XML

Vũ Tuyết Trinh
trinhvt@soict.hust.edu.vn

Department of Systems Information
SoICT - HUST

Outline

- □ Our context: data sharing
- □ XML as a data model
  - DTD
  - XSchema
- □ XML "database"
  - Storing
  - Querying
- □ XML for data exchange
  - Mapping schemas ...

Data sharing – requirements

- A format for exchanging information with
  - a uniform encoding for data
  - self-describing
  - posibility of describing the (irregular) structure in an “understandable” way
  - extensible
- Standard languages, interfaces, and tools

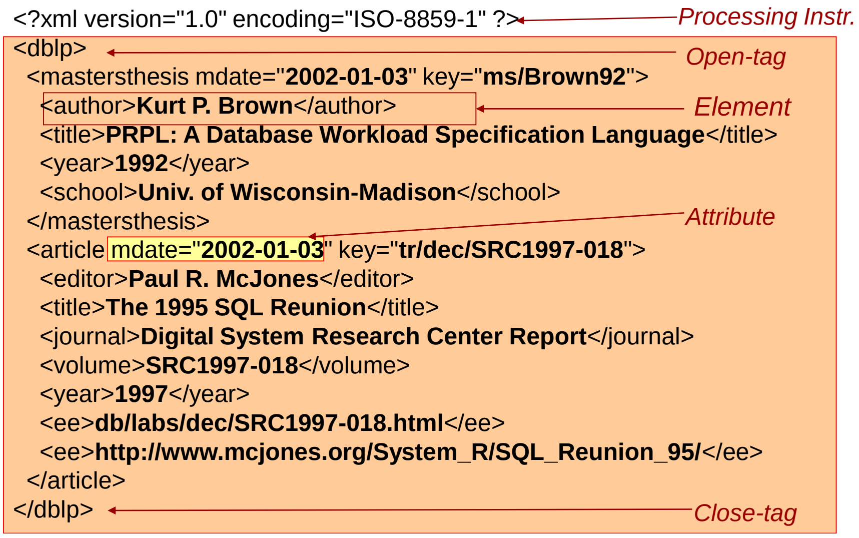
Data representation

- Aims
  - Enabling the representation (or translation) of data from multi sources
  - Enabling the formulation of *user* queries
- Problems
  - Heterogeneity of underlying data models
  - Lacks of structure (or irregular structure) of underlying data sources
- A solution: XML

XML - eXtensible Markup Language

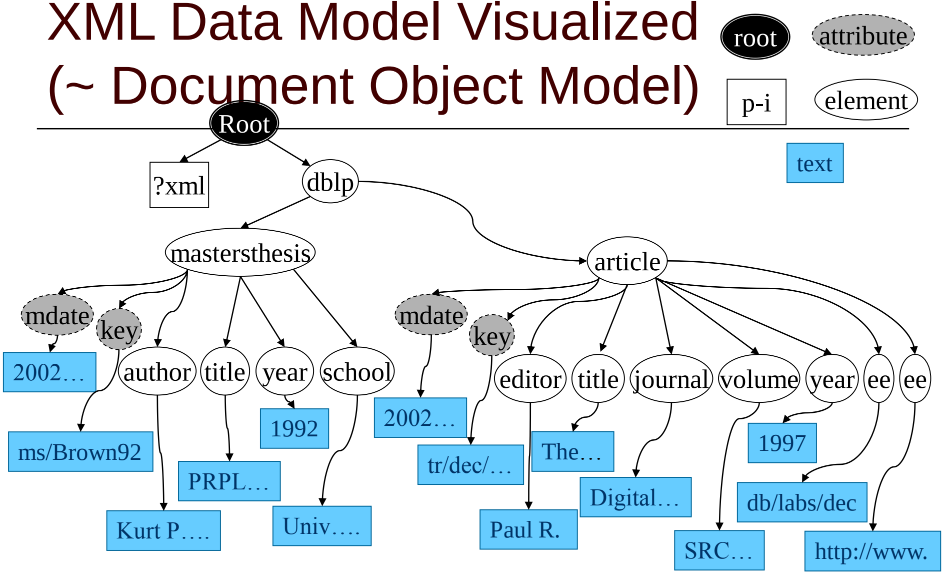
- Origin: defining a format for exchanging documents on Web
- History
  - a W3C standard [W3C]
  - started at 6/1996;
  - 02/1998 : first recommendation ***XML 1.0***
- "Extensions"
  - Structures of documents : ***DTD, XML Schema, XPath, …***
  - Pointors et links : ***XLink, XPointer,*** …
  - Transformation : ***XSLT, XSL-FO, …***
  - Normalized applications : ***RDF, SMIL, XMI, …***

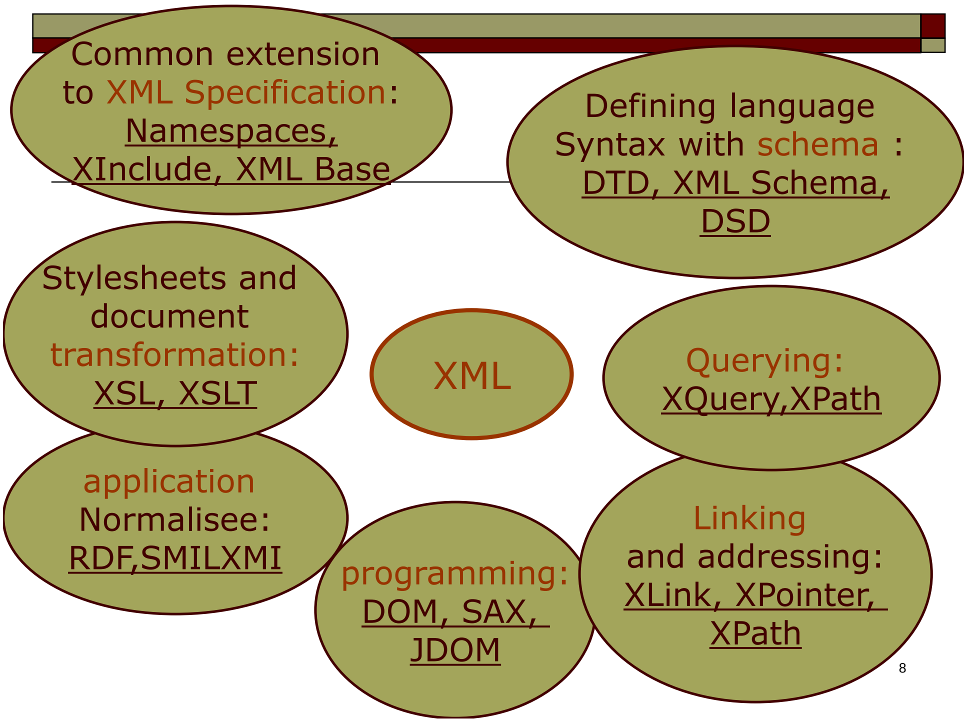
Example XML Document



```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<dblp>
  <mastersthesis mdate="2002-01-03" key="ms/Brown92">
    <author>Kurt P. Brown</author>
    <title>PRPL: A Database Workload Specification Language</title>
    <year>1992</year>
    <school>Univ. of Wisconsin-Madison</school>
  </mastersthesis>
  <article mdate="2002-01-03" key="tr/dec/SRC1997-018">
    <editor>Paul R. McJones</editor>
    <title>The 1995 SQL Reunion</title>
    <journal>Digital System Research Center Report</journal>
    <volume>SRC1997-018</volume>
    <year>1997</year>
    <ee>db/labs/dec/SRC1997-018.html</ee>
    <ee>http://www.mcjones.org/System_R/SQL_Reunion_95/</ee>
  </article>
</dblp>
```

XML Data Model Visualized (~ Document Object Model)





Common extension to XML Specification: Namespaces, XInclude, XML Base

Defining language Syntax with schema : DTD, XML Schema, DSD

Stylesheets and document transformation: XSL, XSLT

XML

Querying: XQuery,XPath

application Normalisee: RDF,SMILXMI

programming: DOM, SAX, JDOM

Linking and addressing: XLink, XPointer, XPath

XML in 10 points

[http://www.w3.org/XML/1999/XML-in-10-points]

1. XML is for structuring data
2. XML looks a bit like HTML
3. XML is text, but isn't meant to be read
4. XML is verbose by design
5. XML is a family of technologies
6. XML is new, but not that new
7. XML leads HTML to XHTML
8. XML is modular
9. XML is the basis for RDF and the Semantic Web
10. XML is license-free, platform-independent and well-supported

Tools

- Editors
  - any editors, save as .xml
  - http://www.xmlsoftware.com (many free editors)
- Parsers
  - Xerces at http://xml.apache.org

Outline

- ✓ Our context: data sharing
- □ XML as a data model
  - DTD
  - XML Schema
- □ XML “database”
  - Storing
  - Indexing
  - Querying
- □ XML for data exchange
  - Mapping schemas

XML as a data model

- □ Reminder of data model [Codd, 1980]
  - a collection of data structures
  - a collection of operations for data manipulation
  - a collection of data integrities
- □ 7 types of nodes in XML
  - Document (root)
  - Element
  - Attribute
  - Processing instruction
  - Text (content)
  - Namespace
  - Comment

Document Type Definitions (DTDs)

- A grammar defining XML structure
  - XML document can have an associated DTD (but not always) & a root element
  - DTD specifies children of the root (and so on)
- XML’s components in DTD
  - Elements: main building block
  - Tags: markup elements.
  - Attributes: extra information about elements
  - Entities: variables used to define common textv (&lt, &gt, &amp, &quot, &apos, ...)
  - PCDATA: parsed character data (between a open and a close tags of an element)
  - CDATA: character data (not be parsed by a parser)
- Special attributes
  - IDs – special attributes as keys for elements
  - IDREFs – references to IDs
  - IDREFS – space-delimited list of IDREFs

Example

- Example DTD:

```
<!ELEMENT dblp((mastersthesis | article)*)>
<!ELEMENT mastersthesis
   (author,title,year,school,committeemember*)>
<!ATTLIST    mastersthesis(
     mdate          CDATA          #REQUIRED
     key            ID             #REQUIRED
     advisor        CDATA          #IMPLIED>
<!ELEMENT author(#PCDATA)>
…
```

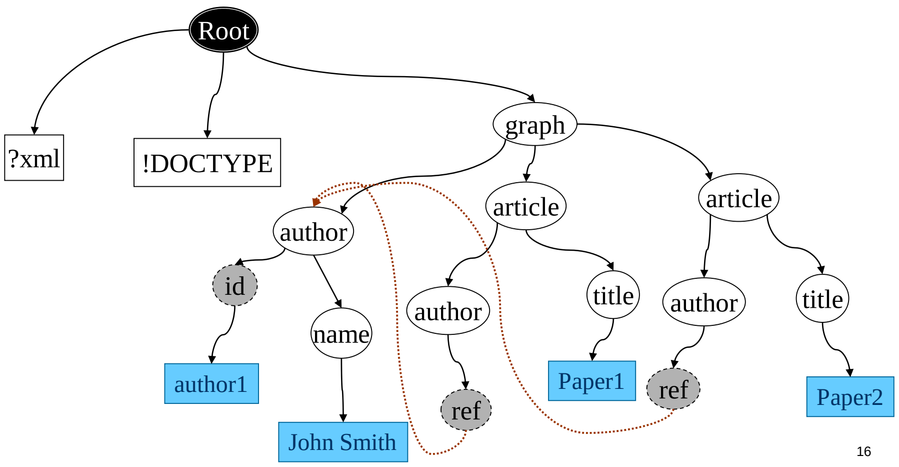
- Example use of DTD in XML file:

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE dblp SYSTEM “my.dtd">
<dblp>…
```

Representing Graphs in XML

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE graph SYSTEM “special.dtd">
<graph>
   <author id=“author1”>
        <name>John Smith</name>
   </author>
   <article>
        <author ref=“author1” />  <title>Paper1</title>
   </article>
   <article>
        <author ref=“author1” />  <title>Paper2</title>
   </article>
…
```

Graph Data Model



DTDs - remarks

- Advantages
  - Simple
  - Enabling structure representation
- Inconvenients
  - Not themselves in XML → need to build tools for them
  - Don’t capture types of scalars
  - Global ID/reference space is inconvenient
  - No way of defining OO-like inheritance
  - XML Schema:
- Standard DTDs at http://www.schema.net

XML Schema

- An official W3C recommendation for defining XML document’s structure (since May 2001).
- Defines:
  - elements that can appear in a document
  - attributes that can appear in a document
  - relationship aamong elemnts (child-parent)
  - the order of child elements
  - the number of child elements
  - if an element is empty or can include text
  - data types for elements and attributes
  - default and fixed values for elements and attributes

XML Schema - Successor of DTD

- Written in XML
- Supporting data types
- Extensible to future additions
- Richer and more useful than DTDs
- Supporting namespaces
  - XML Namespaces provide a method to avoid element name conflicts

Element name conflict - example

```
<table>
    <tr>
        <td>Apples</td>
        <td>Bananas</td>
    </tr>
</table>
```

```
<table>
    <name>African Coffee Table</name>
    <width>80</width>
    <length>120</length>
</table>
```

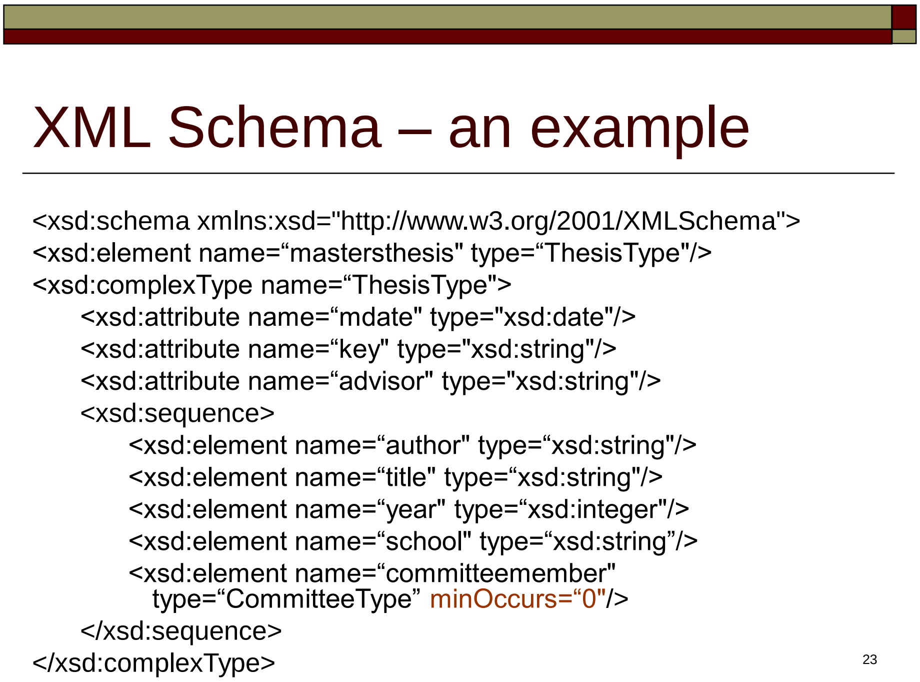
Solving Name Conflicts - prefix

```
<h:table>
    <h:tr>
        <h:td>Apples</h:td>
        <h:td>Bananas</h:td>
    </h:tr>
</h:table>

<f:table>
    <f:name>African Coffee Table</f:name>
    <f:width>80</f:width>
    <f:length>120</f:length>
</f:table>
```

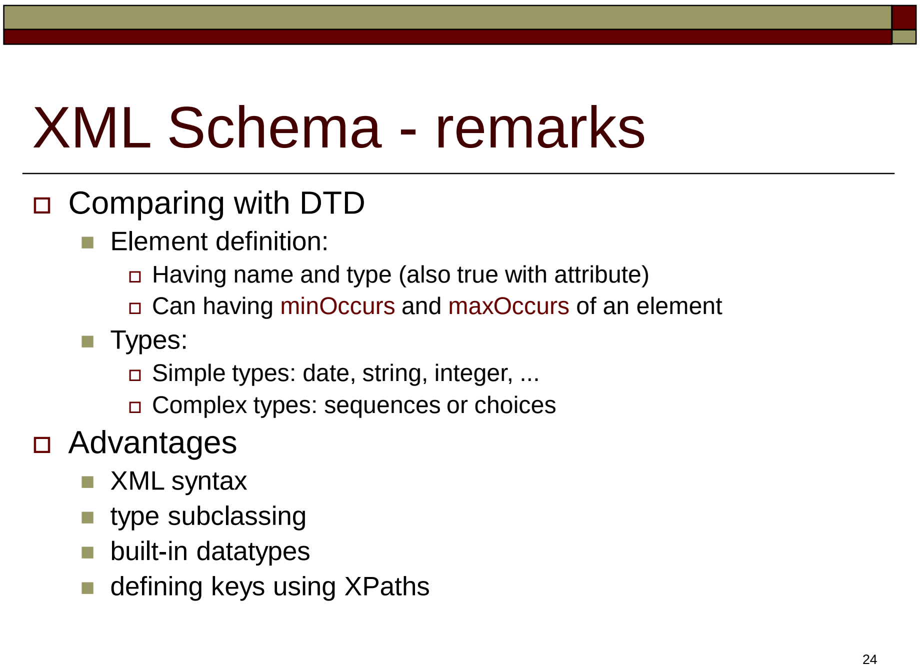
Namespace

```
<h:table xmlns:h="http://www.w3.org/TR/html4/">
    <h:tr>
            <h:td>Apples</h:td>
            <h:td>Bananas</h:td>
    </h:tr>
</h:table>
<f:table xmlns:f="http://www.w3schools.com/furniture">
    <f:name>African Coffee Table</f:name>
    <f:width>80</f:width>
    <f:length>120</f:length>
</f:table>
```



XML Schema – an example

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<xsd:element name="mastersthesis" type="ThesisType"/>
<xsd:complexType name="ThesisType">
    <xsd:attribute name="mdate" type="xsd:date"/>
    <xsd:attribute name="key" type="xsd:string"/>
    <xsd:attribute name="advisor" type="xsd:string"/>
    <xsd:sequence>
        <xsd:element name="author" type="xsd:string"/>
        <xsd:element name="title" type="xsd:string"/>
        <xsd:element name="year" type="xsd:integer"/>
        <xsd:element name="school" type="xsd:string"/>
        <xsd:element name="committeemember"
          type="CommitteeType" minOccurs="0"/>
    </xsd:sequence>
</xsd:complexType>
```



XML Schema - remarks

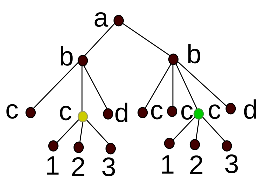
- Comparing with DTD
  - Element definition:
    - Having name and type (also true with attribute)
    - Can having minOccurs and maxOccurs of an element
  - Types:
    - Simple types: date, string, integer, ...
    - Complex types: sequences or choices
- Advantages
  - XML syntax
  - type subclassing
  - built-in datatypes
  - defining keys using XPaths

XML “database”

- Problematics of DB approach
  - Great data amount
  - Persistent storage, résistant to failures, integrity control, data coherence
  - Access ~ reading & writing data
  - Data access ~ query
    - Queries in declaratif language
    - Compiling queries into execution plans (programs) ~ optimization
- Main problems
  - Storage
  - Querying

Impacts of XML properties on its storage

- Heterogeneity
  - XML autorises: optional elements (?), repetitions (+, *), choice (|)
- Identity
  - Two elements having the same content are different



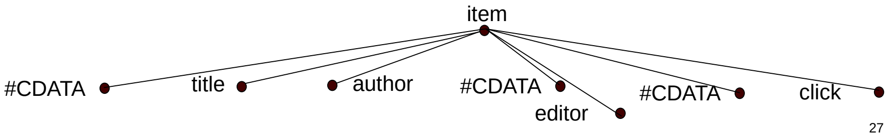
a/b/c[1] <> a/b/c[2]

a/b[count(c)>2]//c

Impacts of XML properties ... (2)

- Presence of "mixed content"
  - Data inside elements
  - Data outside elements

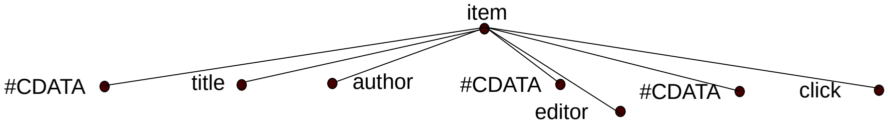
```
<item>Une introduction aux
          <title>Légendes du Graal</title> par
          <author>P.Boulenger</author> aux éditions
          <editor>Dunod</editor> pour juste 5.55 Euros !
          <click>…</click>
</item>
```



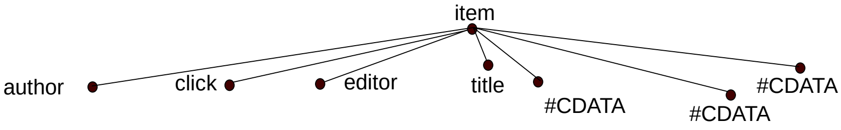
Impacts of XML properties ... (3)

- Elements order

```
<item>Une introduction aux <title>Légendes du Graal</title> par
<author>P.Boulenger</author> aux éditions <editor>Dunod</editor> pour juste 5.55
Euros ! <click>…</click> </item>
```

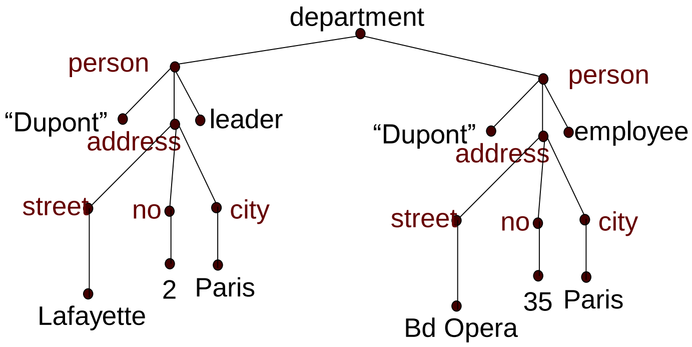


```
<item><author>P.Boulenger</author><click>…</click> <editor>Dunod</editor>
<title>Légendes du Graal</title>par pour juste 5.55 Euros ! Une introduction aux</item>
```



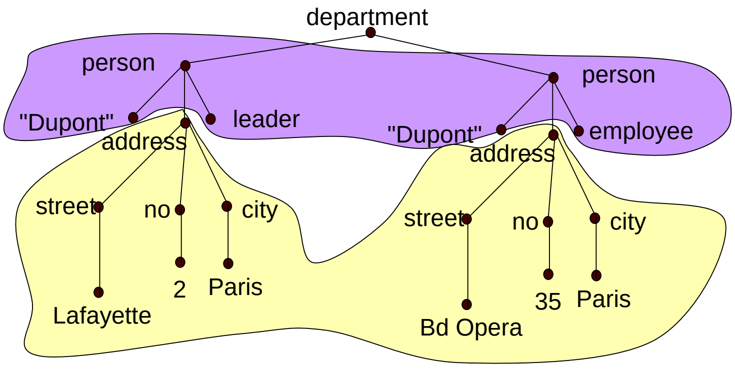
Impacts of XML properties ... (4)

- Structure
  - Implicit in XML document
  - Navigationable structructure



Impacts of XML properties ... (4)

- Existance of a schema



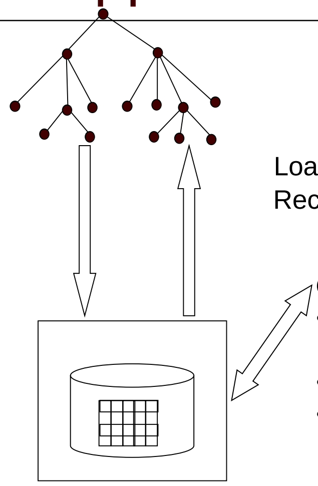
XML Storage - options

- Files
  - Textuel files
  - Characteristics

    + Easy, no special supports
    + Querying: grep, textuel index
    - Storage space
    - Cant querying on data structure
- Extended relational(-object) databases (non-native)
  - Relational schema for XML storage
  - XML elements/attributes/...
  - Querying with SQL
- Native XML database
  - Based on a model for storing and querying graph data
  - Order
  - XML documents as principle objects

Non-native approach

Document XML



Loading RDBMS
Recomposing XML document

Query processing
- translation from XQuery into SQL
- execution of SQL queries
- Building XML results

SGBDR (ou R-O)
- Persistence
- Indexation
- reliability
- Transactions

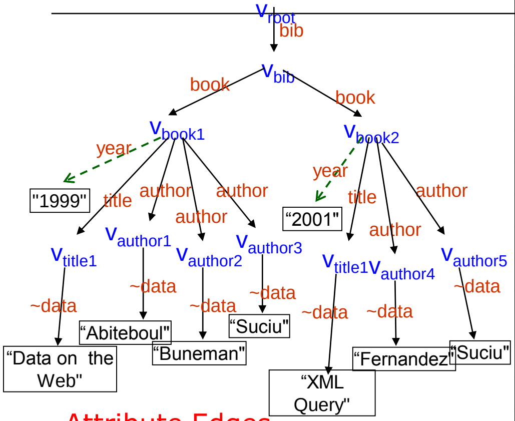
XML in Database Tables

Student-grades

stud-id	course	grade
1	455-S04	A
23	380-F03	B

```
<student-grades>
    <tuple>
        <stud-id>1</stud-id>
        <course>455-S04</course>
        <grade>A</grade>
    </tuple>
    <tuple>
        <stud-id>23</stud-id>
        <course>380-F03</course>
        <grade>B</grade>
    </tuple>
</student-grades>
```

Element/Attribute modeling



Element Containement Edges

Edge	Name	Parent	Child
e1	"bib"	V_{root}	V_{bib}
e2	"book"	V_{bib}	V_{book1}
e3	"book"	V_{bib}	V_{book2}
e5	"title"	V_{book1}	V_{title1}
e6	"Author"	V_{book1}	$V_{author1}$
e7	"Author"	V_{book1}	$V_{author2}$
e8	"Author"	V_{book1}	$V_{author3}$
e10	"title"	V_{book2}	V_{title2}
e11	"Author"	V_{book2}	$V_{author4}$
e12	"Author"	V_{book2}	$V_{author5}$
e13	~data	V_{title1}	"Data on the web"
e14	~data	$V_{author1}$	"Abiteboul"
e15	~data	$V_{author2}$	"Buneman"
e16	~data	$V_{author3}$	"Suciu"
e17	~data	V_{title2}	"XML Query"
e18	~data	$V_{author4}$	"Fernandez"
e19	~data	$V_{author5}$	"Suciu "

Attribute Edges

Edge	Name	Parent	Child
e4	"year"	V_{book1}	("1999" ,String)
e9	"year"	V_{book2}	("2000" ,String)

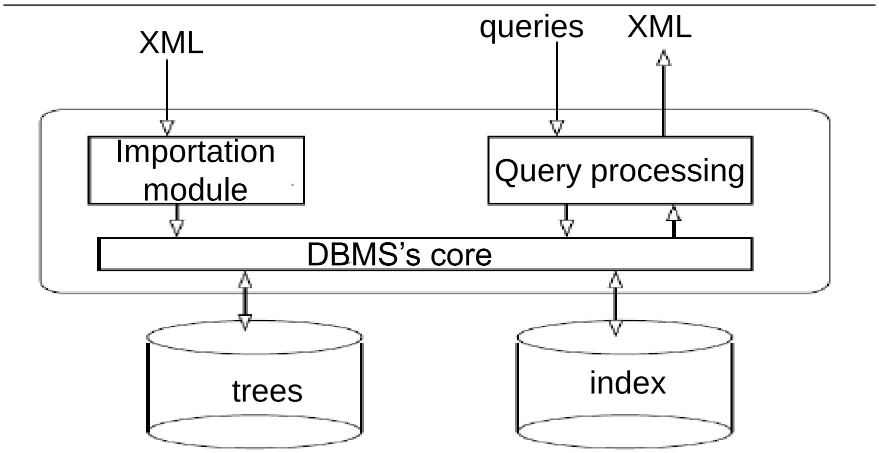
Non-native systems (2)

- Advantage
  - Reuse existant SGBDs
  - costly and difficult to build, install and configure
- Problems
  - Choice of relational schema for the storage
  - Loading XML documents
  - Translation of XML queries into SQL queries
  - Building XML results

Native systems for XML

- Objectives
  - Regrouping objects frequently used simultaneous (access locality)
  - Efficient access of XML elements (e.g. //person)
  - Efficient evaluation of path expressions (e.g. //person//name)
- Techniques
  - Labelling crucial nodes
  - Using indexes such as B-trees, R-trees...

Architecture

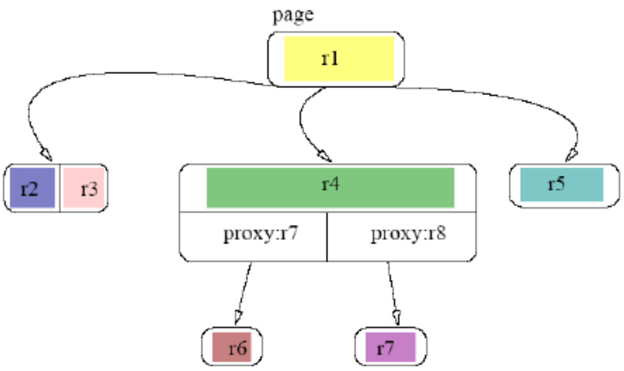


Example - Natix

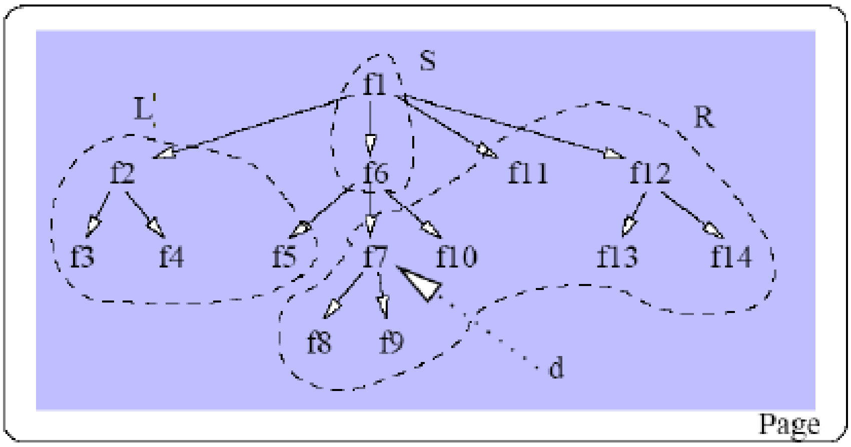
- Model
  - Logical level: XML tree
  - Physical level: XML tree+ nsupplementary odes
- Physical schema
  - Page (fixed siez) ~ {registrations (variablle size)}
  - A registration ~ a continuous memory space
  - Size of a registration <= size of a page

Registration ~ a sub-tree

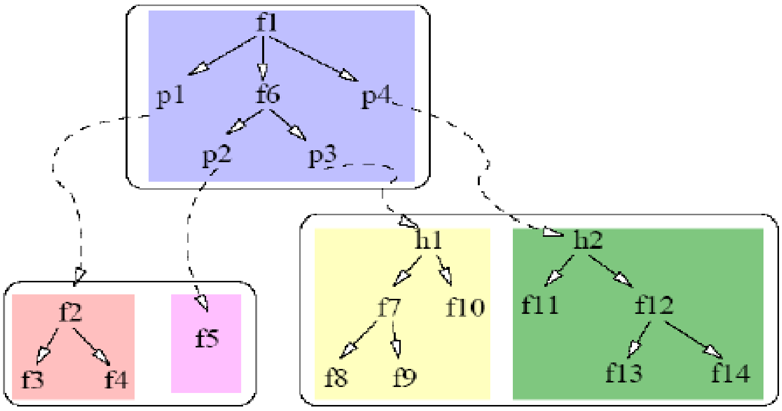
- A registration stores a sub-tree of a XML document
- Connexion between registrations through proxy



Registration

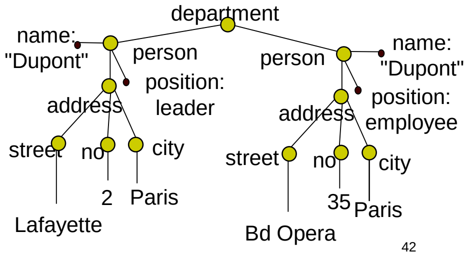


Assembling



XPath by Example

- Path expression in XML documents
- /department
- /department/person/@name
- /department/person[/@name="Dupont"]/address
- //person[/@name="Dupont"]
  /ancestor-or-self::*
- //person
  [/@name="Dupont"]
  /previous-sibling::*
- //person//city/text()



Limitations of XPath

- Impossible to express a correlation between data located in different paths
  - Can not express joins
- Impossible to creat new XML elements
  - Can not reformulate, combine several XML documents
- Limitation to
  - selection, navigation, reconstruction expression

“Users” of XPath

- XML Schema: in defining keys and uniqueness constraints
- XLink and XPointer, hyperlinks for XML
- XSLT – useful for converting from XML to other representations (e.g., HTML, PDF, SVG)
- XQuery – useful for restructuring an XML document or combining multiple documents

XML Query Language

- XQuery standard
  - W3C Proposal
  - Functional in nature
- Main features
  - Path expressions
  - FLWR expressions
  - Sorting and grouping
  - Conditional expressions
  - Extended data types
  - User defined functions

Expr = *Constructor* | *FLWRExpr* | *PathExpr* | *SortExpr* | *OrExpr* | *AndExpr* | *QuantifiedExpr* | *TypesswitchExpr* | *IfExpr* | *GeneralComp* | *ValueComp* | *NodeComp* | *OrderComp* | *InstanceofExpr* | *RangExpr* | *AdditiveExpr* | *MultiplicativeExpr* | *UnionExpr* | *IntersectExceptExpr* | *UnaryExpr* | *CastExpr*

“Iterations” in XQuery

A series of (possibly nested) FOR statements assigning the results of XPaths to variables

```
for $root in document(“http://my.org/my.xml”)
     for $sub in $root/rootElement,
            $sub2 in $sub/subElement, …
```

- Something like a template that pattern-matches, produces a tuple of bindings
- For each of these, we evaluate the WHERE and possibly output the RETURN template
- document() or doc() function specifies an input file as a URI

Example

```
for $i in doc
   (“dblp.xml”)/dblp/inproceedings[author/text() =
   “John Smith”]
return <smith-paper>
               <title>{ $i/title/text() }</title>
               <key>{ $i/@key/text() }</key>
               { $i/crossref/text() }
        </smith-paper>
```

Joins in XQuery

- □ Suppose we want to convert DBLP to DBLQ, with different ID attributes
  - an XML document map.xml with <mapping><from>*X*</from><to>*Y*</to></mapping> rows, describing correspondences
- □ This might be used to translate DBLP keys:

```
<proceedings> {
   for $p in document(“dblp.xml”)/dblp/proceedings,
       $k in $p/@key/text(),
       $m in document(“map.xml”)/root/mapping,
       $f in $m/from/text(),
       $t in $m/to/text()
   return <proc dblqID={ $t }> { $p/* } </proc>
} </proceedings>
```

- □ There are also many other uses for joins

Nesting in XQuery

- put a subquery in the return clause

```
for $u in doc(“dblp.xml”)/universities,
    $n = u/name/text()
where $u/country = “USA”
return <ms-theses-99>
            { $u/title }  {
            for $mt in $u/../mastersthesis,
                $inst in $mt/school/text()
            where $mt/year/text() = “1999”
            return $mt/title }
      </ms-theses-99>
```

Collections & Aggregation

```
let $allpapers := doc(“dblp.xml”)/dblp/article
return <article-authors>
   <count> {fn:count(fn:distinct-values($allpapers/authors)) }
   </count>
{  for $paper in doc(“dblp.xml”)/dblp/article
   let $pauth := $paper/author
   return <paper> {$paper/title}
                <count> { fn:count($pauth) } </count>
          </paper>
}  </article-authors>
```

Sorting in XQuery

- ordering the sequence of “result tuples” output by the return clause

```
for $x in doc(“dblp.xml”)/proceedings
order by $x/title/text()
return $x
```

Querying & Defining Tags

- Getting a node’s name by querying node-name()

```
for $x in document(“dblp.xml”)/dblp/*
   return node-name($x)
```

- Building elements and attributes using computed names:

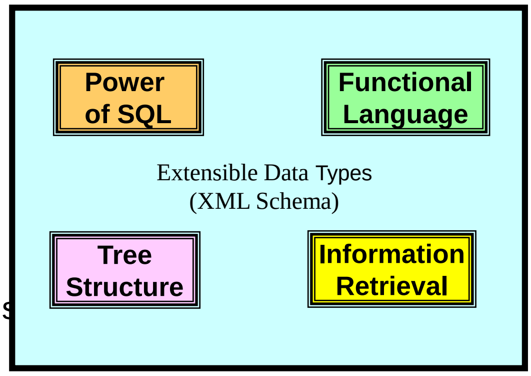
```
for $x in document(“dblp.xml”)/dblp/*,
  $year in $x/year,
  $title in $x/title/text(),
element node-name($x) {
  attribute {“year-” + $year} { $title }
}
```

XQuery Summary

- Very flexible and powerful language for XML
  - Clean and orthogonal: can always replace a collection with an expression that creates collections
  - The core is relatively clean and easy to understand
- Performs several tasks that can't be done with XPath and that are slow to program in Java:
  - Integrating information from multiple sources
  - Joins, based on correspondences of values
  - Computing count, average, etc.
  - (Also a full-fledged programming language that supports recursive, Turing-complete functions)

Power of XQuery

- XPath
- Joins
- Restructurations (« new XML")
- Typage, semantic
- Extensibility:
  - XQuery functions
  - recursive functions



Power of SQL

Functional Language

Extensible Data Types
(XML Schema)

Tree Structure

Information Retrieval

XSLT: Transforming an XML Document

- XSLT: XML Stylesheet Language Transformations
  - Companion to XSL:FO, formatting for XML
- A language for substituting structured fragments for XML content
  - Transforms single document → single document
  - Useful for XML → XML conversions, XML → HTML
  - Runs on server side (Apache Cocoon) or client-side (modern browsers)

A Functional Language for XML

- XSLT is based on a series of *templates* that match different parts of an XML document
  - There's a policy for what rule or template is applied if more than one matches (it's not what you'd think!)
  - XSLT templates can invoke other templates
  - XSLT templates can be nonterminating (beware!)
- XSLT templates are based on XPath "match"es, and we can also apply other templates (potentially to "select"ed XPaths)
  - Within each template, directly describe what should be output

An XSLT Template

- An XML document itself
- XML tags create output *OR* are XSL operations
  - All XSL tags are prefixed with “xsl” namespace
  - All non-XSL tags are part of the XML output
- Common XSL operations:
  - template with a match XPath
  - Recursive call to apply-templates, which may also select where it should be applied
- Attach to XML document with a processing-instruction:

```
<?xml version = “1.0” ?>
<?xml-stylesheet type=“text/xsl” href=“http://www.com/my.xsl” ?>
```

An Example XSLT Stylesheet

```
<xsl:stylesheet version=“1.1”>
  <xsl:template match=“/dblp”>
    <html><head>This is DBLP</head>
    <body>
       <xsl:apply-templates />
    </body>
    </html>
  </xsl:template>
  <xsl:template match=“inproceedings”>
     <h2><xsl:apply-templates select=“title” /></h2>
    <p><xsl:apply-templates select=“author”/></p>
  </xsl:template>
  …
</xsl:stylesheet>
```

XSLT Processing Model

- List of source nodes → result tree fragment(s)
- Start with root
  - Find all template rules with matching patterns from root
    - Find “best” match according to some heuristics
    - Set the current node list to be the set of things it maches
  - Iterate over each node in the current node list
    - Apply the operations of the template
    - “Append” the results of the matching template rule to the result tree structure
      - Repeat recursively if specified to by apply-templates

What If There’s More than One Match?

- Eliminate rules of lower precedence due to importing
- Break a rule into any | branches and consider separately
- Choose rule with highest computed or specified priority
- Simple rules for computing priority based on “precision”:
  - QName preceded by XPath child/axis specifier: priority 0
  - NCName preceded by child/axis specifier: priority -0.25
  - NodeTest preceded by child/axis specifier: pririty -0.5
  - else priority 0.5

Other Common Operations

- □ Iteration:

```
<xsl:for-each select="path">
</xsl:for-each>
```

- □ Conditionals:

```
<xsl:if test="./text() &lt; 'abc'">
</xsl:if>
```

- □ Copying current node and children to the result set:

```
<xsl:copy>
  <xsl:apply-templates />
</xsl:copy>
```

Creating Output Nodes

- □ Return text/attribute data (this is a default rule):

```
<xsl:template match="text()|@*">
  <xsl:value-of select="."/>
</xsl:template>
```

- □ Create an element from text (attribute is similar):

```
<xsl:element name="text()">
  <xsl:apply-templates/>
</xsl:element>
```

- □ Copy nodes matching a path

```
<xsl:copy-of select="*"/>
```

Embedding Stylesheets

- You can “import” or “include” one stylesheet from another:

  <xsl:import href=“http://www.com/my.xsl/”>

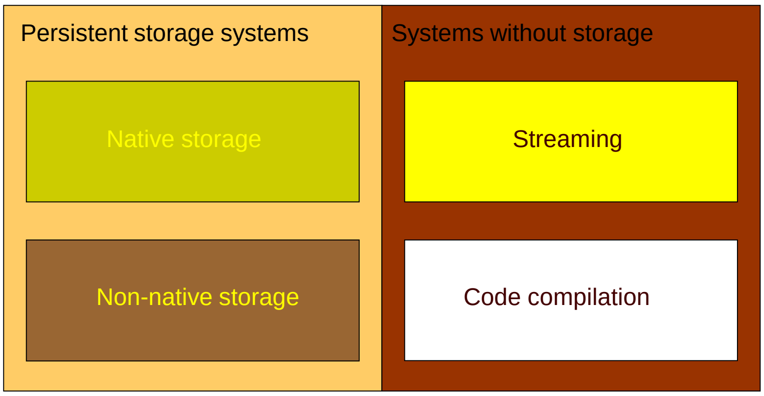
  <xsl:include href=“http://www.com/my.xsl/”>

  - “Include”: the rules get same precedence as in including template
  - “Import”: the rules are given *lower* precedence

XSLT Summary

- A very powerful, template-based transformation language for XML document → other structured document
  - Commonly used to convert XML → PDF, SVG, GraphViz DOT format, HTML, WML, …
- Primarily useful for presentation of XML or for very simple conversions
- But sometimes we need more complex operations when converting data from one source to another
  - Joins – combining and correlating information from multiple sources
  - Aggregation – computing averages, counts, etc.

XML databases



Persistent storage systems

Native storage

Non-native storage

Systems without storage

Streaming

Code compilation

XML databases

- Native storage systems for XML
  - Natix, Xyleme, Timber, OrientX, Sedna, Jungle, GeX,...
- non-native systems (based on relational model)
  - IBM, Oracle, MS, LegoDB, Rainbow, XQuark ...
- Streaming systems
  - Enosys, BEA
- Code compilation systems
  - Galax, Kawa, XDuce, CDuce, QizX,...

References

- **Link**
  - http://www.w3.org/XML/
  - http://www.w3.org/TR/
  - http://www.w3schools.com/
- **XML-Enabled Relational Databases**
  - Relational Databases for Querying XML Documents: Limitations and Opportunities. Jayavel Shanmugasundaram, H. Gang, Kristin Tufte, Chun Zhang, David DeWitt, and Jeffrey F. Naughton. VLDB 1999.
  - From XML schema to relations: a cost-based approach to XML storage , Bohannon, P.; Freire, J.; Roy, P.; Simeon, J. Page(s): 64 - 75, ICDE 2002
  - Storing and Querying Ordered XML Using a Relational Database System,, Igor Tatarinov, Stratis Viglas, Kevin Beyer, Jayavel Shanmugasundaram, Eugene Shekita, Chun Zhang, SIGMOD 2002
  - RRXS: Redundancy reducing XML storage in relations, Yi Chen, Susan Davidson (University of Pennsylvania, USA), Carmem Hara (Universidade Federal do Parana, Brazil), Yifeng Zheng (University of Pennsylvania, USA), VLDB 2003
  - A Comprehensive XQuery to SQL Translation using Dynamic Interval Encoding, David DeHaan, David Toman, Mariano P. Consens, M. Tamer Özsu, SIGMOD 2003

- **Native XML Database**
  - Timber: A native XML database, H. V. Jagadish, Shurug Al-Khalifa, Laks Lakshmanan, Andrew Nierman, Stylianos Paparizos, Jignesh Patel, Divesh Srivastava, and Yuqing Wu. Technical report, University of Michigan, April 2002.
  - TAX: A Tree Algebra for XML, H. V. Jagadish, Laks V. S. Lakshmanan, Divesh Srivastava, Keith Thompson, 8th International Workshop on Database Programming Languages, DBLP 2001, pp. 149-164
  - A Succinct Physical Storage Scheme for Efficient Evaluation of Path Queries in XML, Ning Zhang, Varun Kacholia and M. Tamer Ozsu, ICDE 2004
- **Indexes for Path Expressions**
  - DataGuides: Enabling Query Formulation and Optimization in Semistructured Databases, Roy Goldman, Jennifer Widom:. pp. 436-445, VLDB 1997
  - Index Structures for Path Expressions. Tova Milo. and Dan Suciu, ICDT, pp. 277-295, 1999
  - A Fast Index for Semistructured Data, Brian Cooper, Neal Sample, Michael J. Franklin, Gisli R. Hjaltason, Moshe Shadmon: .341-350, VLDB 2001

- Exploiting local similarity for indexing paths in graph-structured data, Kaushik, R.; Shenoy, P.; Bohannon, P.; Gudes, E., Page(s): 129 - 140, ICDE 2002
- Covering Indexes for Branching Path Queries, Raghav Kaushik, Philip Bohannon, Jeffrey Naughton, Henry Korth, SIGMOD 2002
- APEX: An Adaptive Path Index for XML data, Chin-Wan Chung, Jun-Ki Min, Kyuseok Shim, SIGMOD 2002
- Efficient Algorithms for Processing XPath Queries, Georg Gottlob, Christoph Koch, Reinhard Pichler (Technische Universität Wien, Austria), VLDB 2002
- Updates for Structure Indexes, Raghav Kaushik (University of Wisconsin - Madison, U.S.A.), Philip Bohannon (Lucent Technologies - Bell Labs, U.S.A.), Jeffrey F. Naughton (University of Wisconsin - Madison, U.S.A.), Pradeep Shenoy (University of Washington - Seattle, U.S.A.), , VLDB 2002
- Covering Indexes for XML Queries: Bisimulation - Simulation = Negation, Prakash Ramanan (Wichita State University, USA), VLDB 2003
- D(K)-Index: An Adaptive Structural Summary for Graph-Structured Data, Qun Chen, Andrew Lim, Kian Win Ong, SIGMOD 2003

- **XML Query Optimization**
  - Minimization of Tree Pattern Queries, Sihem Amer-Yahia, SungRan Cho, Laks V. S. Lakshmanan, Divesh Srivastava. SIGMOD Conference 2001
  - Efficient Algorithms for Minimizing Tree Pattern Queries, Prakash Ramanan, SIGMOD 2002
  - On the minimization of Xpath queries, Sergio Flesca, Filippo Furfaro (Università della Calabria, Italy), Elio Masciari (ICAR - CNR, Italy), VLDB, 2003

- **General XML Query Evaluation**
  - From Tree Patterns to Generalized Tree Patterns: On Efficient Evaluation of XQuery, Zhimin Chen (Univ. of British Columbia, Canada), H.V. Jagadish (Univ. of Michigan, USA) , Laks V. S. Lakshmanan (Univ. of British Columbia, Canada), Stelios Paparizos (Univ. of Michigan, USA)
  - Mixed Mode XML Query Processing, Alan Halverson, Josef Burger, Leonidas Galanis, Ameet Kini, Rajasekar Krishnamurthy, Ajith Nagaraja Rao, Feng Tian, Stratis D. Viglas, Yuan Wang, Jeffrey F. Naughton, David J. DeWitt (University of Wisconsin-Madison, USA), VLDB 2003
  - ViST: A Dynamic Index Method for Querying XML Data by Tree Structures, Haixun Wang, Sanghyun Park, Wei Fan, Philip Yu, SIGMOD 2003
  - Projecting XML Documents, Amélie Marian (Columbia University, USA), Jérôme Siméon (Bell Laboratories, USA), VLDB 2003
  - Efficient Processing of Expressive Node-Selecting Queries on XML Data in Secondary Storage: A Tree Automata-based Approach, Christoph Koch (University of Edinburgh, UK), VLDB 2003