Summary of Emerging Information & Knowledge Management Technologies

Richard Marciano

Reagan Moore

San Diego Supercomputer Center

April 17, 2001

Outline

~ 1:30 – 3:00

- **XML language core (README.FIRST)**
  - overview, the XML 1.0 Specification: syntax, namespaces, DTDs, ...

~ 3:30 – 5:00

- **Querying and transforming XML**
  - XPath, XQuery, XSLT, ...
- **Knowledge Management**
  - Semantic Web (RDF), Topic Maps, Knowledge-based data grids, ...

Overview

- XML is…
- XML for data exchange (messages) and persistent data
- XML syntax and data model
- XML DTDs
- Data Modeling
- Processing XML:
  - APIs (DOM, SAX)
  - addressing XML: XPath, XLink, XPointer

XML is …

- … an e**X**tensible **M**arkup **L**anguage
- … HTML – *presentation tags* + *your-own-tags*
- … a meta-language for defining other languages
- … a semistructured data model
- … not a data model but just an exchange syntax
- … the ASCII of the Web
- … many good (and some bad) Computer Science ideas reinvented (but now for the masses!)

…

Some History

- SGML (Standard Generalized Markup Language)
  - ISO Standard, 1986, for data storage & exchange
  - Metalanguage for defining languages (through DTDs)
  - A famous SGML language: HTML!!
  - Separation of content and display
  - Used in U.S. gvt. & contractors, large manufacturing companies, technical info. Publishers,…
  - SGML reference is 600 pages long
- XML (eXtensible Markup Language)
  - W3C (World Wide Web Consortium) -- http://www.w3.org/XML/ recommendation in 1998
  - Simple subset (80/20 rule) of SGML: "ASCII of the Web", "Semantic Web"
  - XML specification is 26 pages long

Emerging Trends

- Canonical XML
  - "normalization", equivalence testing of XML documents
- SML (Simple Markup Language)
  - "Reduce to the max": No Attributes / No Processing Instructions (PI) / No DTD / No non-character entity-references / No CDATA marked sections / Support for only UTF-8 character encoding / No optional features
- XML Schema
  - XML Schema definition language
  - Back to **complex**:
    - Part I (Structures), Part II (Data Types), Part III ooops: 0 (Primer)
- X-Zoo (Xoo?), "Brave New X-World"
  - Specifications **CSS** • **Digital Signatures** • **ebxml Project Teams** • **ebXML** • **IETF Specifications** • **Internationalization** • **IOTP (Internet Open Trading Protocol)** • **OASIS** • **Requirements Documents** • **SMIL** • **SVG (Scalable Vector Graphics)** • **Topic Maps** • **W3C Activity Pages** • **W3C Notes** • **W3C Standards** • **W3C Standards-in-progress** • **WAP** • **WebDAV** • **XHTML** • **XLink** • **XPath** • **XSLT**
  - Vocabularies **DTDs** • **Music** • **P3P** • **RDF** • **RSS** • **SMIL** • **W3C Standards** • **W3C Standards-in-progress** • **WML** • **XHTML** • **XSL FO's** • **XSLT** • **XUL**
  - Vertical Industries **Advertising** • **Commerce** • **Consortiums** • **Construction** • **Food** • **Insurance** • **Legal** • **Medical** • **Music** • **OASIS** • **Real Estate** • **Science** • **Space Exploration** • **Telecommunications** • **Travel** • **Weather**

Data Exchange with the Past

A time traveler sends a message in the virtual bottle, containing parts of the universal library of human and post-human mankind back into the last third of the 20th century…

- … when the Web, XML, WAP, B2B, supercomputing, wireless RX, and Petabytes were unheard of
- … RAM was so precious that it was ok to deal with nibbles
- … MS-DOS was called CP/M
- … and in fact Bill hadn't moved into the garage yet but worked on a homework assignment by Christos, trying to sort pancakes even faster (Gates, W.H. and Papadimitriou, C. "Bounds for Sorting by Prefix Reversal." *Discr. Math.* **27**, 47-57, 1979.)
- Task (in the past):
  - ***application programming & information exchange with the futuristic data***

Our past friend's **SUPERCOMPUTER** looked like this ...

```
62k CP/M VER 2.23 (Z80/DJDMA/VT100)

A>dir
A: ARK      COM : ASM      COM : CLS      COM : COPY     ASM
A: CPM2     HLP : CBIOS    ASM : CBOOT    ASM : DDT      COM
A: DDTZ     COM : DUMP     COM : ED       COM : EDFILE   COM
A: ERAQ     COM : FORMAT   ASM : FORMAT   COM : HELP     COM
A: HELP     HLP : LIB      COM : LINK     COM : LINK     HLP
A: LOAD     COM : LS       COM : LT       COM : LU       COM
A: LU       HLP : MAC      COM : MAC      HLP : MOUNT    ASM
A: MOVCPM   COM : PIP      COM : PTRDSK   ASM : PTRDSK   COM
A: PUTCPM   ASM : PUTCPM   COM : SAP      COM : SQ       COM
A: STAT     COM : SUBMIT   COM : SURVEY   COM : SYSGEN   SUB
A: THISSIM  HLP : UNARK    COM : UNCR     COM : UNERASE  COM
A: UNZIP    COM : USQ      COM : VDE      COM : XSUB     COM
A: MBASIC   HLP : MBASIC   COM : WS       HLP
A>mbasic
BASIC-80 Rev. 5.22
[CP/M Version]
32783 Bytes free
Ok
```

Ever wondered where those 8 letter filenames, 3 letter extensions came from? ;-)

Message in the Bottle (or: towards the Digital Rosetta Stone)

- Degree of "self-description":

not quite — *not bad* — *pretty good*

ĐÏ^Qà¡±^Zá^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@>^@^C^@þÿ^@^F^@^@^@^@^@^@^@^@^@^@^@^A^@^@^@#^@^@^@^@^@^@^@^@^@^P^@^@%^@^@^@^A^@^@^@þÿÿÿ^@^@^@^@"^@^@^@ÿÿìÿÁ^@q^@^D^@^@^@^R¿^@^@^@^@^@^@^P^@^@^@^@^@^D^@^@Ç^G^@^@^N^@bjbjt+t+^@^@^@

^@Some Quotations from the Universal Library^M1 Famous Quotes^M1.1 By William I^M[2, Sonnet XVIII]^MShall I compare thee to a summer's day?^MThou art more lovely and more temperate.^MRough winds do shake the darling buds of May,^MAnd summer's lease hath all too short a date.^MSometime too hot the eye of heaven shines,^MAnd often is his gold complexion dimmed.^MAnd every fair from fair some declines,^MBy chance or nature's changing course untrimmed.^MBut thy eternal summer shall not fade,^MNor lose possession of that fair thou owest,^MNor shall Death brag thou wander'st in his shade^MWhile in eternal lines to time thou growest.^MSo long as men can breathe, or eyes can see,^MSo long live this, and this gives life to thee.^M1.2 By William II^M[1, p.265]^M\223The obvious mathematical breakthrough would be development of^Man easy way to factor large prime numbers."^MReferences^M[1] W. H. Gates. The Road Ahead. Viking Penguin, 1995.^M[2] W. Shakespeare. The Sonnets of Shakespeare.609.^M^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@

```
\documentclass{article}
 \begin{document}
 \title{Some Quotations from the Universal
        Library}
...
\section{Famous Quotes}
 \subsection{By William I}
 \textbf{\cite[Sonnet XVIII]{shakespeare-
        sonnets-1609}}
   \begin{verse}
    Shall I compare thee to a summer's day?\\
    Thou art more lovely and more temperate.
        \\
    Rough winds do shake the darling buds of
        May, \\
    And summer's lease hath all too short a
        date. \\
    Sometime too hot the eye of heaven shines,
        \\
    And often is his gold complexion dimmed. \\
  ...
    \qquad So long as men can breathe, or eyes
        can see,\\
    \qquad So long live this, and this gives life
        to thee.   \\
  \end{verse}
...
 \bibliographystyle{abrv}
\bibliography{msg}

\end{document}
```

```
<?xml version="1.0"?>
<universal_library>
  <books>
    <book> <title>Some Quotations from the Universal
           Library</title>
      <section> <title>Famous Quotes</title>
        <subsection>  <title>By William I</title>
          <quote bibref="shakespeare-sonnets-1609">
          <title>Sonnet XVIII</title>
          <verse>
            <line>Shall I compare thee to a summer's
          day?</line>
            <line>Thou art more lovely and more temperate.
          </line>
            <line>Rough winds do shake the darling buds of May,
          </line>
          </verse>
   ...
      <subsection> <title>By William II</title>
        <quote bibref="gates-road-ahead-1995">
          <title>Page 265</title>
          <line>``The obvious mathematical breakthrough would
          be development of an easy way to factor large prime
          numbers.''</line>
        </quote>
      </subsection>
      </section>
</book>
...
</books>
</universal_library>
```

HTML vs. XML

HTML tags:
presentation, generic document structure

```
<h1> Bibliography </h1>
<p> <i> Foundations of DBs</i>, Abiteboul, Hull, Vianu
    <br> Addison-Wesley, 1995
<p> <i> Logics for DBs and ISs </i>, Chomicki, Saake, eds.
    <br> Kluwer, 1998

<bibliography>
    <book>  <title> Foundations of DBs </title>
            <author> Abiteboul </author>
            <author> Hull </author>
          <author> Vianu </author>
            <publisher> Addison-Wesley </publisher>
            ....
    .</book>
    <book>  ... <editor> Chomicki </editor>... </book>    ...
</bibliography>
```

XML tags:

content, "semantic", (DTD-) specific

XML vs SGML

- origins: HTML + SGML (ISO Standard, 1986, ~600pp)
- W3C standard (~26 pp): XML syntax + DTDs
- XML = HTML – presentational tags
  + user-defined DTD (tags+nesting)

=> really a metalanguage for defining other languages via DTDs

=> XML is more like SGML than HTML

- XML = SGML – {complexity, document perspective}
  + {simplicity, data exchange perspective}

XML as a **Self-Describing** Data Exchange Format

- can be easily “understood” by our friend (... even using CP/M & edlin)
- can be parsed easily
- contains its own structure (=parse tree) in the data

=> allows the application programmer to rediscover schema and content/semantics (to which extent???)

- may include an explicit schema description (e.g., DTD)

=> meta-language: definition of a language w.r.t. which it is valid

- allows separation of marked-up content from presentation (=>style sheets)

- many tools (and many more to come -- (re)use code): parsers, validators, query languages, storage, ...
- standards (good for interoperation, integration, etc):
  - => generic standards (XML, DTDs, XML Schema, XPath,...)
  - => community/industry standards (=specific markup languages)

Different Perspectives on XML

- Document (SGML) Community
  - data = linear text documents
  - mark up (annotate) text pieces to describe context, structure, semantics of the marked text
- Database Community
  - XML as a (most prominent) example of the semistructured data model

  => captures the whole spectrum from highly structured, regular data to unstructured data (relational, object-oriented, HTML, marked up text, ...)

More (Partisan) Perspectives on XML

- *"XML is the cure for your data exchange, information integration, e-commerce, [x-2-y, U name it] problems"*
  ("snake oil", "silver bullet ")

- *"XML is just (another) syntax (for Lisp, trees,...)"*
  ("nothing new under the sun")

```
(books (book (author "Shakespeare" )
             (title "Sonnets")
             (verse (line "Shall I compare thee..." )
                    (line ...)   ...)))
```

Many X-cellent(?) Acronyms...

- XML (Extensible Markup Language)
- XML Namespaces
- XML DTDs, XML Schema
- RDF (Resource Description Framework)
- XSL (Extensible Style Sheet Language)
- XPath (=XSLT∩ XPointer), XLink
- XQL, XML-QL (XML Query Language), XQuery
- XMAS (XML Matching And Structuring language)
- eXcelon, ...

=> XML++ (i.e. += X-tensions), so more than just syntax

=> a family of technologies (extensions, tools, ...)

=> generic standards and industry/community standards

XML Applications & Industry Initiatives

http://www.oasis-open.org/cover/xml.html#applications

- Advertising: adXML **place an ad onto an ad network or to a single vendor**
- Literature: Gutenberg **convert the world's great literature into XML**
- Directories: dirXML **Novell's Directory Services Markup Language (DSML)**
- Web Servers: apacheXML **parsers, XSL, web publishing**
- Travel: openTravel **information for airlines, hotels, and car rental places**
- News: NewsML **creation, transfer and delivery of news**
- Human Resources: XML-HR **standardization of HR/electronic recruiting XML definitions**
- International Dvt: IDML **improve the mgt. and exchange of info. for sustainable development**
- Voice: VoxML **markup language for voice applications**
- Wireless: WAP **(Wireless Application Protocol) wireless devices on the World Wide Web**
- Weather: OMF **Weather Observation Markup Format (simulation)**
- Geospatial: ANZMETA **distributed national directory for land information**
- Banking: MBA **Mortgage Bankers Association of America --> credit report, loan file, underwriting...**
- Healthcare: HL7 **DTDs for prescriptions, policies & procedures, clinical trials**
- Math: MathML **(Mathematical Markup Language)**
- Surveys: DDI **(Data Documentation Initiative) "codebooks" in the social and behavioral sciences**

XML E-commerce Initiatives

- CommerceNet
  - **eCo Framework** XML specs. to support interoperability among e-businesses
  - **Commerce One** Common Business Library (CBL): set of business components, docs. In DTD, XDR, SOX
  - **BizTalk** Microsoft spec. based on XML schemas
  - **cXML** (Commerce XML) -- tag-sets for e-procurement into BizTalk
- Electronic Data Interchange (EDI)
  - **RosettaNet** Common format for online ordering
  - **FpML** (Financial products Markup Language): sharing of financial data (interest rate & foreign exchange products)
- Open Buying on the Internet (OBI)
  - **OBI** high volume b2b purchasing transactions over the Internet (Office Depot, Lockheed, barnesandnoble, AX...
- E-commerce and XML
  - **VISA Invoices** The Visa Extensible Markup Language (XML) Invoice Specification provides a comprehensive list of data elements contained in most invoices, including: **Buyer/Supplier, Shipping, Tax, Payment, Currency, Discount, and Line Item Detail.**
- B2B Integration
  - **code360** XML-Broker is middleware software that manages XML based transactions
  - **Bluestone XML Suite** Enables to develop and deploy e-commerce, electronic data interchange, application integration and supply chain management applications. Bluestone XML Suite products include: XML-Server, Visual-XML, XML-Contact and XwingML.
  - **webMethods** Provides companies with integrated direct links to buyers and suppliers
- Business-Process Modeling
  - **BPML** Business Process Modeling Language, an XML-Schema from http://www.bpmi.org
- Business Directory Services
  - **UDDI** Universal Description, Discovery and Integration

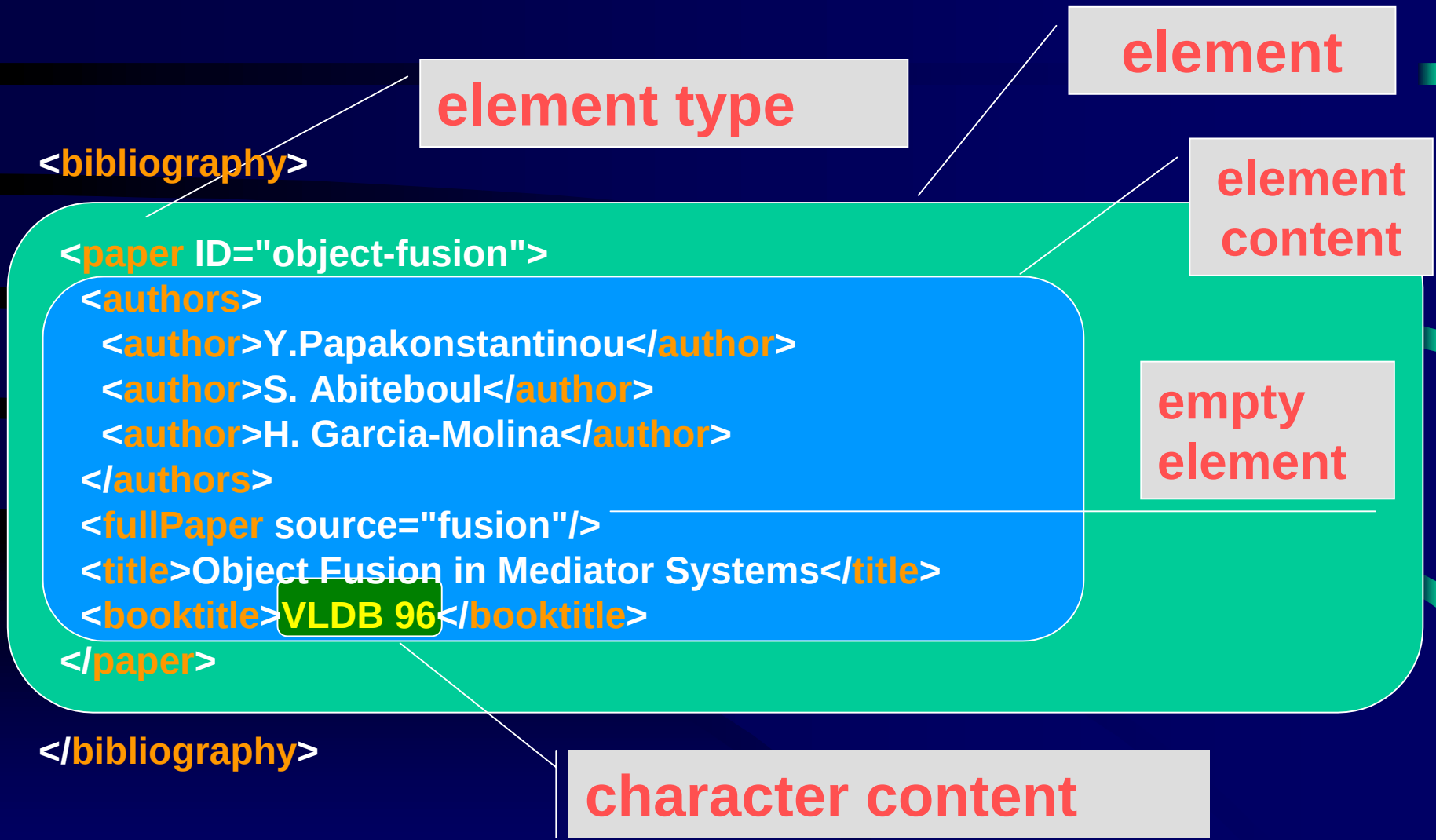
XML is Based on Markup

Markup indicates structure (and semantics!?)

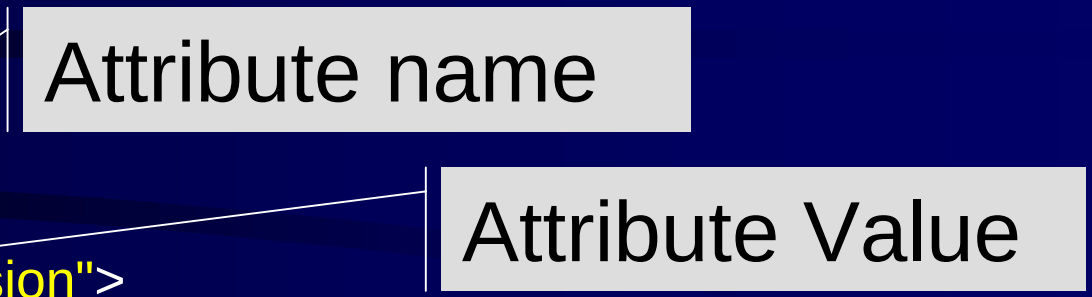
```
<bibliography>

 <paper ID= "object-fusion">
  <authors>
   <author>Y.Papakonstantinou</author>
   <author>S. Abiteboul</author>
   <author>H. Garcia-Molina</author>
  </authors>
  <fullPaper source="fusion"/>
  <title>Object Fusion in Mediator Systems</title>
  <booktitle>VLDB 96</booktitle>
 </paper>

</bibliography>
```

Decoupled from presentation

Elements and their Content



Element Attributes

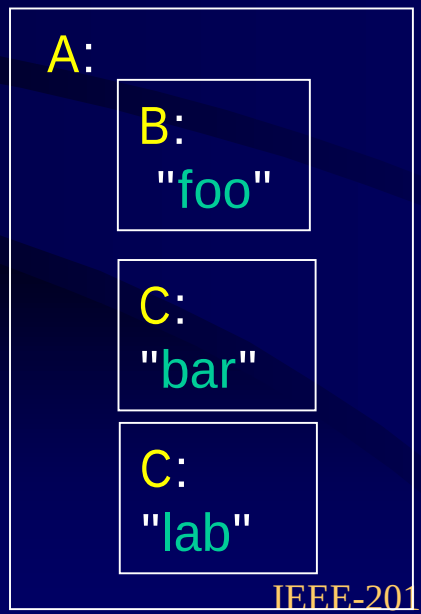


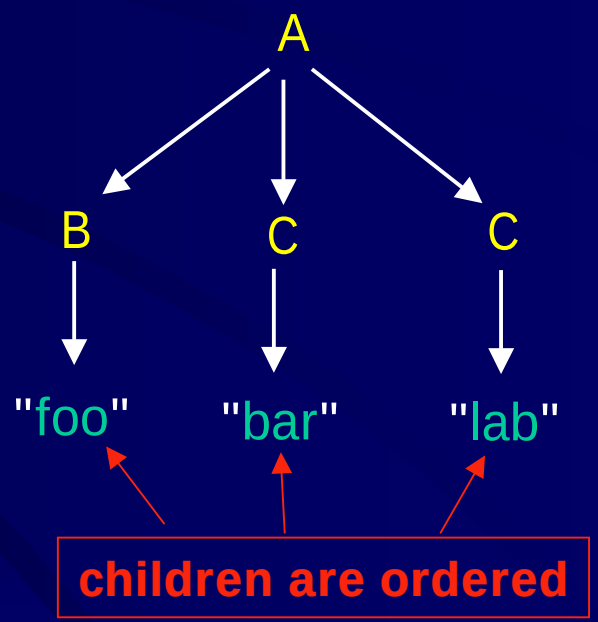

```
<bibliography>

  <paper pid="object-fusion">
    <authors>
      <author>Y.Papakonstantinou</author>
      <author>S. Abiteboul</author>
      <author>H. Garcia-Molina</author>
    </authors>
    <fullPaper source="fusion"/>
    <title>Object Fusion in Mediator Systems</title>
    <booktitle>VLDB 96</booktitle>
  </paper>

</bibliography>
```

Pure XML -- Instance Model

- XML 1.0 Standard:
  - no explicit data model
  - only syntax of well-formed and valid (wrt. a DTD) documents
- implicit data model:
  - nested containers ("boxes within boxes")
  - labeled ordered trees (=a semistructured data model)
  - relational, object-oriented, other data: easy to encode

```
<A>
  <B>foo</B>
  <C>bar</C>
  <C>lab</C>
</A>
```





In Search of the Lost Structure & Semantics

How do I learn and use the **element structure** of a document?

How do I **share structure** and **metadata/semantics** with my community?

How to make all this **automatable**?

Adding Structure and Semantics

- XML Document Type Definitions (DTDs):
  - define the structure of "allowed" documents (i.e., *valid wrt. a DTD*)
  - ≈ database schema

  => improve query formulation, execution, ...
- XML Schema
  - defines structure and data types
  - allows developers to build their own libraries of interchanged data types
- XML Namespaces
  - identify your vocabulary

XML DTDs as Extended Context Free Grammars

XML DTD

```
<!element bibliography paper*>
<!element paper        (authors,fullPaper?,title,booktitle)>
<!element authors      author+>
```

Grammar

bibliography ⟶ paper*
paper ⟶ authors fullPaper? title booktitle
authors ⟶ author+

lhs = element (name)
*rhs = **regular expression** over elements + strings (PCDATA)*

Document Type Definitions (DTDs)

Define and Constrain
Element Names & Structure

```
<!element bibliography paper*>
<!element paper (authors, fullPaper?, title, booktitle)>
<!element authors author+>
<!element author (#PCDATA)>
<!attlist author age CDATA>

<!element fullPaper EMPTY>
<!element title (#PCDATA)>
<!element booktitle (#PCDATA)>
```

Element Type Declaration

Attribute List Declaration

Element Declarations

Sequence of 0 or more papers

Authors followed by optional fullpaper, followed by title, followed by booktitle

```
<!element bibliography paper*>
<!element paper (authors, fullPaper?, title, booktitle)>
<!element authors author+>
<!element author (#PCDATA)>
<!attlist author age CDATA>

<!element fullPaper EMPTY>
<!element title (#PCDATA)>
<!element booktitle (#PCDATA)>
```

Sequence of 1 or more authors

Character content

Element Content Declarations

Declaration	*Meaning*
<element 2>	Exactly one <element 2>
cardinality: R?	Zero or one instances of R
R*	Zero or more instances of R
R+	One or more instances of R
$R_1\|R_2\|\ldots\|R_n$	One instance of R_1 or R_2 or … R_n
$R_1, R_2, \ldots, R_n$	Sequence of R's, order matters
#PCDATA	Character content
EMPTY	Empty element
(#PCDATA e*)*	Mixed Content
ANY	Anything goes

Attribute Types (DTD)

Type	*Meaning*
ID	**Token unique within the document**
IDREF	**Reference to an ID token**
IDREFS	**Reference to multiple ID tokens**
ENTITY	**External entity (image, video, …)**
ENTITIES	**External entities**
CDATA	**Character data**
NMTOKEN	**Name token**
NMTOKENS	**Name tokens**
NOTATION	**Data other than XML**
Enumeration	**Choices**
Conditional Sec	**INCLUDE & IGNORE declarations**

Attributes may be: REQUIRED, IMPLIED (optional)

can have: default values, which may be FIXED

Attribute Declarations

```
<!element bibliography paper*>
<!element paper (authors, fullPaper?, title, booktitle)>
<!element authors author+>
<!element author (#PCDATA)>

<!element fullPaper EMPTY>
<!element title (#PCDATA)>
<!element booktitle (#PCDATA)>
<!attlist fullPaper source ENTITY #REQUIRED>
<!attlist person pid ID>
<!attlist author authorRef IDREF>
```

Pointer (IDREF) and target (ID) declarations for intradocument “pointers”

XML **Attributes**

```
<person pid=”joyce">    …     </person>

<bibliography>

 <paper pubid="wsa" role="publication">

  <authors>
   <author authorRef=”joyce” age=“???”>
                    J. L. R. Colina </author>
         </authors>
  <fullPaper source="http://...confusion"/>
  <title>Object Confusion in a Deviator System </title>
  <related papers= "deviation1[illegible]x_deviators"/>
 </paper>

</bibliography>
```

Object Identity Attribute

CDATA (character data)

IDREF intradocument reference

Reference to external ENTITY

Uses of XML Entities

- Physical partition
  - size, reuse, "modularity", ... (both XML docs & DTDs)
- Non-XML data
  - unparsed entities → binary data
- Non-standard characters
  - character entities
- Shorthand for phrases & markup,

=> effectively are macros

Types of Entities

- Internal (to a doc) vs. External ($\rightarrow$ use URI)
- General (in XML doc) vs. Parameter (in DTD)
- Parsed (XML) vs. Unparsed (non-XML)

Internal Text Entities

DTD

Internal Text Entity Declaration

```
<!ENTITY WWW "World Wide Web">
```

XML

Entity Reference

```
<p>We all use the &WWW;.</p>
```

Logically equivalent to actually appearing

```
<p>We all use the World Wide Web.</p>
```

Entities & Physical Structure

A logical element can be split into multiple physical entities

External Text Entities

DTD

External Text Entity Declaration

```
<!ENTITY chap1 SYSTEM "http://...chap1.xml">
```

URL

XML

Entity Reference

```
<mylife> &chap1; &chap2;</mylife>
```

Logically equivalent to inlining file contents

```
<mylife> <teen>yada yada</teen>
                <adult> blah blah</adult>
</mylife>
```

Unparsed (& "Binary") Entities

DTD

... and unparsed entity

Declare external...

```
<!ENTITY fusion SYSTEM "http://... fusion.ps" NDATA ps>
```

Declare attribute type to be entity

```
<!attlist fullPaper source ENTITY #REQUIRED>
```

XML

Element with ENTITY attribute

```
<fullPaper source="fusion"/>
```

NOTATION declaration (helper app)

```
<!NOTATION ps SYSTEM "ghostview.exe">
```

Pure XML Model (DTD)

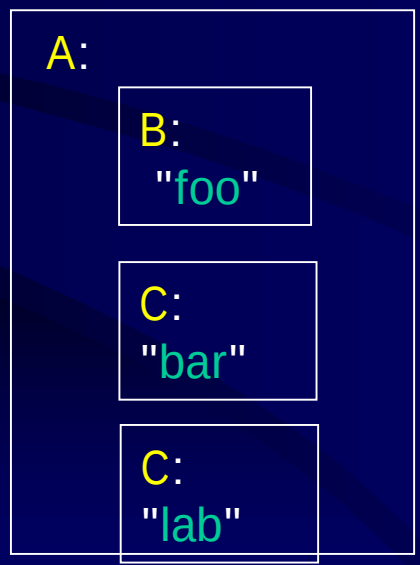
- Any DTD *myDTD* defines a language valid(*myDTD*):
  valid(*myDTD*) = {docs *D* | *D* is valid wrt. *myDTD*}

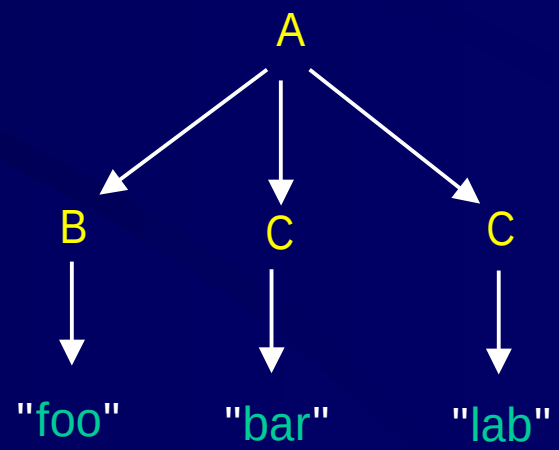
- <!ELEMENT A (B,C*)>
- <!ELEMENT B (#PCDATA)>

Content ("container") model: A contains one B, followed by any number of Cs

B is a leaf, contains actual data

```
<A>
  <B>foo</B>
  <C>bar</C>
  <C>lab</C>
</A>
```





From Documents to Data: Example

Document-Oriented:

```
<memo importance='high' date='1999-03-23'>
<from>Paul V. Biron</from> <to>Ashok Malhotra</to>
<subject>Latest draft</subject>
<body> We need to discuss the latest draft
<emph>immediately</emph>. Either email me at
<email> mailto:paul.v.biron@kp.org</email> or call
<phone>555-9876</phone>
 </body> </memo>
```

Data-Oriented:

```
<invoice>
<orderDate>1999-01-21</orderDate>
<shipDate>1999-01-25</shipDate>
 <billingAddress>
<name>Ashok Malhotra</name>
<street>123 IBM Ave.</street>
<city>Hawthorne</city> <state>NY</state>
<zip>10532-0000</zip>
</billingAddress>
<voice>555-1234</voice>
<fax>555-4321</fax>
</invoice>
```

Data Modeling with DTDs

- XML element types ~ "object types"
- content model for children elements ~ "subobject structure"
- recursive types (container analogy!?)

  <!ELEMENT A (B|C)> *"an A can contain a B..."*

  <!ELEMENT B (A|C)> *"... which contains an A!"*

  <!ELEMENT C (#PCDATA)>

  – found in doc world: document **DIV**ision (=generic block-level container)
- loose typing

  – <!ELEMENT A ANY> *"so what's in the box, please??"*
- no context-sensitive types:

  DTDs cannot distinguish between the publisher in

  – <journal> <publisher>... </publisher> </journal>

  – <website> <publisher> ... </publisher> </website>

  => renaming “hack” <j_pub> and <w_pub>

  => DTD extensions (XML SCHEMA)

Where is the Data??

- Actual data can go into leaf elements and/or attributes

- Common/good practice (!?):
  - XML element ~ container (object)
  - XML element type (tag) ~ container (object) type
  - XML attribute ~ properties of the container as a whole ("metadata")
  - XML leaf elements ~ contain actual data

- Problems with DTDs:
  - no data types
  - no specialization/extension of types
  - no "higher level" modeling (classes, relationships, constraints, etc.)

Extending DTDs: Data Modeling Approaches

- XML main stream: XML Schema
  - data types
  - user defined types, type extensions/restrictions ("subclassing")
  - cardinality constraints
- XML side streams:
  - RELAX (REgular Language description for XML), SOX (Schema for Object-Oriented XML), Schematron, ...
- alternative approach:
  - use well-established data modeling formalisms like (E)ER, UML, ORM, OO models, ...

... and just encode them in XML!

  - e.g. UML: XMI (standardized, has much more=>big), UXF (UML eXchange Format)

XML-Extensions as **Constraint Languages**

(a unifying perspective on XML schema-languages)

- XML schema languages (DTD, XML Schema, RELAX, RDF-Schema, ...) act as constraint languages *CL*, separating "good" (=valid) from "bad" (=invalid) documents

- EXAMPLE: *CL*={XML DTDs}, constraint *c* (in *CL*) = BioML-DTD
  => valid(*c*) = all valid BioML XML documents
  = *the BioML language!!??*
  => valid(*CL*) = all languages that can be captured using *CL*

- PROBLEM: DTDs capture only the **structural aspect** of BioML (i.e., allowed names, nesting, multiplicity of tags)
  => no datatypes, no other BioML semantics
  => specialized validators (for BioML, GeoML, ...)
  ... or generic validators for more expressive constraint languages (XML Schema, ...)

Identifying Vocabularies: XML Namespaces

- My element may not be your element:
  - geometry context: <element>line</element>
  - chemistry context: <element>oxygen</element>
  - SGML/XML context: ....

⇒ use XML namespaces to identify the vocabulary

XML Namespaces

- mechanism for globally unique tag names:

```
<h:html        xmlns:xdc="http://www.xml.com/books"
               xmlns:h="http://www.w3.org/HTML/1998/html4">
 <h:head><h:title>Book Review</h:title></h:head>
 ...
 <xdc:bookreview>
  <xdc:title>XML: A Primer</xdc:title>
 ...
</h:html>
```

⇒ mix of different tag vocabularies without confusion

- namespaces only identify the vocabulary; additional mechanisms required for structure and meaning of tags

Processing XML

- Non-validating parser:
  - checks that XML doc is syntactically well-formed
- Validating parser:
  - checks that XML doc is also valid w.r.t. a given DTD
- Parsing yields tree/object representation:
  - Document Object Model (DOM) API
- Or a stream of events (open/close tag, data):
  - Simple API for XML (SAX)

DOM Structure Model and API

- hierarchy of Node objects:
  - document, element, attribute, text, comment, ...

- language independent programming DOM API:
  - get... first/last child, prev/next sibling, childNodes
  - insertBefore, replace
  - getElementsByTagName
  - ...

- alternative event-based SAX API (Simple API for XML)
  - does not build a parse tree (reports events when encountering begin/end tags)
  - for (partially) parsing very large documents

DOM Summary

- Object-Oriented approach to traverse the XML node tree
- Automatic processing of XML docs
- Operations for manipulating XML tree
- Manipulation & Updating of XML on client & server
- Database interoperability mechanism
- Memory-intensive

SAX Event-Based API

- Pros:
  - The whole file doesn't need to be loaded into memory
  - XML stream processing
  - Simple and fast
  - Allows you to ignore less interesting data
- Cons:
  - limited expressive power (query/update) when working on streams

  => application needs to build (some) parse-tree when necessary

XML Information Set (XIS)

- W3C Working Draft, July 2000
- describes information content as "seen" by XML processors
- Example:

```
<?xml version="1.0"?>
 <msg:message doc:date="19990421"
      xmlns:msg="http://www.message.example/"
      xmlns:doc=http://www.doc.example/namespaces/doc
>
          Phone home!

</msg:message>
```

- A document information item.
- An element information item with the namespace name "http://www.message.example/" and the local part " message".
- An attribute information item with the namespace name "http://www.doc.example/namespaces/doc" and the local part "date".
- Two namespace information items for the http://www.doc.example/namespaces/doc and http://www.message.example/ namespaces.
- Eleven character information items for the character data, eight character information items for the attribute value, and 64 more for the namespace declarations.

Querying XML

- Different XML QL paradigms depending on the community:
  - (relational, oo, semistructured) database perspective
    - Lorel, YaTL, XML-QL, XMAS, FLORA/FLORID, ...
  - document processing perspective
    - XQL, XSL(T), XPath, ...
  - functional programming perspective
    - QLs with structural recursion, ...
- Patching desirable features together: XQuery

Important QL Features (DB Perspective)

- typical parts of a query:
  - (match) **pattern** (selects parts of the source XML tree without looking at data)
  - **filter** condition (selects further, now looking at the data)
  - answer **construction** (putting the results together, possibly reordered, grouped, etc.)
- reordering based on nested queries, grouping, sorting, or Skolem functions
- tag variables, path expressions for defining the patterns without requiring knowledge of the DTD

XML Path Language: XPath

- W3C Recommendation Nov. 1999
- for addressing parts within an XML document
- (non-XML) syntax used for XSLT and XPointer

- Find the root element (bookstore) of this document:
- /bookstore
- Find all author elements anywhere within the current document:
- //author

More Selection Queries with Path

- Find all books where the value of the style attribute on the book is equal to the value of the specialty attribute of the bookstore element at the root of the document:

- //book[/bookstore/@specialty = @style]

- Find all books with author/first-name equal to 'Bob' and all magazines with price less than 10:

- // (book[author/first-name = 'Bob']
  $union$ magazine[price lt 10])

XML Pointer Language (XPointer)

- W3C Candidate Recommendation, June/2000
- for locating internal structures of XML documents
- XLinks URIs can include XPointer parts
- extends HTML's named anchors:
  - target doc: ...
  - source doc: ...
- ... and select via XPath expressions

  + some extension (*points* and *ranges*, ...)

Example:

- intro/14/3 ("intro" is an ID attribute value)
- /1/2/5/14/3
- xpointer(id("chap1")) xpointer(//*[@id="chap1"])

XML Linking Language (XLink)

- W3C Candidate Recommendation, July/2000
- language for typed links between documents
- extends the simple untyped href links in HTML:
  - multidirectional links
  - any element can be the source (not just <a ... >)
  - link to arbitrary positions within a document (via URIs and XPointer)
- richer custom applications possible
- xlink:type declaration: {simple, extended, locator, arc}
- optional "semantic attributes": {role, arcrole, title}
- Example:

```
<author   xmlns:xlink="... "
          xlink:href="....itmaven.com/peter.html"
          xlink:title="Peter's   homepage"
          xlink:role="further info about the book author"
> Peter Pan Sr. </author>
```

References

- W3C Standards: w3.org
- XML portal (news, resources, ...): xml.com
- Meta:
  - {google,yahoo,...} to {"xml", "dtd", ...}

Querying and Transforming XML

Overview

- Querying XML
  - from walking the XPath to
  - making the XQuery
- Transforming XML: XSLT
- Demonstrations:
  - XML queries and transformations

Querying XML

- Different XML QL paradigms depending on the community:
  - (relational, oo, semistructured) database perspective
    - Lorel, YaTL, XML-QL, XMAS, FLORA/FLORID, ...
  - document processing perspective
    - XQL, XSL(T), XPath, ...
  - functional programming perspective
    - QLs with structural recursion, ...
- Patching desirable features together: XQuery

Querying XML

- No "official" W3C XML QL yet (but bits and pieces)
- numerous quite different XML QLs are popping up
- some XML QL overviews, comparisons, and resources:
  - XML Query Languages: Experiences and Exemplars **(co-authored by several XML QL gurus)**
  - XML and Query Languages **(Oasis Cover Pages)**
  - Comparative Analysis of Five XML Query Languages **(A. Bonifati, S. Ceri)**
  - A Data Model and Algebra for XML Query **(Philip Wadler et.al. "functional (Haskell) perspective")**
  - XML-QL vs XSLT queries **(Geert Jan Bex and Frank Neven; for (future) XSLT experts only ;-)**
  - Introduction to XMAS **(the XML QL of the MIX project)**

Important QL Features (DB Perspective)

- typical parts of a query:
  - (match) **pattern** (selects parts of the source XML tree without looking at data)
  - **filter** condition (selects further, now looking at the data)
  - answer **construction** (putting the results together, possibly reordered, grouped, etc.)
- reordering based on nested queries, grouping, sorting, or *Skolem functions*
- tag variables, path expressions for defining the patterns without requiring knowledge of the DTD

An XML Query (XMAS @ SDSC/UCSD)

```
$C:<*.condo>
       <address zip=$Z/>
   </condo> AT www.condo.com
 AND
$S:<*.school type=elementary>
       <address zip=$Z/>
   </school> AT schools.org
```

```
<folder>
 $C
 $S   {$S}
</folder> {$C}
```

```
<RealEstateAgent>
 <name>J. Smith</name>
 <condos>
   <condo>
     <address ... zip=92037>
     <price>$170k OBO</price>
     <bedrooms>2</bedrooms>
   </condo>
 <condos>
</RealEstateAgent>
```

```
<condosAndSchools>
 <folder>
   <condo>
     <address ... zip=92037>
     <price>$170k OBO</price>
     <bedrooms>2</bedrooms>
   </condo>
   <school>
    <name>La Jolla High</name>
    <address … zip=92037>
   </school>
   <school>…</school>
 </folder>
```

Quilt (pre-Xquery) (Chamberlin, Robie, Florescu)

Q: "find every book written by Crockett Johnson"

```
<Result> (
FOR     $b IN document("http://www.biblio.com/books.xml")//book,
        $a IN $b/author
WHERE $a/firstname = "Crockett"
   AND $a/lastname = "Johnson"
RETURN $b
) </Result>
```

Q: as above, but "inverted"

```
<Result> (
FOR     $a IN DISTINCT document("http://www.biblio.com/books.xml")//author RETURN
   <BooksByAuthor>
        <Author> $a/lastname/text() </Author>
        ( FOR $b IN document("http://www.biblio.com/books.xml")//book[author=$a]
        RETURN $b/title SORTBY(title) )
   </BooksByAuthor> SORTBY(Author)
) </Result>
```

XQuery

- Data model:
  - the XML Query working group data model
- Language description:
  - borrows features from OQL, XML-QL, LoreL, XQL, ML.
  - as ML, OQL, Lorel: it is a functional language
  - includes a subset of Xpath as a sublanguage
  - as ML, it uses IF-THEN-ELSE and LET constructs
  - as YaTL, it uses local function definitions
  - as XQL, it uses BEFORE and AFTER operators (global topological order of the XML document)
  - new FILTER operator to do projection while preserving the hierarchy and the order

XQuery

- A query:= a list of local function definitions + the main expression to evaluate
- A XQuery expression:
  - constant (all XML Schema atomic types)
  - variable
  - f(exp1,...exp2)
    - +, -, and, or, union, intersection, etc
  - LET var=expr1 in expr2
  - Xpath expression (for navigation)
  - FLWR expression
  - SORT expr1 by expr2
  - XML node constructors (elements, attributes, etc)

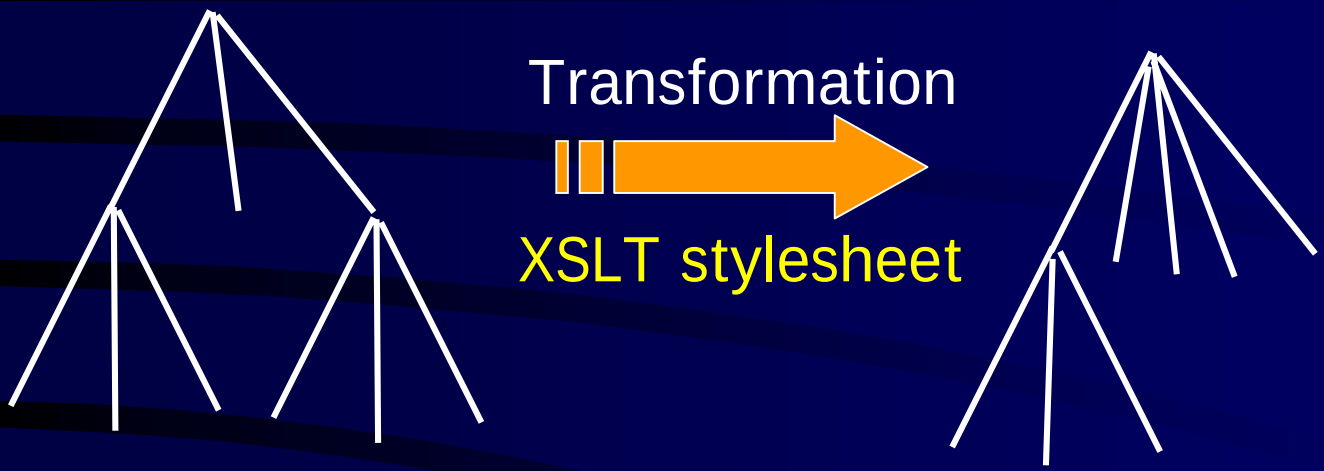
Presenting XML: Extensible Stylesheet Language (XSL)

- Why Stylesheets?
  - **separation of content (XML) from presentation (XSL)**
- Why not just CSS for XML?
  - **XSL is far more powerful:**
    - **selecting elements**
    - **transforming the XML tree**
    - **content based display (result may depend on actual data values)**

XSL(T) Overview

- XSL stylesheets are denoted in XML syntax
- XSL components:
  1. a language for transforming XML documents (XSLT: integral part of the XSL specification)
  2. an XML formatting vocabulary (Formatting Objects: >90% of the formatting properties inherited from CSS)

XSLT Processing Model



XML source tree

XML,HTML,csv, text... result tree

XSLT Elements

- <xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  - root element of an XSLT stylesheet "program"
- <xsl:template match=*pattern* name=*qname* priority=*number* mode=*qname*>
  ...template...
  </xsl:template>
  - declares a rule: (*pattern* => *template*)
- <xsl:apply-templates select = *node-set-expression* mode = *qname*>
  - apply templates to selected children (default=all)
  - optional mode attribute
- <xsl:call-template name=*qname*>

XSLT Processing Model

- XSL stylesheet: collection of template rules
- template rule: (pattern ⇒ template)
- main steps:
  - match pattern against source tree
  - instantiate template (replace current node “.” by the template in the result tree)
  - select further nodes for processing
- control can be a mix of
  - recursive processing ("push": **<**xsl:apply-templates**>** …)
  - program-driven ("pull": **<**xsl:foreach**> ...)**

Template Rule: Example

pattern

template

```
<xsl:template match="product">
  <table>
    <xsl:apply-templates select="sales/domestic"/>
  </table>
  <table>
   <xsl:apply-templates select="sales/foreign"/>
  </table>
</xsl:template>
```

(i) match pattern: process <product> elements
(ii) instantiate template: replace each product element with two HTML tables
(iii) select the <product> grandchildren (“sales/domestic”, “sales/foreign”) for further processing

Match/Select Patterns

- match patterns ⊂ select patterns = defined in **http://w3.org/TR/xpath**
- Examples:
  - **/mybook/chapter[2]/section/***
  - **chapter|appendix**
  - **chapter//para**
  - **div[@class="appendix" and position() mod 2 = 1]//para**
  - **../@lang**

Recursive Descent Processing with XSLT

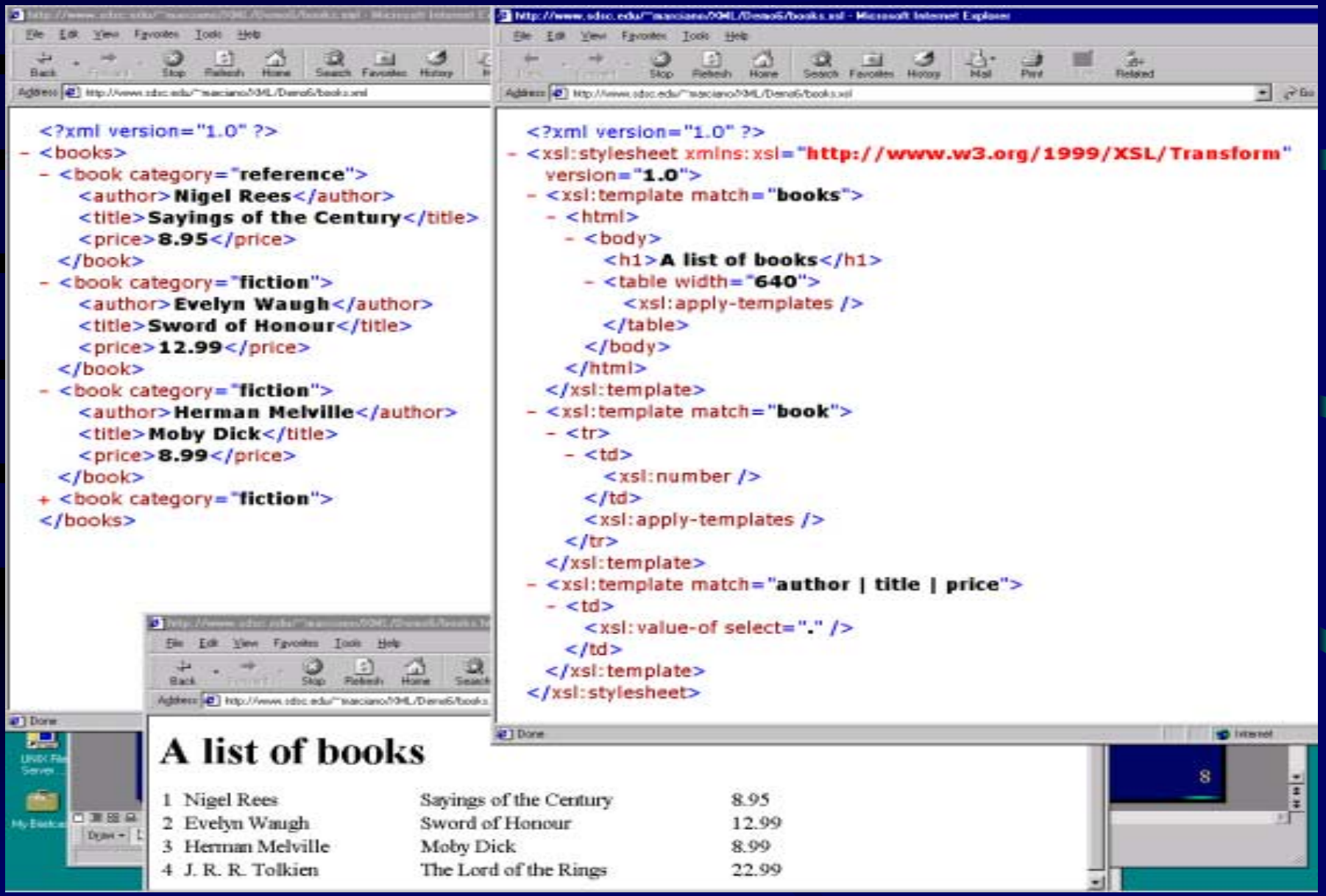
- take some XML file on books: books.xml
- now prepare it with style: books.xsl
- and enjoy the result: books.html
- the recipe for cooking this was:

```
java com.icl.saxon.StyleSheet books.xml books.xsl > books.html
```

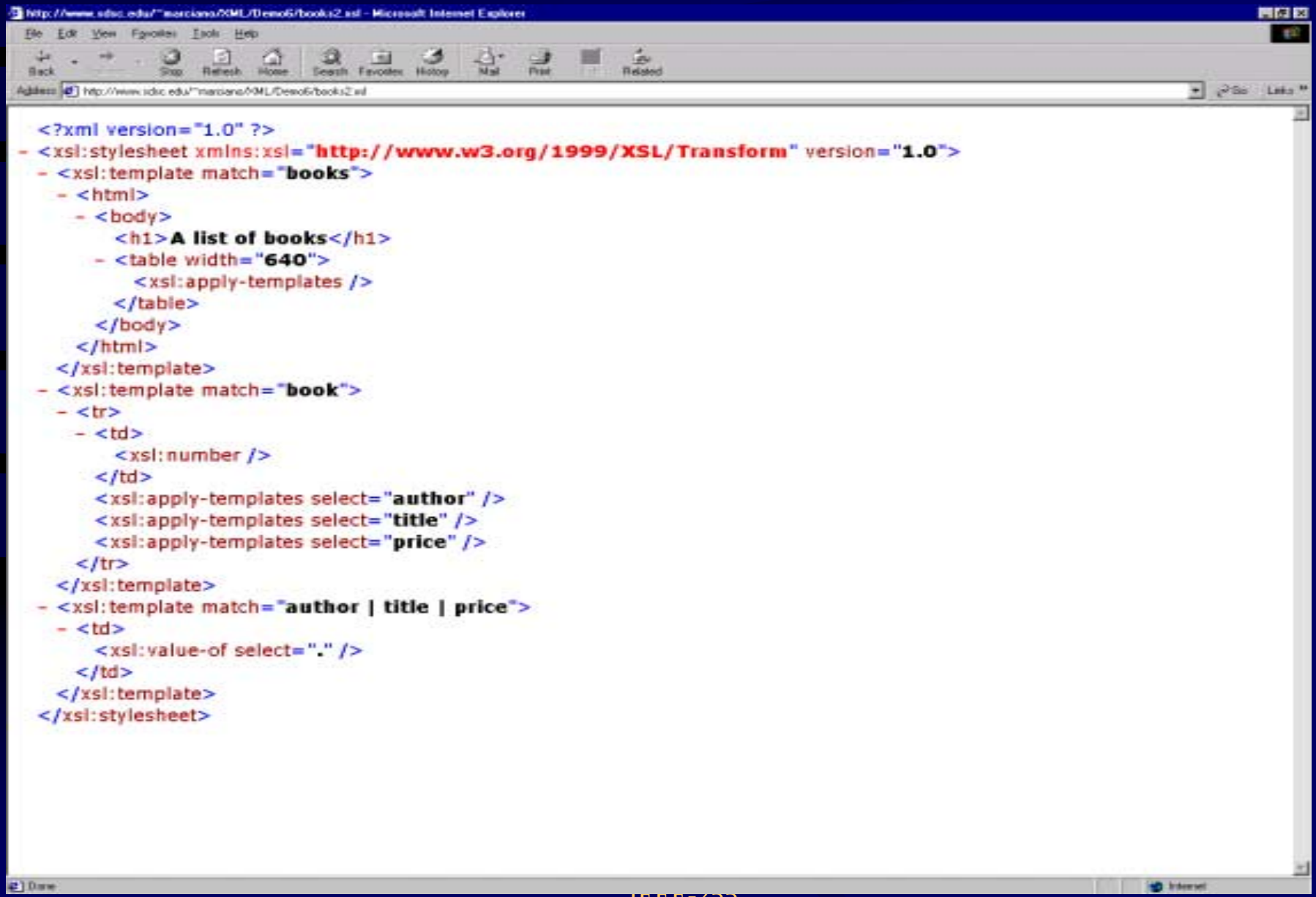
- and now some different flavors: books2.xsl books3.xsl

Source: *XSLT Programmer's Reference,* Michael Kay, WROX

XSLT Example

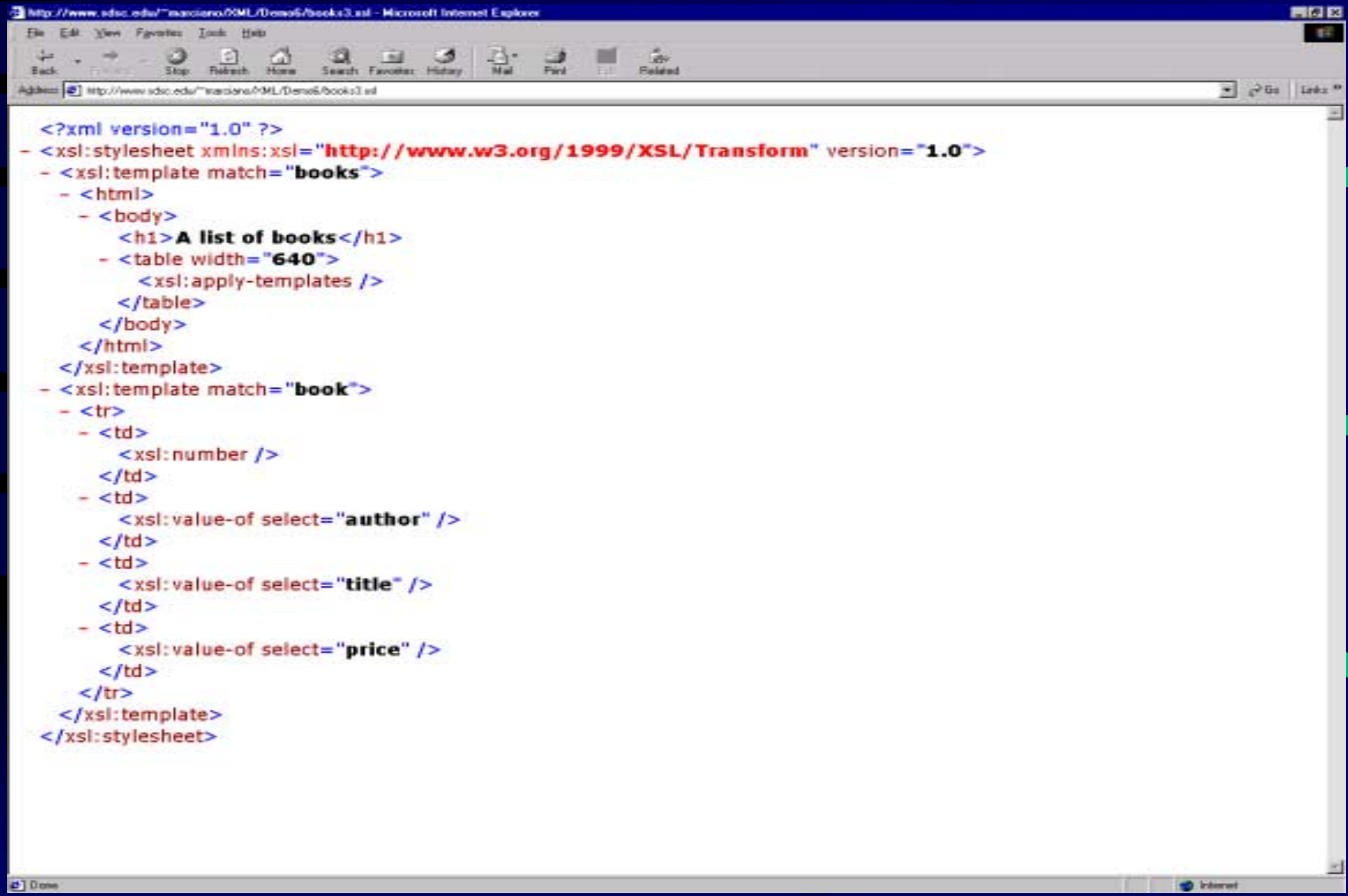


XSLT Example (cont'd)




```
<?xml version="1.0" ?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
  <xsl:template match="books">
    <html>
      <body>
        <h1>A list of books</h1>
        <table width="640">
          <xsl:apply-templates />
        </table>
      </body>
    </html>
  </xsl:template>
  <xsl:template match="book">
    <tr>
      <td>
        <xsl:number />
      </td>
      <xsl:apply-templates select="author" />
      <xsl:apply-templates select="title" />
      <xsl:apply-templates select="price" />
    </tr>
  </xsl:template>
  <xsl:template match="author | title | price">
    <td>
      <xsl:value-of select="." />
    </td>
  </xsl:template>
</xsl:stylesheet>
```

XSLT Example (cont'd)



```
<?xml version="1.0" ?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
  <xsl:template match="books">
    <html>
      <body>
        <h1>A list of books</h1>
        <table width="640">
          <xsl:apply-templates />
        </table>
      </body>
    </html>
  </xsl:template>
  <xsl:template match="book">
    <tr>
      <td>
        <xsl:number />
      </td>
      <td>
        <xsl:value-of select="author" />
      </td>
      <td>
        <xsl:value-of select="title" />
      </td>
      <td>
        <xsl:value-of select="price" />
      </td>
    </tr>
  </xsl:template>
</xsl:stylesheet>
```

Creating the Result Tree...

- Literal result elements: non-XSL elements (e.g., HTML) appear “literally” in the result tree
- Constructing elements:

```
<xsl:element name = "...">
  attribute & children definition
</xsl:element>
```

(similar for **xsl:attribute, xsl:text, xsl:comment**,...)

- Generating text:

```
<xsl:template match="person">
  <p>
   <xsl:value-of select="@first-name"/>
   <xsl:text> </xsl:text>
   <xsl:value-of select="@surname"/>
  </p>
</xsl:template>
```

Creating the Result Tree...

- Further XSL elements for ...
  - Numbering
    - <**xsl:number value**="position()" **format**="1 ">
  - Conditions
    - <**xsl:if test**="position() mod 2 = 0">
  - Repetition...

Creating the Result Tree: Repetition

```
<xsl:template match="/">
  <html>
    <head>
      <title>customers</title>
    </head>
    <body>
      <table>
        <tbody>
          <xsl:for-each select="customers/customer">
            <tr>
              <th>
                <xsl:apply-templates select="name"/>
              </th>
              <xsl:for-each select="order">
                <td>
                  <xsl:apply-templates/>
                </td>
              ...
    </html>
  </xsl:template>
```

Creating the Result Tree: Sorting

```
<xsl:template match="employees">
      <ul>
       <xsl:apply-templates select="employee">
         <xsl:sort select="name/last"/>
         <xsl:sort select="name/first"/>
      </xsl:apply-templates>
      </ul>
</xsl:template>

<xsl:template match="employee">
 <li>
   <xsl:value-of select="name/first"/>
   <xsl:text> </xsl:text>
   <xsl:value-of select="name/last"/>
 </li>
</xsl:template>
```

More on XSLT

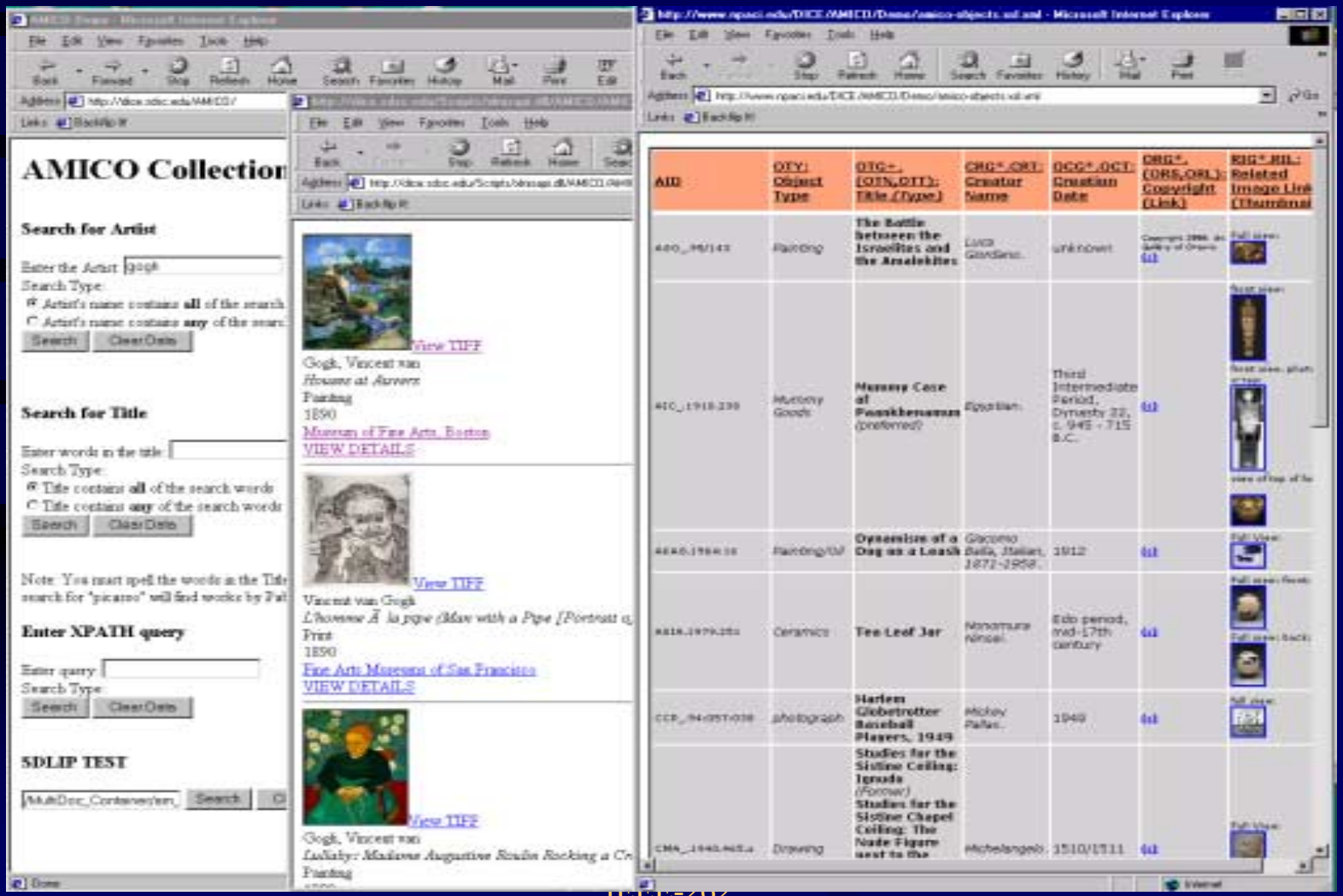
- XSL(T):
  - Conflict resolution for multiple applicable rules
  - Modularization <**xsl:include**> <**xsl:import**>
  - …
- XSL Formatting Objects
  - **a la CSS**
- XPath (navigation syntax + functions)

= XSLT ∩ XPointer

- xslt.com, xml.com

Demonstrations

- XML Queries and Transformations



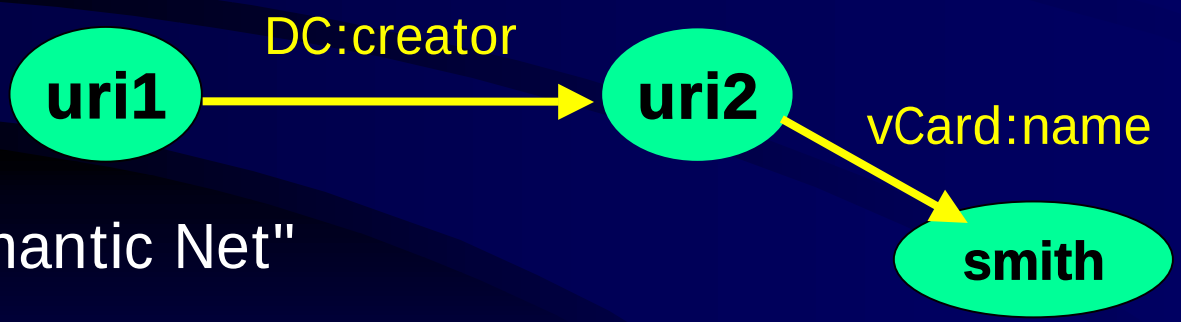
Knowledge Management

Normalized Data/Metadata Representation

- Resource Description Framework (RDF)
  - Metadata model
  - The designer can describe objects, add properties to define and describe them, and also make complicated statements about the objects (statements about relationships between resources).
  - The specification comes in two sections:
    - Model & Syntax (viewed as directed, labeled graphs)
    - RDF Schemas (using an XML vocabulary)

Resource Description Framework (RDF)

- Metadata is useful for information retrieval (esp. if no other schema info or semantics is available)
- Idea: representation independent encoding of metadata as triples (Resource, PropertyType, Value):
  - (uri1, DC:creator, uri2), (uri2, vCard:name, smith), ...



- "Semantic Net"

TOPIC MAPS

ISO/IEC 13250 (Jan. 2000)

Bridging knowledge representation & information management

STANDARD FOR:

- describing knowledge structures
- associating them with information resources
- solution for organizing and navigating large and large information pools

XTM SPECIFICATION

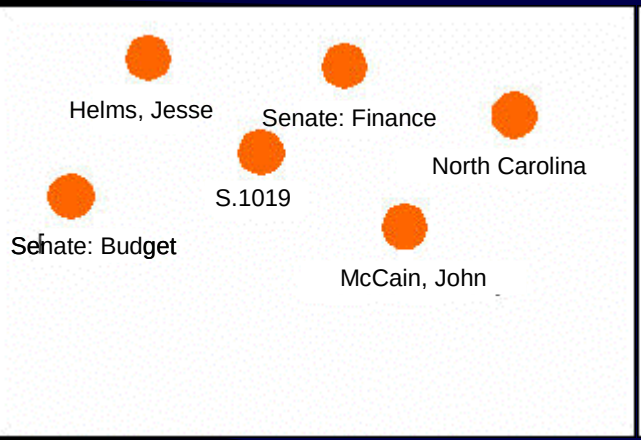
TOPIC MAPS

- New paradigm for K. navigation & synthesis
- Concept of creating style sheets for K.-based information access and navigation
- “GPS for the Web”

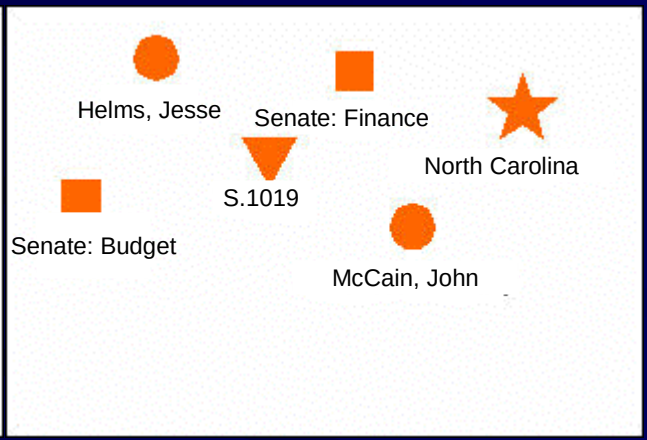
TM’s define semantically customized views

The TAO of Topic Maps

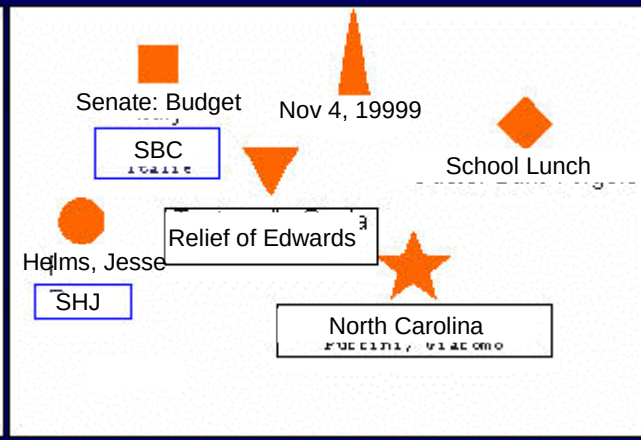
T is for Topic



Topics



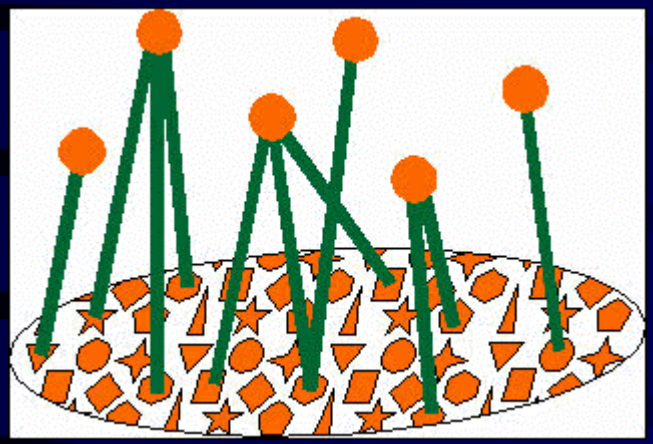
Topic types



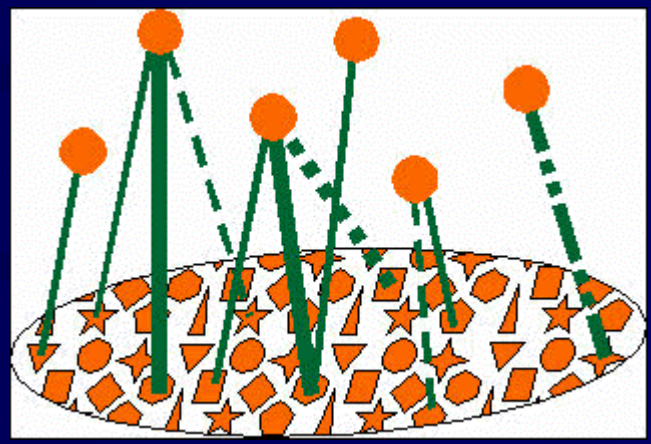
Topic names

The TAO of Topic Maps (cont.)

O is for Occurence



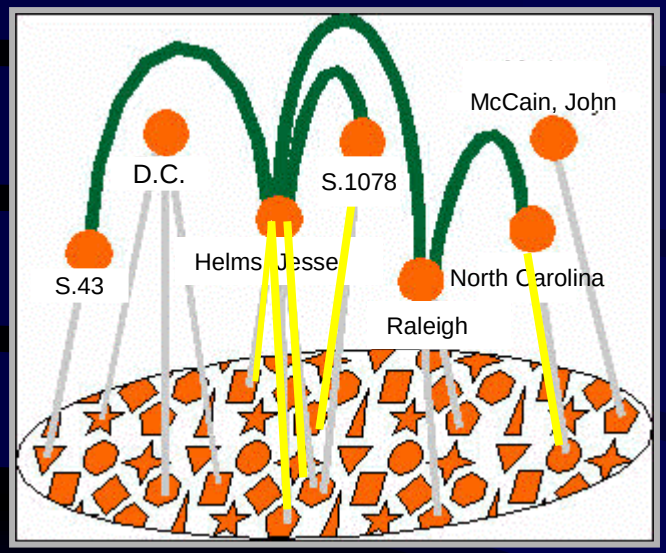
Occurrences



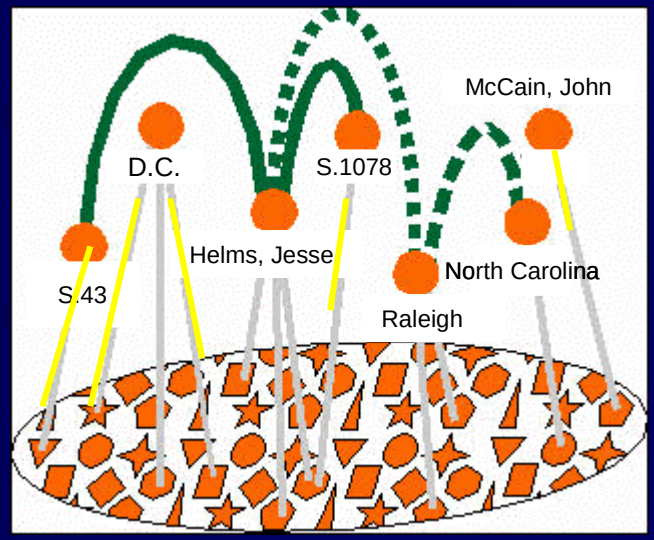
Occurrence Roles

The TAO of Topic Maps (cont.)

A is for Association



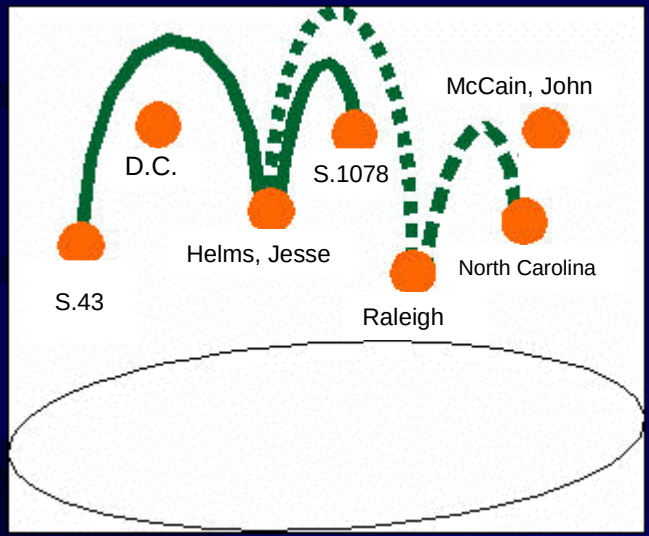
Topic associations



Association types

The TAO of Topic Maps (cont.)

==> *Independence of topic associations & topic occurrences (information resources)*

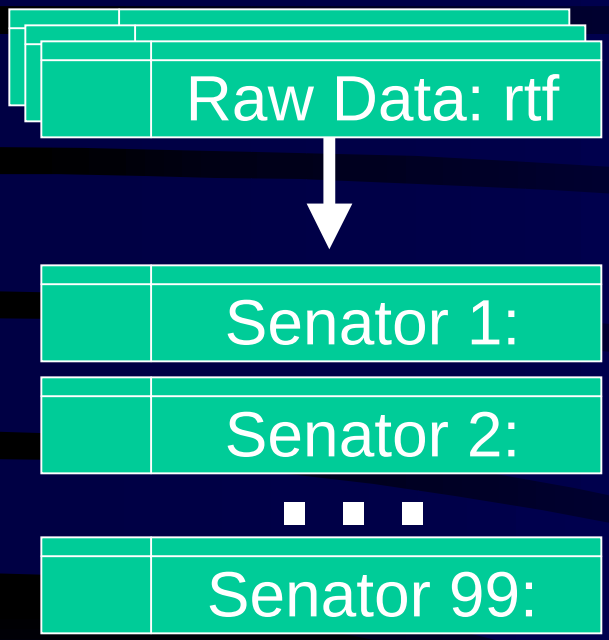


Topic maps as portable semantic networks

References

- XTM DTD -- http://www.topicmaps.org/xtm/index.html

"Senate Legislative Activities" Collection: NARA: 106th Senate



Paul S. Sarbanes of Maryland

(see p. 135, p. 151, etc.)

January 06, 1999 to March 31, 2000

Section I: Sponsored measures
Section II: Cosponsored measures
Section III: Sponsored measures organized by committee referral
- ***Senate: Armed Services***
- ***Senate: Banking***
- ***House: Judiciary***

Section IV: Cosponsored measures organized by committee referral
- ***Senate: Agriculture***
- ***House: Science***

Section V: Sponsored amendments
Section VI: Cosponsored amendments
Section VII: Subject index to measures and amendments

****** S. 151**

Date Introduced: 01/19/1999
Cosponsors: NONE
Official title: A bill to amend the International Maritime Satellite Telecommunications Act…
Latest status: ***Jan 19, 1999*** Read twice and referred to the Committee on Commerce
Abstract: NONE

Subject Index:

Academic Performance:	*S.7, S.514, S.564*
Access to Health Care:	*S.6, S.1678, S.1690*
…	
Zoning and zoning law:	*S.9, S.Con.Res.10, S.Res.41, S.J.Res.39*

TM Example (“XTM-like”) DTD 1/2

```
<!DOCTYPE topicmap [
    <!ELEMENT topicmap (topic | assoc )* >
    <!ELEMENT topic (topname | occurs)* >
    <!ATTLIST topic       id          ID    #REQUIRED
                          types       CDATA #IMPLIED>

    <!ELEMENT topname (basename, dispname, sortname)>
    <!ELEMENT basename (#PCDATA) >
    <!ELEMENT dispname (#PCDATA) >
    <!ELEMENT sortname (#PCDATA) >
```

DTD 2/2

```
<!ELEMENT occurs (locator*) >
<!ELEMENT locator EMPTY >
<!ATTLIST locator    role  CDATA   #REQUIRED
                     href  CDATA   #REQUIRED>
<!ELEMENT assoc (assocrl*) >
<!ATTLIST assoc      types  CDATA  #IMPLIED>
<!ELEMENT assocrl EMPTY >
<!ATTLIST assocrl    role   CDATA   #REQUIRED
                     href   CDATA   #REQUIRED>
]>
```

TM Example – The XML doc itself (1/4)

```
<topicmap>
  <topic id="t1" types="SubjectEntry">
    <topname>
      <basename>Apartment houses</basename>
      <dispname>Apt. Houses</dispname>
      <sortname>APARTMENTHOUSES</sortname>
    </topname>
    <occurs>
      <locator role="DiscussedIn" href="#S.463" />
    </occurs>
  </topic>
```

TM XML Document (2/4)

```
<topic id="t2" types="SubjectEntry">
    <topname>
      <basename>Children</basename>
      <dispname>Child.</dispname>
      <sortname>CHILDREN</sortname>
    </topname>
    <occurs>
      <locator role="DiscussedIn" href="#S.300" />
      <locator role="DiscussedIn" href="#S.463" />
      <locator role="DiscussedIn" href="#S.1638" />
      <locator role="DiscussedIn" href="#S.1673" />
      <locator role="DiscussedIn" href="#S.1709" />
      <locator role="DiscussedIn" href="#S.Res.125" />
      <locator role="DiscussedIn" href="#S.Res.258" />
    </occurs>
</topic>
```

TM XML Document (3/4)

```
<topic id="t3" types="SubjectEntry">
    <topname>
       <basename>Welfare</basename>
       <dispname>Welf.</dispname>
       <sortname>WELFARE</sortname>
    </topname>
    <occurs>
       <locator role="DiscussedIn" href="#S.463" />
       <locator role="DiscussedIn" href="#S.1277" />
       <locator role="DiscussedIn" href="#S.1709" />
       <locator role="DiscussedIn" href="#S.Con.Res.28" />
       <locator role="DiscussedIn" href="#S.Res.125" />
       <locator role="DiscussedIn" href="#S.Res.260" />
    </occurs>
 </topic>

 <topic id="t4" types="SubjectEntry">
    <topname>
       <basename>Youth employment</basename>
       <dispname>Youth empl.</dispname>
       <sortname>YOUTEMPLOYMENT</sortname>
    </topname>
    <occurs>
       <locator role="DiscussedIn" href="#S.463" />
    </occurs>
 </topic>
```

TM XML Document (4/4)

```
  <assoc types="CoDiscussedInExactlyOneBill">
      <assocrl role="DiscussedInSameBill" href="t1" />
      <assocrl role="DiscussedInSameBill" href="t2" />
      <assocrl role="DiscussedInSameBill" href="t3" />
      <assocrl role="DiscussedInSameBill" href="t4" />
   </assoc>

   <assoc types="CoDiscussedInTwoOrMoreBills">
      <assocrl role="DiscussedInSameBill" href="t2" />
      <assocrl role="DiscussedInSameBill" href="t3" />
   </assoc>

</topicmap>
```

Topic Maps Self-Control

Extreme ML 2000, Montreal – Hans Holger Rath

- Topic Map templates
  - Logical container for the “schema” part of the map:
    - Type/theme declarations
    - Constraints
    - Inference rules
- Association properties
  - Transitivity
  - Support inferencing capabilities
- Type hierarchies: commercial site (www.ontopia.net)
  - Super-subclassing
  - Inferencing
- Consistency checking with constraints
  - Rule-based constraints control validation process
  - Constraint patterns

Topic Maps Self-Control (... continued)

- Inference rules
  - Deduce additional knowledge
  - Inference patterns
  - Examples:
    - If $topic1 is a sibling of $topic2 and $topic1 is a male then $topic1 is a brother
    - <assoc id="ir-male"
      type="class-instance"
      scope="ir-schema">
      <assocrl type="instance"> ir-topic-A-PERSON</assocrl>
      <assocrl type="class"> male </assocrl>
      </assoc>
- → THE TM control their own structure and content!

Model-Based Mediation

Integrated-**DTD** :=

XML-QL(Src1-**DTD**,...)

Domain Map

Integrated-**CM** :=

CM-QL(Src1-**CM**,...)

IF φ THEN ψ

Logical Domain Constraints

XML DTDs

A = (B*|C),D
B = ...

C1
C2
R
C3

Classes, Relations, *is-a*, *has-a*, ...

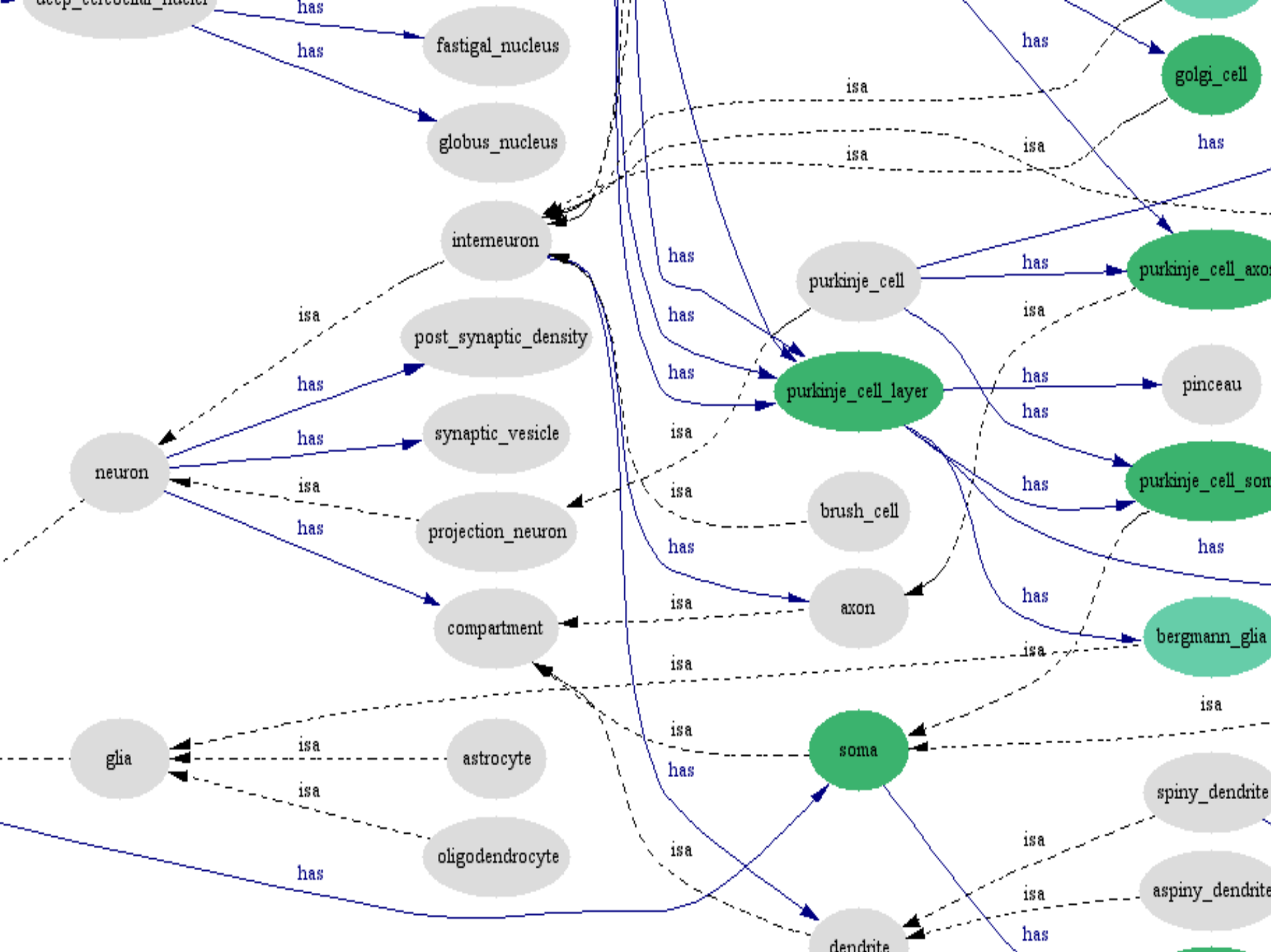
XML Elements

(XML) Objects

Raw Data

***XML** Models*

Conceptual Models


has
fastigal_nucleus
has
globus_nucleus
isa
golgi_cell
has
isa
isa
interneuron
has
purkinje_cell
has
purkinje_cell_axon
isa
post_synaptic_density
has
has
purkinje_cell_layer
has
pinceau
has
has
synaptic_vesicle
isa
neuron
has
purkinje_cell_som
isa
isa
brush_cell
has
has
projection_neuron
has
axon
has
compartment
isa
bergmann_glia
isa
isa
isa
soma
isa
glia
isa
astrocyte
has
spiny_dendrite
isa
isa
oligodendrocyte
isa
has
aspiny_dendrite
isa
has
dendrite
has

Storage Transparencies

- Location transparency
  - Distribution of data collection across multiple physical resources
- Name transparency
  - Attributed based access to data
- Protocol transparency
  - Common API for access to remote data resources
- Time transparency
  - Minimization of data access latency

SDSC Storage Resource Broker & Meta-data Catalog

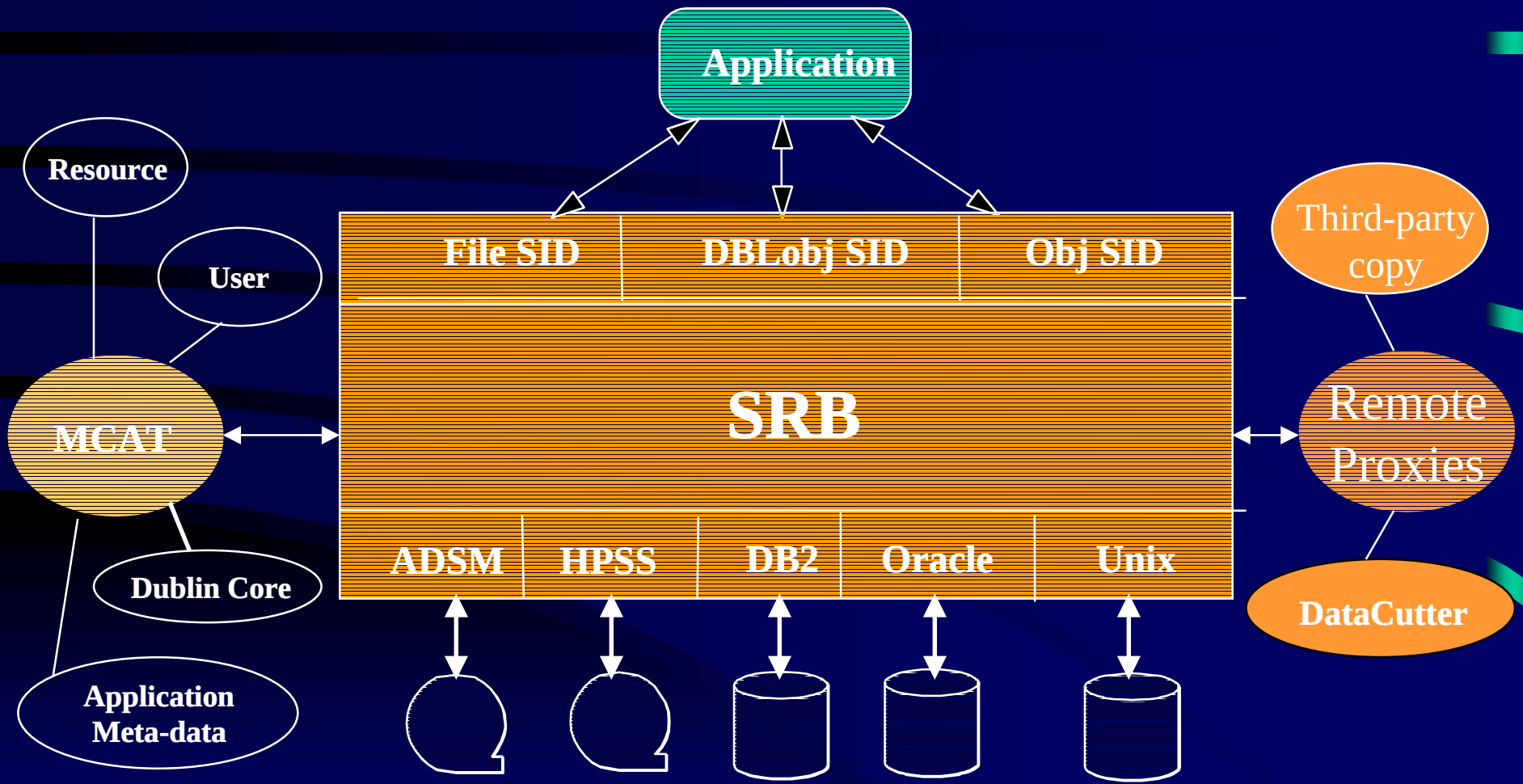


Table Access Interface

- Facility to access tabular data using SRB API
- View SQL queries as Locators (Path Names or URI)
- Apply open, close, read, write operations
- Provide for very general queries to specific queries
  - any query on a database to soft queries to hard-coded queries
- Access Result Table as a Stream
- Provide Server-side operations to present results
  - Forms, HTML, XML, ...
  - Data Wetting, Charting, Visualization
- Multi-modal Ingestion
  - SQL ingestion
  - Packed Ingestion - useful in data movement and replication
  - Directly ingest data marked by HTML, XML, ...

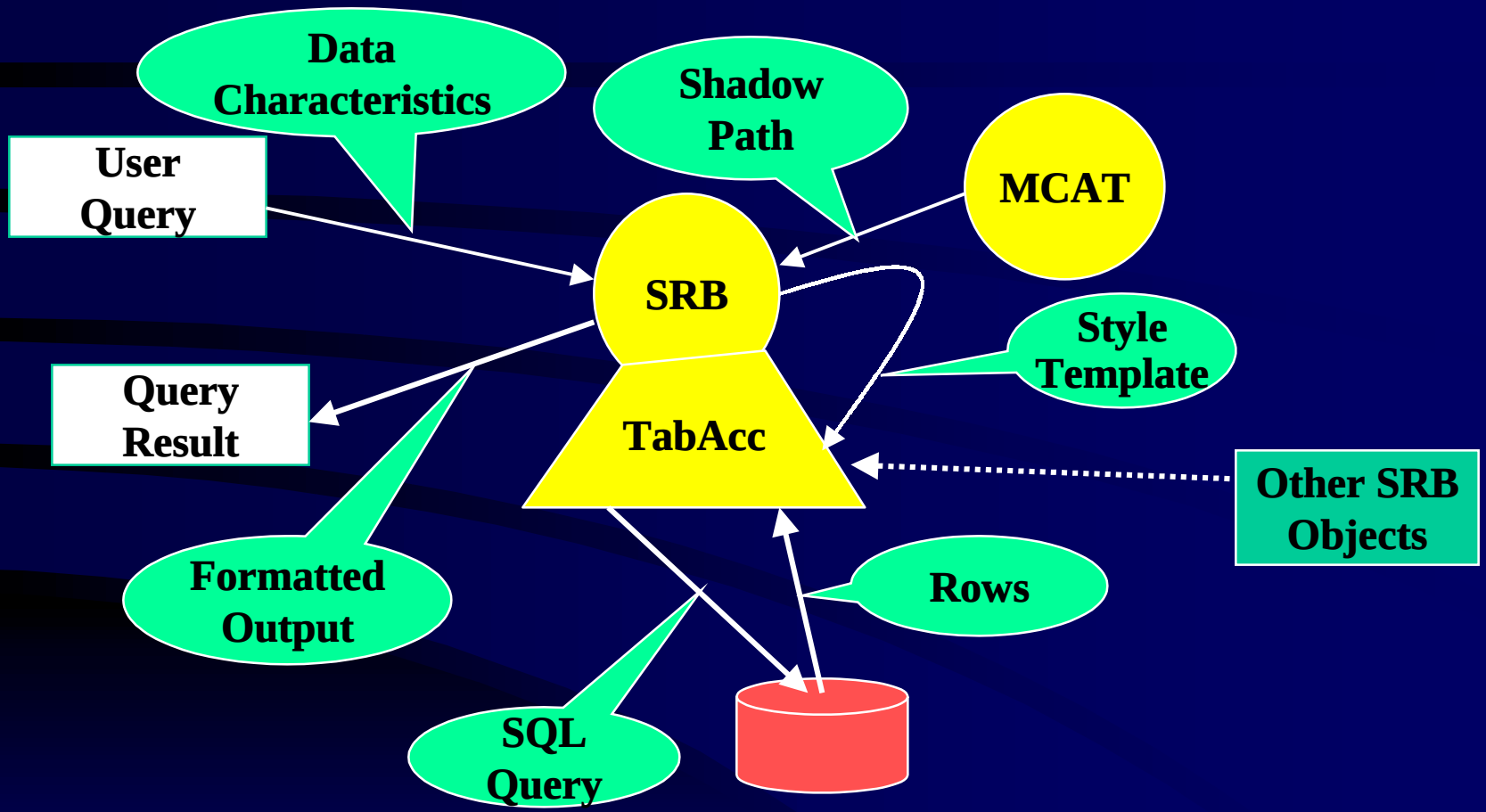
Server-side Presentation

- Mark up data before sending to client
- Generic mark ups - HTML, XML
- Specific mark ups - Template
- Template Language
  - Allows data element variables
  - Control structure - if-then-else, for , nested
  - Object-in-object
- User specifies mark up at query time
- Can be used for other data streams also!

Shadow Objects

- A feature for registering partial physical locations
  - Partial path in a file system allows one to access files under a directory
  - Partial SQL query allows for modification at access time.
- Registering a null query allows for any query to be allowed in a database

Table Access



T-Language

- Mix of Interpretable Code & Viewable data
- Interpretable Code
  - Control Flow: if-then-else, for-loops
  - System Variables: database, table, query information
  - User-definable Variables
  - Evaluable Expressions:
    - arithmetic, logical, string & regular expressions
  - Embedded SRB Objects
  - Built-in Functions

Sample T-code

```
  :::
<TFORMIF> ('$$$2:' ? 'file system') == 1
   <TFORMTHEN>
      <TFORMIF> ('$$$2:' ? 'hpss.*system') == 1
         <TFORMTHEN> <TR BGCOLOR="#AAFFFF">
         <TFORMELSE><TR BGCOLOR="#FFAAFF">
      <TFORMENDIF>
   <TFORMELSE>
      <TR BGCOLOR="#FFFFAA">
 <TFORMENDIF>
  :::
```

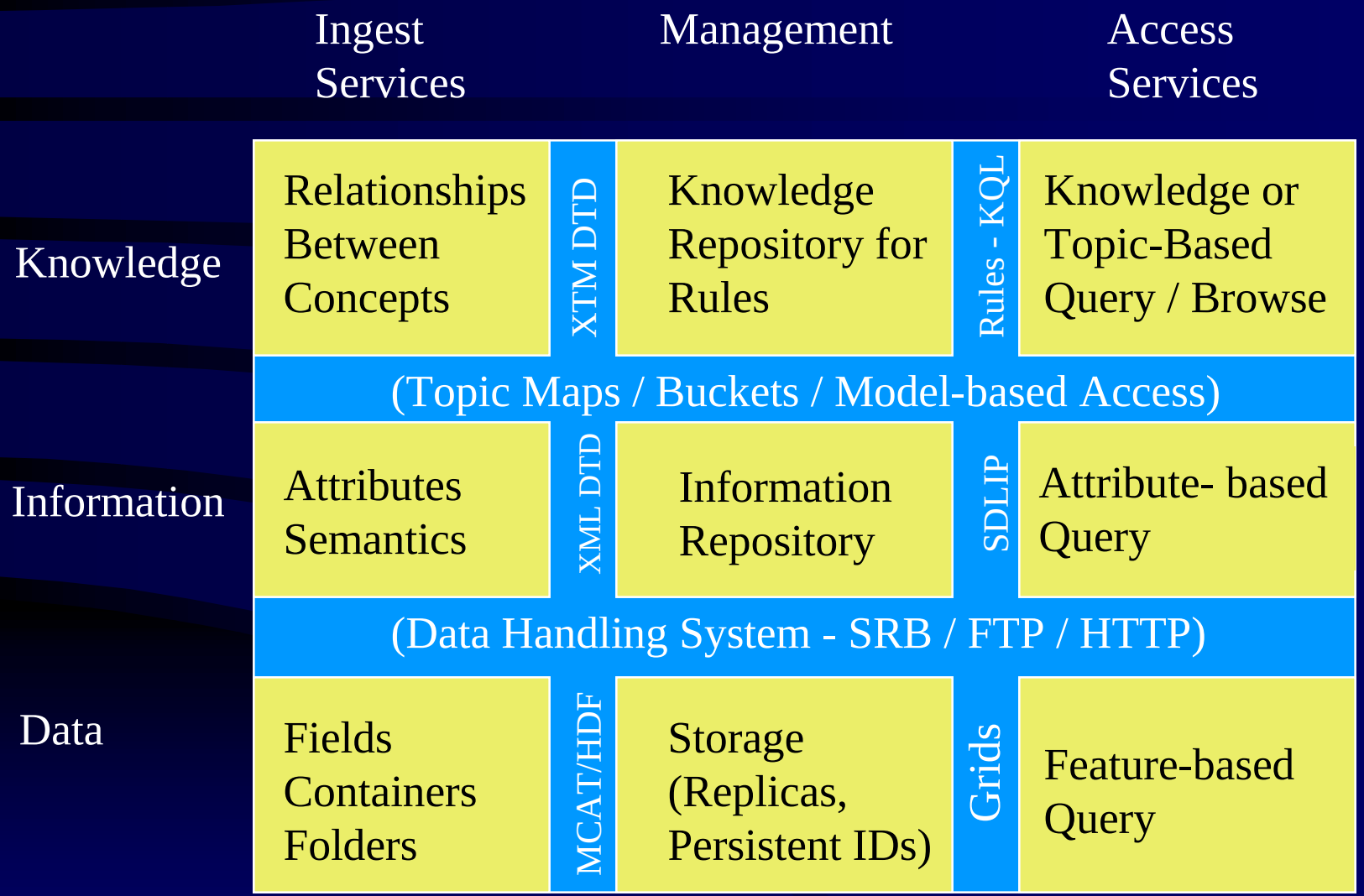
Simplest Definitions

- Data
  - Digital object
  - Objects are streams of bits
- Information
  - Any tagged data, which is treated as an attribute.
  - Attributes may be tagged data within the digital object, or tagged data that is associated with the digital object
- Knowledge
  - Relationships between attributes
  - Relationships can be procedural/temporal, structural/spatial, logical/semantic, functional

Types of Knowledge Relationships

- Logical / semantic
  - Digital Library cross-walks
- Temporal / procedural
  - Workflow systems
- Spatial / structural
  - GIS systems
- Functional / algorithmic
  - Scientific feature analysis

Knowledge Based Persistent Archive



Further Information

http://www.npaci.edu/DICE