - XQuery: a "SQL for XML" query language
  - XSLT: a pattern-matching, recursive transformation language
- Schemas (DTDs, XML Schema)
  - validation, automata
  - Constraints (keys)

XML and databases

- XML "shredding": Storing/querying XML in relational databases
  - Shredding XML trees into relations (with or without schema)
  - Translating XML queries/updates to SQL over shredded representation
- XML "publishing": Providing XML views of legacy relational data
  - Translating XML queries over views to SQL
  - Schema-directed publishing

XML updates

- Beyond DOM: Updating XML stored as relations
- XQuery Update Facility
  - extending XQuery to support updates
- Updating XML views of relations
  - how to translate updates to XML views to SQL

Typechecking and static analysis

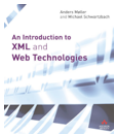
- Typed programming with XML
  - regular expression types and inference
- Predicting structure of result of queries
  - application: finding "path errors" in queries
- Using this to improve performance
  - e.g. detecting when a query and update do (or don't) "overlap"
  - can save time recomputing views

Provenance for semi-structured data

- Provenance: Understanding where data came from, how it has been produced
  - Increasingly important for scientific data
- Why and where provenance
- Annotation and XML query languages
- Provenance for curated (evolving) data

Reading

- Web Data Management (Abiteboul et al. 2011)
  - ch. 1-5 provide excellent overview of XML, Schemas, XPath, XQuery and shredding
  - also good coverage of RDF/OWL/SPARQL and cloud computing/ MapReduce
  - webdam.inria.fr/Jorge/



- Introduction to XML and Web Technologies (Moller, Schwartzach 2006)
  - now a bit dated but good coverage of XML and Java-based Web programming
  - slides, ch. 4 online
- Research papers - listed week-by-week on course web page

XML background

Some history

- SGML - (Charles Goldfarb, ISO 8879, 1986)
  - widely used for document management
  - but complex & hard to implement
- HTML - (Tim Berners-Lee, 1991)
  - most successful application of SGML
- XML - (W3C, 1998)
  - simplify SGML, for Web data/content
  - still pretty complicated, though

HTML: limitations

- HTML was intended as a declarative markup language
  - emphasizing structure over presentation
- But with success of Web, intense pressure for more presentation features
  - CSS helps separate content from structure, a little
- nevertheless, while great for human consumption, HTML is not suitable for representing general data
  - fixed set of tags
  - describe display format, not structure of data

Good things about XML

- Tags can be defined for specific applications other than HTML
- The structure of the data can be defined more precisely
  - DTDs, XML Schemas
- Structures can be arbitrarily nested
  - even including recursion
- XML standard does not define how data should be displayed
  - Style sheets (XSLT) can transform XML to HTML or other forms

Scraping data from HTML

```
<h2>Some data</h2>
<table border="2">
  <tr><th>A</th><th>B</th>
  <tr><td>1<td>2</td>
  <tr><td>3</td><td>4
</table>
```


Scraping data from HTML

```
<h2>Some data</h2>
<table border="2">
  <tr><th>A</th><th>B</th>
  <tr><td>1<td>2</td>
  <tr><td>3</td><td>4
</table>
```

Optional closing tags - hard to parse

Data as XML

```
<?xml version="1.0"?>
<root title="Some data">
 <table>
  <row><A>1</A><B>2</B></row>
  <row><A>3</A><B>4</B></row>
 </table>
</row>
```

Data as XML

Header

```
<?xml version="1.0"?>
<root title="Some data">
 <table>
  <row><A>1</A><B>2</B></row>
  <row><A>3</A><B>4</B></row>
 </table>
</row>
```


Data as XML

Attribute

```
<?xml version="1.0"?>
<root title="Some data">
 <table>
  <row><A>1</A><B>2</B></row>
  <row><A>3</A><B>4</B></row>
 </table>
</row>
```


Data as XML

Element

```
<?xml version="1.0"?>
<root tit                ta">
 <table>
  <row><A>1</A><B>2</B></row>
  <row><A>3</A><B>4</B></row>
 </table>
</row>
```


Data as XML

Text

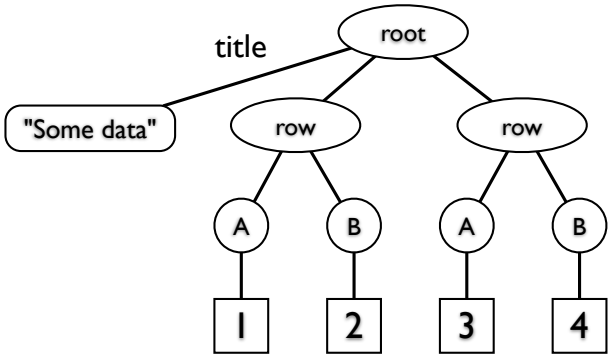
```
<?xml version="1.0"?>
<root title="Some data">
 <table>
  <row><A>1</A><B>2</B></row>
  <row><A>3</A><B>4</B></row>
 </table>
</row>
```

Data as XML

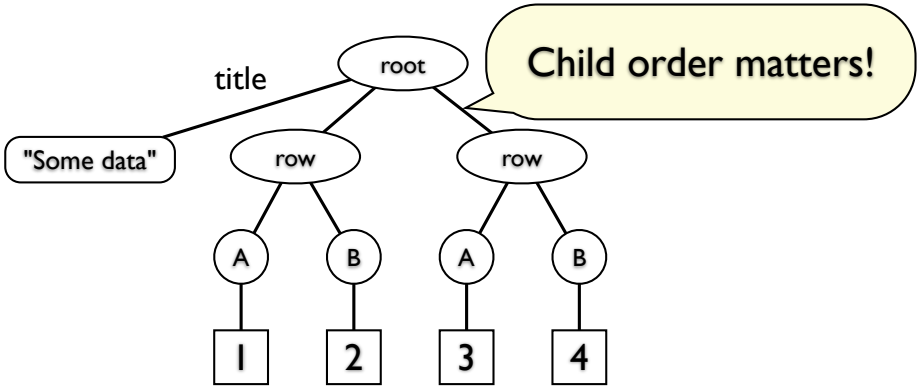
```
<?xml version="1.0"?>
<root title="Some data">
 <table>
  <row><A>1</A><B>2</B><
  <row><A>3</A><B>4</B></row>
 </table>
</row>
```

NOTE: Tag names have NO pre-defined meaning!

XML Data as trees



XML Data as trees



Basics

- An XML document consists of
- elements - `<a>...</a>`
  - tags come in pairs
  - tags must be properly nested (can't skip closing tag!)
- attributes - `<a b="x" c="y">`
  - key-value pairs associated with elements
- text values - `<a>foo<b>123</b></a>`
  - unquoted text inside elements

Elements

- Element: the segment between an start and its corresponding end tag
  - **Unique root element**
- subelement: the relation between an element and its component elements.

```
<person>
    <name> James Cheney </name>
    <tel> 0131 651 5658 </tel>
    <email> jcheney@inf.ed.ac.uk </email>
    <email> cheneyj@acm.org </email>
</person>
```


Nested structure

- nested tags can be used to express various structures, e.g., "records":

```
<person>
    <name> James Cheney </name>
    <tel> 0131 651 5658 </tel>
    <email> jcheney@inf.ed.ac.uk </email>
    <email> cheneyj@acm.org </email>
</person>
```

- a list: represented by using the same tags repeatedly:

```
<person> … </person>
<person> … </person>
...
```


Ordering

- XML elements are ordered!
  - How to represent sets in XML?
  - How to represent an unordered pair (a, b) in XML?
- Can one directly represent the following in a relational database?

```
<person> … </person>
<person> … </person>   …
    <person>
        <name> James Cheney </name>
        <tel> 0131 651 5658 </tel>
        <email> jcheney@inf.ed.ac.uk </email>
        <email> cheneyj@acm.org </email>
    </person>
```


Attributes

- A start tag may contain attributes describing certain "properties" of the element (e.g., dimension or type)

```
<picture>
    <height dim="cm"> 2400</height>
    <width dim="in"> 96 </width>
    <data encoding="gif"> M05-+C$ … </data>
</picture>
```

- References (meaningful only when a DTD is present):

```
<person id = "011"   pal="012">
    <name> George Bush</name>
</person>
<person id = "012"   pal="011">
    <name> Saddam Hussein </name>
</person>
```

Attribute structure

- XML attributes cannot be nested -- flat
  - the names of XML attributes of an element must be unique.
  - one can't write `<person pal="Blair" pal="Saddam"> ...`
- XML attributes are not ordered:

```
<person id = "011" pal="012">
    <name> George Bush</name>
</person>
```

is the same as

```
<person pal="012" id = "011">
    <name> George Bush</name>
</person>
```

- Attributes vs. subelements: unordered vs. ordered, and
  - attributes cannot be nested (flat structure)
  - subelements cannot represent references

Extras

- *entity references*: `& &quot; >`
  - textual substitution; allows escaping special characters
  - you can define your own if you want
- *processing instructions*: `<? foo : bar ?>`
  - can be used to pass information to processors
- *comments*: `<!-- foo -->`
- *CDATA sections:* - `<![CDATA[ I <3 XML ]]>`
  - allows including raw text (<, >, &, etc. uninterpreted)
- Luckily, these are mostly irrelevant to use of XML **for data**
  - but you need to know about them when writing reading/writing XML as text

Quiz

- Groups of 2-3: Find as many errors in this XML document as you can

```
<?xml version="1.0">
<books>
 <book id="1'>
  <title>Data on the Web</title>
  <authors>
    <author id="a1">Abiteboul
    <author id=a2>Buneman & </author>
    <author id='a3'>Suciu</authors>
  </author>
  <year>2000/year>
  <publisher>Addison-Wesley</publisher>
</books>
<foo>bar</foo>
```


Quiz

- Groups of 2-3: Find as many errors in this XML document as you can

```
<?xml version="1.0"?>
<books>
 <book id="1">
  <title>Data on the Web</title>
  <authors>
    <author id="a1">Abiteboul</author>
    <author id="a2">Buneman & </author>
    <author id='a3'>Suciu</authors>
  </author>
  <year>2000</year>
  <publisher>Addison-Wesley</publisher>
</books>
<foo>bar</foo>
```

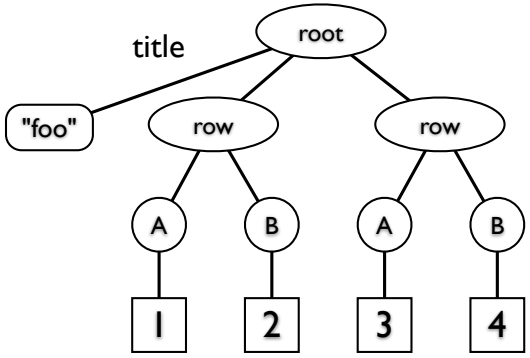
Processing XML

- Most programming languages have XML libraries
  - parsers (SAX, DOM)
  - in-memory manipulation (DOM)
  - validation (schemas)
- Thus, usually don't need to worry about all the fiddly details of character sets
  - XML supports UNICODE, many other encodings

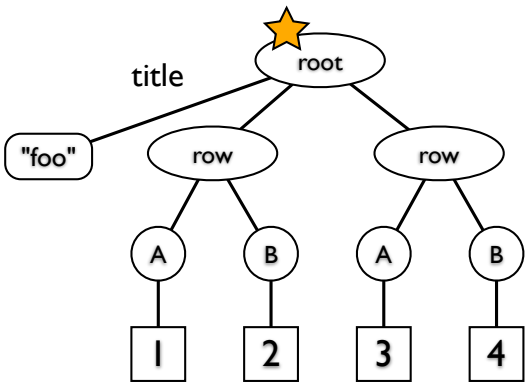
SAX: Basic idea

- SAX: Streaming API for XML (de facto standard)
  - reads XML document incrementally
  - generates calls to event handlers
  - you write code that handles these events

SAX: Example

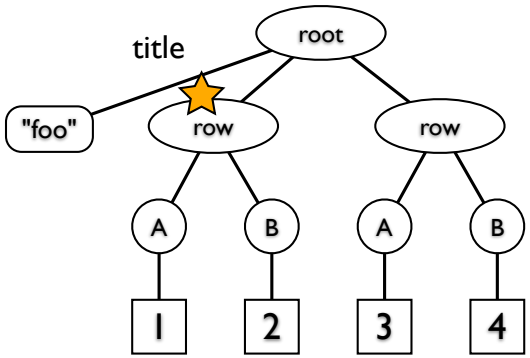


SAX: Example



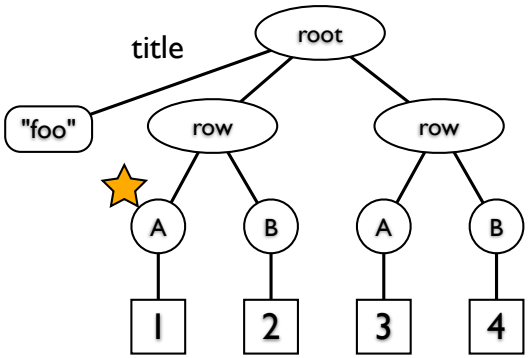

```
beginElt("root",[title="foo"])
```

SAX: Example



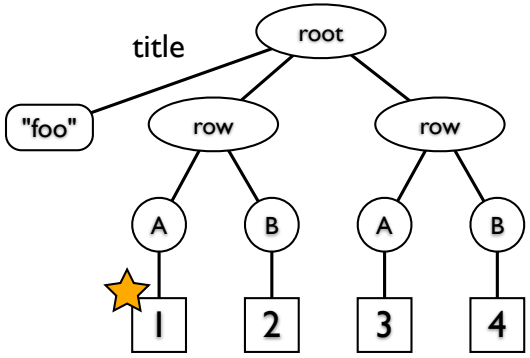

```
beginElt("root",[title="foo"])
 beginElt("row")
```


SAX: Example



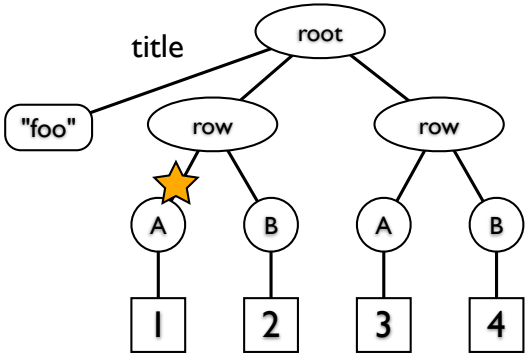

```
beginElt("root",[title="foo"])
 beginElt("row")
  beginElt("A")
```


SAX: Example



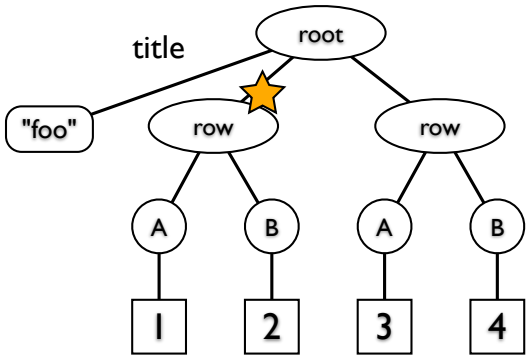

```
beginElt("root",[title="foo"])
 beginElt("row")
  beginElt("A")
   text("1")
```


SAX: Example



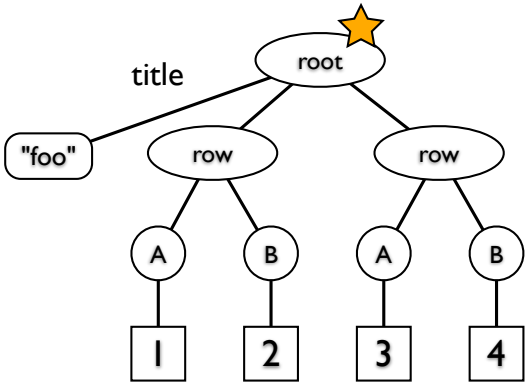

```
beginElt("root",[title="foo"])
 beginElt("row")
  beginElt("A")
   text("1")
  endElt("A")
```

SAX: Example




```
beginElt("root",[title="foo"])
 beginElt("row")
  beginElt("A")
   text("1")
  endElt("A")
  beginElt("B")
   text("2")
  endElt("B")
 endElt("row")
```


SAX: Example




```
beginElt("root",[title="foo"])
 beginElt("row")
  beginElt("A")
   text("1")
  endElt("A")
  beginElt("B")
   text("2")
  endElt("B")
 endElt("row")
 beginElt("row")
  beginElt("A")
   text("1")
  endElt("A")
  beginElt("B")
   text("2")
  endElt("B")
 endElt("row")
endElt("root")
```


SAX: Advantages

- Very widely supported
- Can be very efficient
  - for operations that are streaming-friendly
  - only realistic option for documents too large for memory
- Can easily ignore parts of the document
  - comments, etc.
  - these are still processed, however (SAX reads in whole XML file whether or not it is all needed)

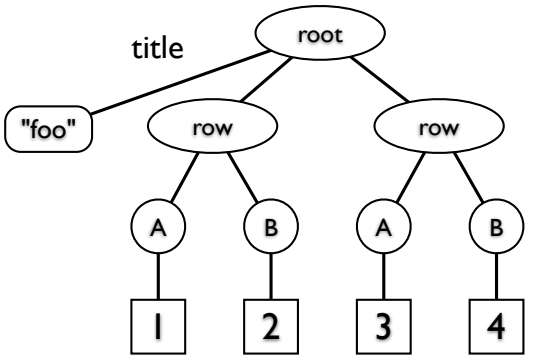
SAX: Disadvantages

- Non-starter if random access needed
- Not suitable for transformations to persistent or large data
  - e.g. in-browser or database updates
- Can be tricky to program
  - due to need to handle atomic events
  - need to figure out how to incrementalize processing & maintain state

DOM: Basic idea

- DOM: Document Object Model (W3C)
  - reads XML document all at once
  - allocates tree structure in memory
  - provides standard methods for traversing, modifying doc
  - widely used in JavaScript to dynamically update HTML page
- parses/loads document into memory

DOM: Example

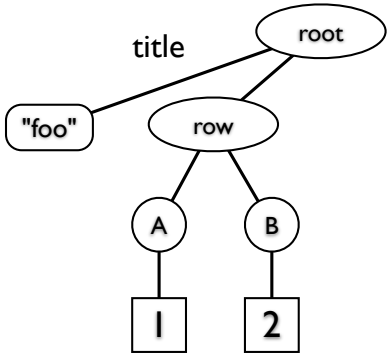


root = document.body

c1 = root.getFirstChild()

c2 = root.getLastChild()

DOM: Example



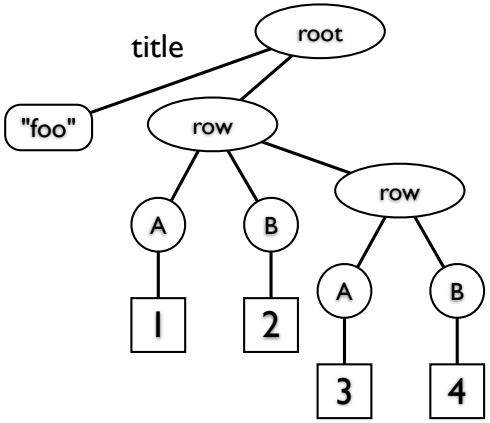
root = document.body

c1 = root.getFirstChild()

c2 = root.getLastChild()

root.deleteChild(c2)

DOM: Example



root = document.body

c1 = root.getFirstChild()

c2 = root.getLastChild()

root.deleteChild(c2)

c1.appendChild(c2)

DOM: Advantages

- Much easier to program
- Offers random access & dynamic updates
- Best if scalability to large data not a concern
- Library support for path queries can be very convenient (e.g. JQuery)
  - though naive implementations of queries can be very slow

DOM: Disadvantages

- Memory footprint can be several times that of XML text
  - which is already bloated!
- Thus, cannot be used for data > size of memory (gigabytes)
- At a programming level, side-effecting updates can be tricky to get right
  - but no realistic alternatives yet to JavaScript for browser interactivity

Next week

- XPath: Navigating through XML trees
- XQuery: SQL-like queries for constructing new XML trees from existing data