XML for Scientific Applications

Outline

- SQ1 returned
- History
- Manipulating XML
- XML for Science

XML History

- late 1960s
  - "Generic Coding" for elect. manuscripts
    - 'heading' rather than 'format-17'
- 1969
  - **G**oldfarb, **M**osher, **L**orie @ IBM
    - ***G**eneralized **M**arkup **L**anguage*
- 1980
  - SGML; *Standardized* by ANSI
  - IRS, DoD were early adopters

SGML

- Representation of documents
  - Device-Independent, Software-Independent
- Content
  - Tagged Text
- Structure
  - Document Type Definition (DTD)
- Style
  - Usually proprietary

SGML: Content

- Tagged Text
  - start tags and end tags
  - indentation not important

```
<section>
  <heading>SGML Example</heading>
  <paragraph>
    <line>first line of SGML</line>
    <line>second line of SGML</line>
  </paragraph>
</section>
```

SGML: Structure

- Dcoument Type Definition (DTD)

```
<!ELEMENT section - - (heading?, paragraph+)>
<!ELEMENT paragraph - O (line+ | quote)>
<!ELEMENT heading - O (#PCDATA) >
<!ELEMENT line - O (#PCDATA) >
<!ELEMENT quote - O (#PCDATA) >
```

A *section* is an optional *heading*, followed by one or more *paragraphs*.

A *paragraph* is one or more *lines* or a *quote*.

A *heading* is just text.

A *line* is just text.

SGML: Benefits and Weaknesses

- Benefits
  - Human-readable
  - Portable
  - Structure-oriented
- Weaknesses
  - DTD difficult to define up-front
  - Difficult to implement
    - syntax abbreviations/rules inference

AN SGML Application: HTML

- 1990
  - Tim Berners-Lee @ CERN
    - a Scientific Data Management problem…
  - writes a DTD for "Hypertext Markup Language"
- 1994-1996
  - Web takes off
  - Page designers need guarantees about how their page is going to look
  - <font> tags, etc. introduced; purists despair
  - MS and Netscape fight

Data on the Web

- <title>,<h1>,<font> are for humans
- Back to SGML for data, but
  - Stricter syntax (must use end tags)
    - good for implementors
  - looser semantics (DTD not required)
    - good for users
- Non-standard markup is better than no markup at all

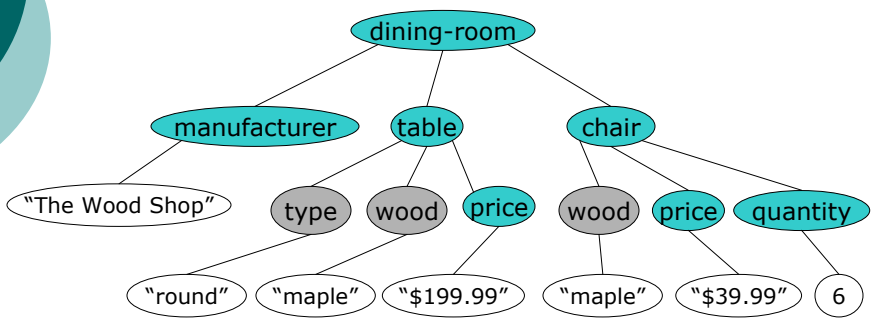
Growth

- Sun introduces Java
  - New hope for interoperability
- Microsoft pushes XML
  - because…it's not Java
- Individuals can publish data *unilaterally*
  - Contrast: How do you publish relations?
  - Contrast: Documents? (MS Word? as pdf?)
  - A *non-partisan*, *unimpeachable* format

XML File Sample

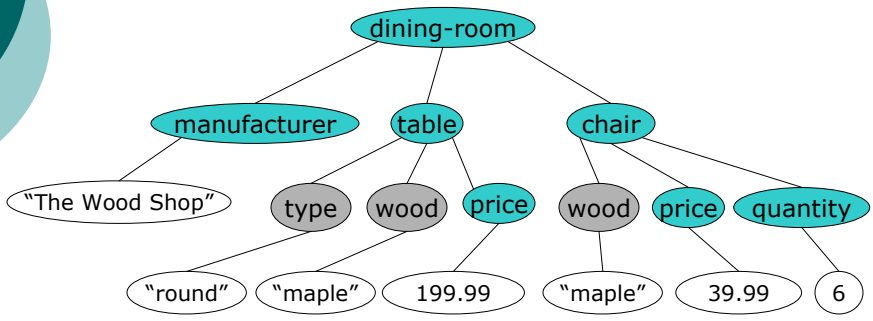
```
<?xml version="1.0"?>
<dining-room>
        <manufacturer>The Wood Shop</manufacturer>
        <table type="round" wood="maple">
                <price>$199.99</price>
        </table>
        <chair wood="maple">
                <quantity>6</quantity>
                <price>$39.99</price>
        </chair>
</dining-room>
```

Abstract View of XML



Manipulating XML

XPath



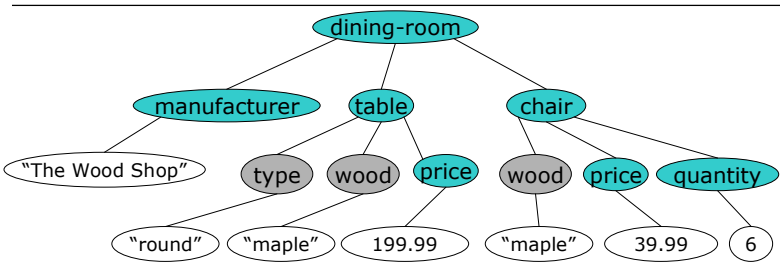
1) //price
2) /*/*/wood
3) //wood/../

4) /dining-room/*[price>100]/type
5) //wood[1]/text()

XPath

- XML → [Node]
  - language is not "closed"
  - cannot compose xpath expressions
- Answers are references to nodes
  - works for local memory…

XSLT

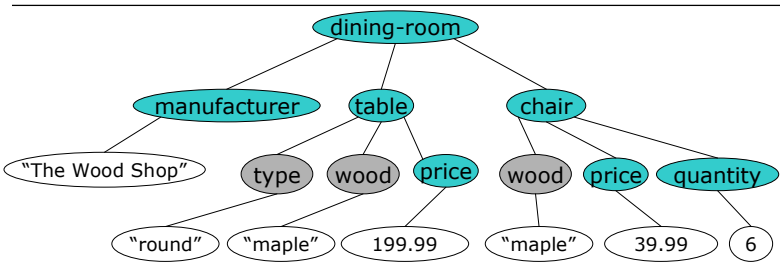



```
<xsl:stylesheet version="1.0" xmlns:xsl="…">
<xsl:template match="//wood">
        <xsl:if test="/price > 100">
                <material>
                        <xsl:value-of select="."/>
                </material>
        </xsl:if>
</xsl:template>
</xsl:stylesheet>
```

XSLT

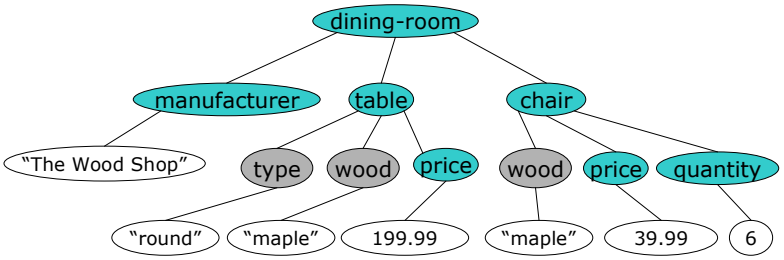
- logically, a set of rules
  - pattern1 -> result1
  - pattern2 -> result2
  - …
- physically, an XML document
  - why is this a good idea?
- XML → XML, via XML

XQuery




```
for $v in doc("example.xml")//wood
where $v/../price > 100
return
  <material>
        $v
  </material>
```

XQuery




```
for $m in doc("example.xml")//table[wood="maple"]
for $o in doc("example.xml")//table[wood="oak"]
where $m/price > $o/price
return
<pair>
        <maple>$m/type </maple>
        <oak>$m/type </oak>
</pair>
```

XQuery

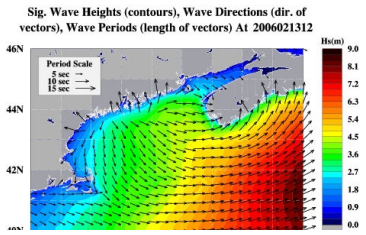
- Feels more like a Query language
- Joins are natural to express
- XML → XML, via special syntax

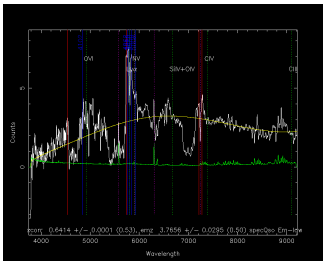
XSLT and XQuery

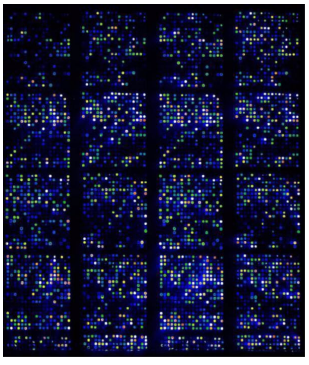
- Today's author sums up:
  - XSLT is for transformation
    - document-centric view of XML
  - XQuery is for query
    - data-centric view of world

XML for Science

- Exercise:
  - Sensor Data again







Sensor Data

- Would Homework 1 be any easier with XML?

Benefits for Science

- "better, more precise searching of full-content;
- automatic extraction of object metadata;
- better support for compound document formats and dynamic formatting generally; and,
- improved processes for scholarly dissemination tasks such as peer review, versioning, and archiving."

-- from NSF / NSDL Workshop on Scientific Markup Languages

XML for Science

- Recall features of Science Data:
  - Read-oriented access
  - Provenance
    - who, what, when, where, why
  - Interesting Data Types
    - timeseries
    - spatial
    - arrays
    - images
  - Scale

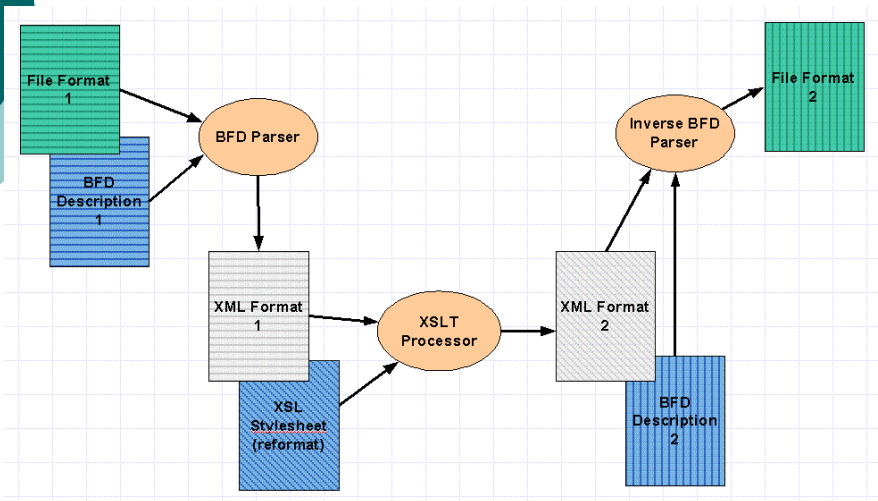
XML for Science

- Read-oriented access?
  - perfect!
- Provenance
  - requires some flexibility; no problem
- Interesting Data Types
  - …and special file formats
- Scale
  - could get ugly

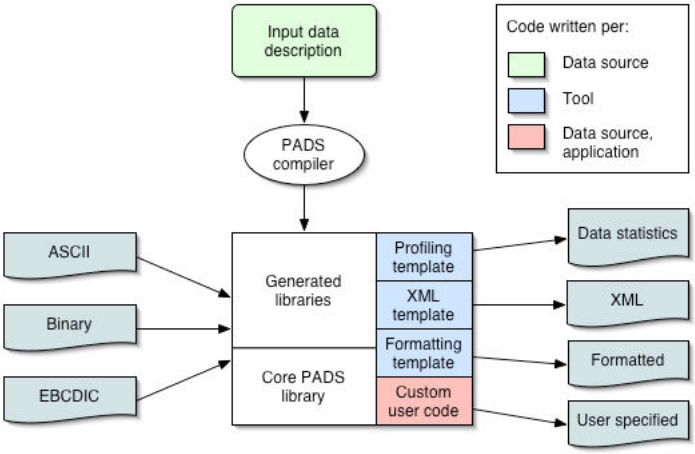
Interesting Data Types

- Data locked in binary file formats
  - **Binary Format Description Language**
    - [Myers, Chappell 2000]
  - **Data Format Description Language**
    - [OpenGrid Project]
  - **Retrofitting Data Models**
    - [Howe, Maier SSDBM 2005]
  - **PADX**
    - [Fernandez et al, PLANX 2006]
  - **XDTM**
    - [Foster, Voeckler et al. Global Grid Forum 2005]

Binary Format Description Language



PADX




```
1. Precord Pstruct summary_header_t {
2. "0|";
3. Punixtime tstamp;
4. };
5. Pstruct no_ramp_t {
6. "no_ii";
7. Puint64 id;
8. };
9. Punion dib_ramp_t {
10. Pint64 ramp;
11. no_ramp_t genRamp;
12. };
13. Pstruct order_header_t {
14. Puint32 order_num;
15. '|'; Puint32 att_order_num;
16. '|'; Puint32 ord_version;
17. '|'; Popt pn_t service_tn;
18. '|'; Popt pn_t billing_tn;
19. '|'; Popt pn_t nlp_service_tn;
```

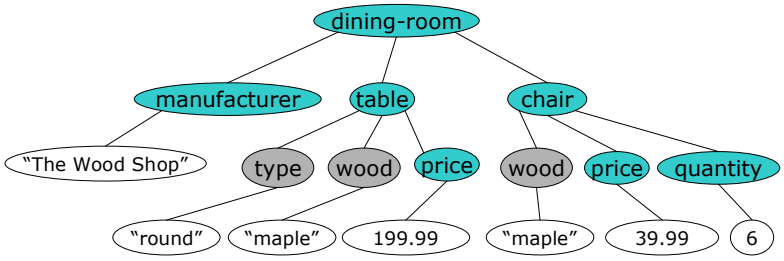
XML as Data

Maier's Maxim:

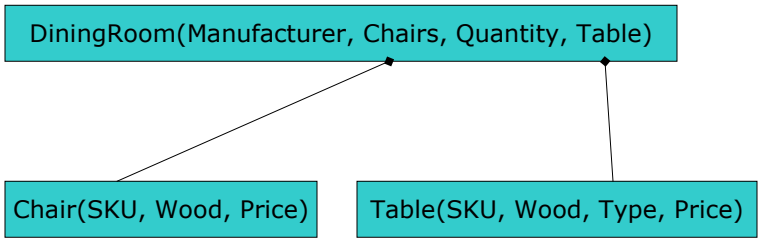
Every popular data exchange format eventually becomes a data storage format.

XML Storage: Shredding

- Use RDBMS as your storage engine
- Two approaches:
  - Schema-aware
  - Schema-oblivious



XML Storage: Schema-aware

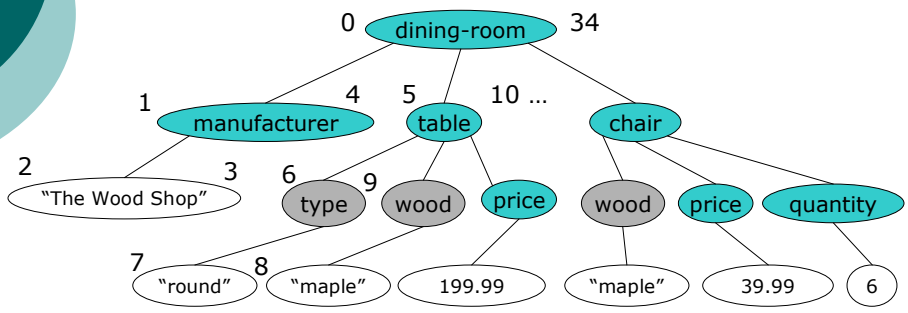


XML Storage: Schema-oblivious

Edge(NodeId, Tag, Value, ParentNodeId)

- Remember fancy node-labeling schemes…

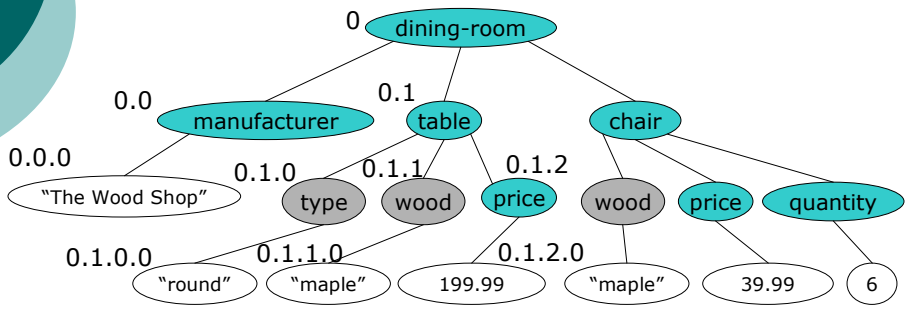
Left/Right Labeling



Which queries are easy and fast?

What did we say the problems were?

Path Labeling



What queries are fast and/or easy?

What did we say the problems were?

Next time

- Geographic Information Systems
- Spatial Data

- Study Questions 2
  - (I'll post them tonight)