ECS 165B: Database System Implementation
Lecture 16

UC Davis
May 3, 2010

Acknowledgements: portions based on slides Zack Ives (which incorporate some content due to Susan Davidson and Raghu Ramakrishnan)

Class Agenda

- Last time:
  - Query optimization, continued
- Today:
  - Introduction to XML databases
- Reading
  - Chapter 15 of Ramakrishnan and Gehrke (or Chapter 14 of Silberschatz et al)

Announcements

DavisDB Part 3: out ~~Sunday night~~ Tuesday night, due ~~Sunday 5/8~~ Tuesday 5/10 @11:59pm

XML: A Semi-Structured Data Model

Why XML?

XML is the confluence of several factors:

- The Web needed a more declarative format for data
- Documents needed a mechanism for extended tags
- Database people needed a more flexible interchange format
- "Lingua franca" of data
- It's parsable even if we don't know what it means!

Original expectation:

- The whole web would go to XML instead of HTML (cf. Xyleme SA)

Today's reality:

- Not so… but XML is used all over "under the covers"

Why DB People Like XML

Can get data from all sorts of sources

- Allows us to touch data we don't own!
- This was actually a huge change in the DB community

Interesting relationships with DB techniques

- Useful to do relational-style operations
- Leverages ideas from object-oriented, semistructured data

Blends schema and data into one format

- Unlike relational model, where we need schema first
- … But too little schema can be a drawback, too!

XML Anatomy

```
<?xml version="1.0" encoding="ISO-8859-1" ?>   ← Processing Instr.
<dblp>   ← Open-tag
 <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>   ← Element
 </mastersthesis>
 <article mdate="2002-01-03" key="tr/dec/SRC1997-018">
  <editor>Paul R. McJones</editor>   ← Attribute
  <title>The 1995 SQL Reunion</title>
  <journal>Digital System Research Center Report</journal>
  <volume>SRC1997-018</volume>   ← Close-tag
  <year>1997</year>
  <ee>db/labs/dec/SRC1997-018.html</ee>
  <ee>http://www.mcjones.org/System_R/SQL_Reunion_95/</ee>
 </article>
```

Well-Formed XML

A legal XML document – fully parsable by an XML parser

- All open-tags have matching close-tags (unlike so many HTML documents!), or a special:

  <tag/> shortcut for empty tags (equivalent to <tag></tag>)

- Attributes (which are unordered, in contrast to elements) only appear once in an element

- There's a single root element

- XML is case-sensitive

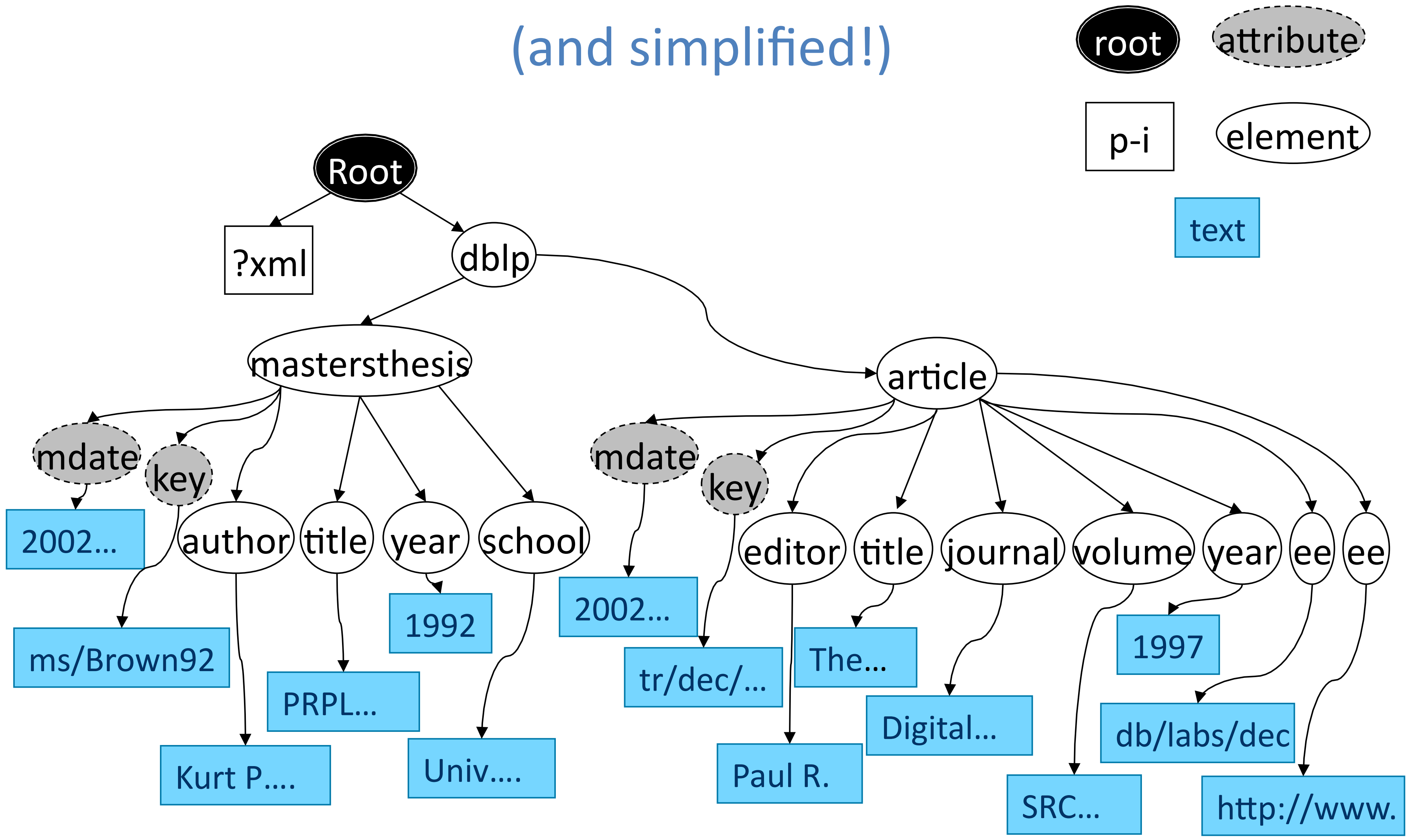
XML as a Data Model

XML “information set” includes 7 types of nodes:

- Document (root)
- Element
- Attribute
- Processing instruction
- Text (content)
- Namespace
- Comment

XML data model includes this, plus typing info, plus order info and a few other things

XML Data Model Visualized (and simplified!)



What Does XML Do?

- Serves as a document format (super-HTML)
  - Allows custom tags (e.g., used by MS Word, OpenOffice)
  - Supplement it with stylesheets (XSL) to define formatting
- Data exchange format (must agree on terminology)
- Marshalling and unmarshalling data in SOAP and Web Services

XML as a Super-HTML (MS Word)

```
<h1 class="Section1"><a name="_top“/>
  ECS165B: Database Systems Implementation
</h1>
<h2 class="Section1">Spring 2010</h2>
<p class="MsoNormal">
  <place>1 Wellman</place>, Monday/Wednesday/Friday
  <time Hour="16" Minute="10">4:10PM – 5:00PM
  </time>
</p>
```

XML Easily Encodes Relations

Student-course-grade

sid	serno	exp-grade
1	570103	B
23	550103	A

```
<student-course-grade>
  <tuple><sid>1</sid><serno>570103</serno>
    <exp-grade>B</exp-grade></tuple>
  <tuple><sid>23</sid><serno>550103</serno>
    <exp-grade>A</exp-grade></tuple>
</student-course-grade>
```

But XML is More Flexible… “Non-First-Normal-Form” (NF^2)

```
<parents>
  <parent name=“Jean” >
   <son>John</son>
   <daughter>Joan</daughter>
   <daughter>Jill</daughter>
  </parent>
  <parent name=“Feng”>
   <daughter>Felicity</daughter>
  </parent>
…
```

Coincides with “semi-structured data”, invented by DB people at Penn and Stanford

XML and Code

- Web Services (.NET, recent Java web service toolkits) are using XML to pass parameters and make function calls
  - Why?
    - Easy to be forwards-compatible
    - Easy to read over and validate (?)
    - Generally firewall-compatible
  - Drawbacks? XML is a verbose and inefficient encoding!
- XML is used to represent:
  - SOAP: the “envelope” that data is marshalled into
  - XML Schema: gives some typing info about structures being passed
  - WSDL: the IDL (interface def language)
  - UDDI: provides an interface for querying about web services

Integrating XML: What If We Have Multiple Sources with the Same Tags?

- *Namespaces* allow us to specify a context for different tags
- Two parts:
  - Binding of namespace to URI
  - Qualified names

```
<root xmlns="http://www.first.com/aspace"
  xmlns:otherns="…">
 <tag xmlns:myns="http://www.fictitious.com/mypath">
  <thistag>is in the default namespace (aspace)</thistag>
  <myns:thistag>is in myns</myns:thistag>

  <otherns:thistag>is a different tag in otherns
  </otherns:thistag>
 </tag>
</root>
```

XML Isn't Enough on Its Own

It's too unconstrained for many cases!

- How will we know when we're getting garbage?
- How will we query?
- How will we understand what we got?

We also need:

Some idea of the structure

- Our focus next

Presentation, in some cases – XSL(T)

- We'll talk about this soon

Some way of interpreting the tags…?

- We'll talk about this later in the quarter

Structural Constraints: Document Type Definitions (DTDs)

The DTD is an EBNF grammar defining XML structure

- XML document specifies an associated DTD, plus the root element
- DTD specifies children of the root (and so on)

DTD defines special significance for attributes:

- IDs – special attributes that are analogous to keys for elements
- IDREFs – references to IDs
- IDREFS – a nasty hack that represents a list of IDREFs

An Example DTD

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)>
```

…

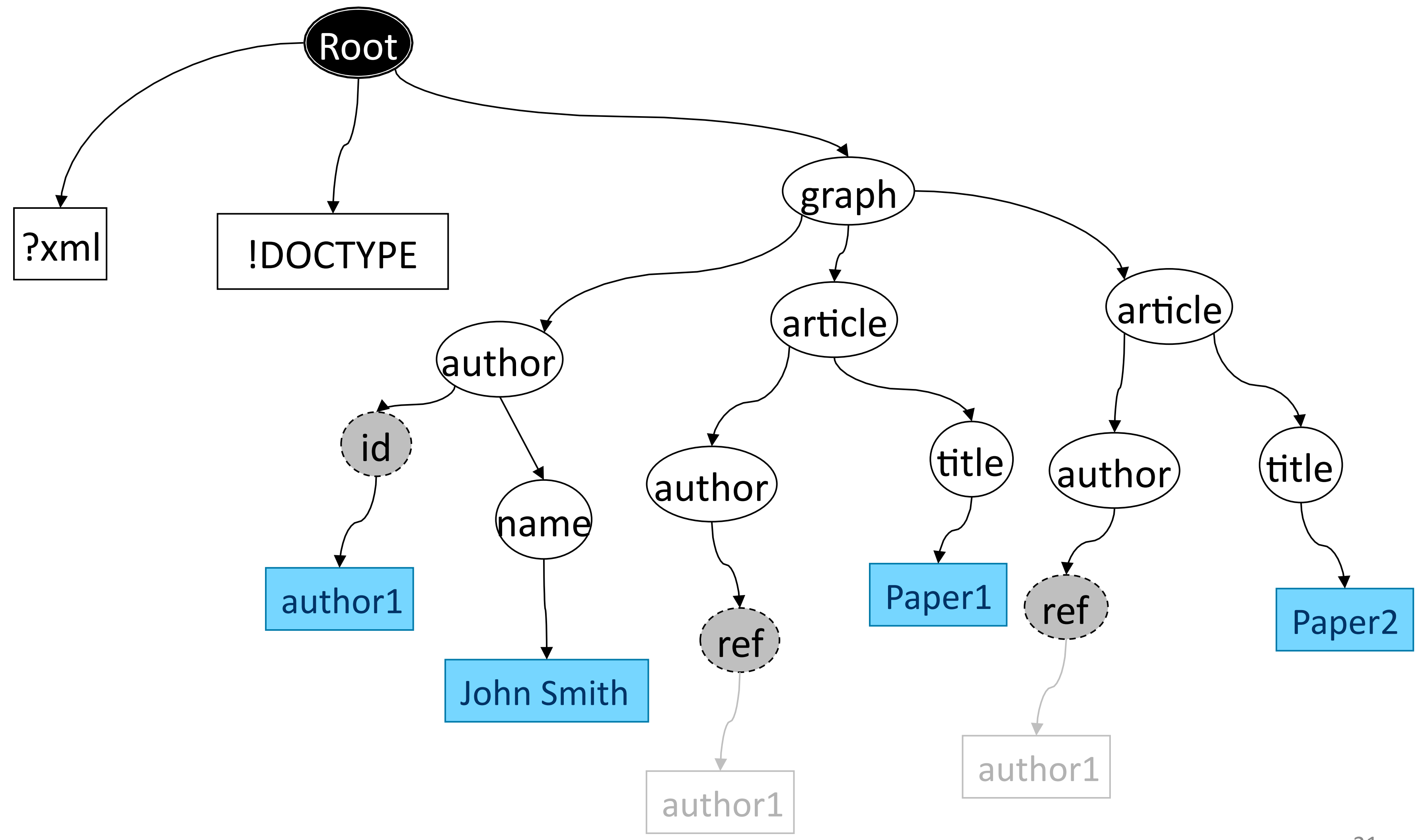
Example use of DTD in XML file:

```
<?xml version="1.0" encoding="ISO-8859-1" ?>

<!DOCTYPE dblp SYSTEM “my.dtd">

<dblp>…
```

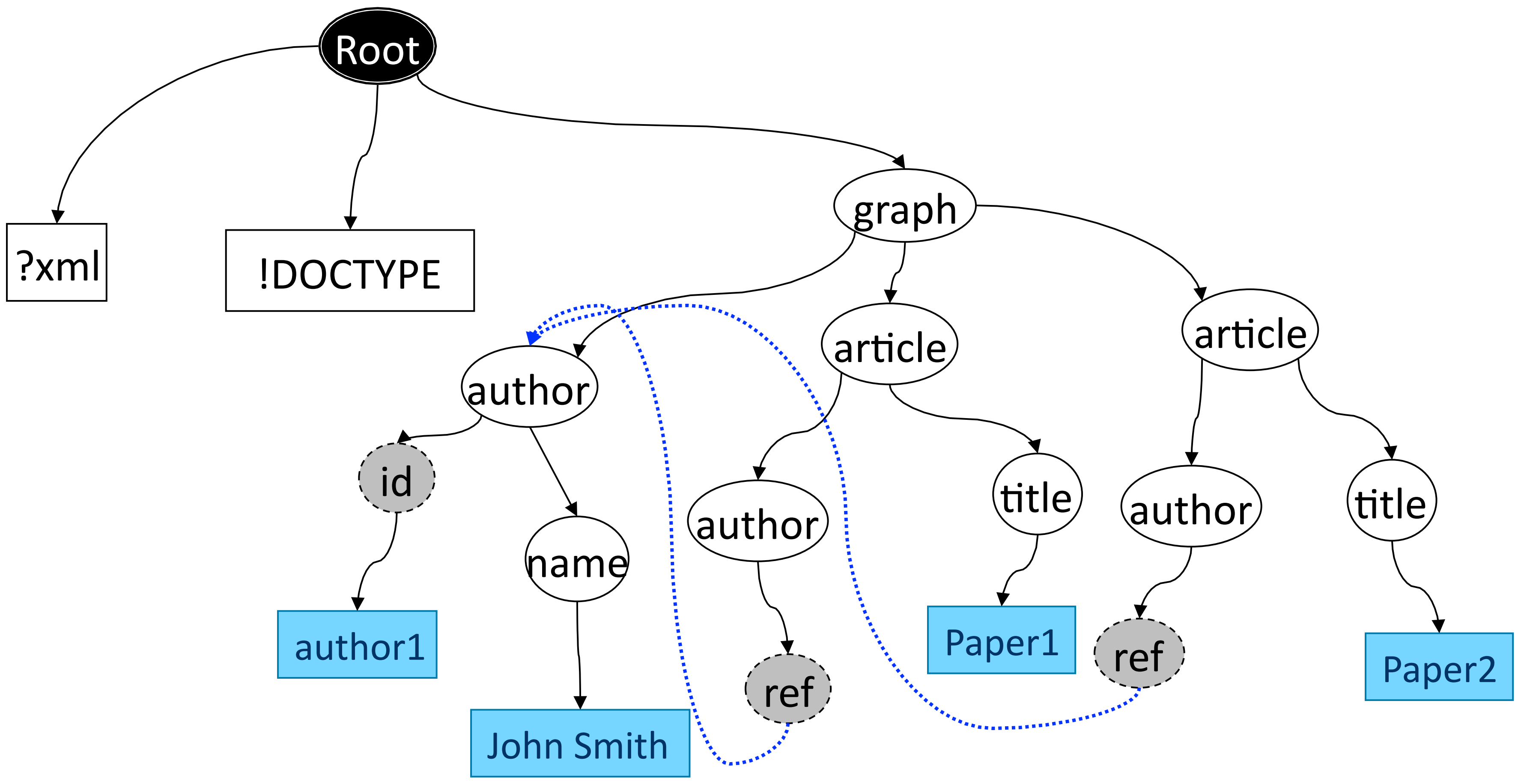
Representing Graphs and Links in XML

```
<?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



Graph Data Model



DTDs Aren’t Expressive Enough

DTDs capture grammatical structure, but have some drawbacks:

- Not themselves in XML – inconvenient to build tools for them
- Don’t capture database's datatype domains
- No way of defining OO-like inheritance

XML Schema

Aims to address the shortcomings of DTDs

- XML syntax (verbose!)
- Can define keys using XPaths
- Type subclassing that's more complex than in a programming language
  - Programming languages don't consider order of member variables!
  - Subclassing "by extension" and "by restriction"
- … And, of course, domains and built-in datatypes

Basics of XML Schema

Need to use the XML Schema namespace (generally named `xsd`)

- simpleTypes are a way of restricting domains on scalars
  - Can define a simpleType based on integer, with values within a particular range
- complexTypes are a way of defining element/attribute structures
  - Basically equivalent to !ELEMENT, but more powerful
  - Specify sequence, choice between child elements
  - Specify minOccurs and maxOccurs (default 1)
- Must associate an element/attribute with a simpleType, or an element with a complexType

Simple Schema 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>
</xsd:schema>
```

Designing an XML Schema/DTD

Not as formalized as relational data design

- We can still use ER diagrams to break into entity, relationship sets
- ER diagrams have extensions for “aggregation” – treating smaller diagrams as entities – and for composite attributes
- Note that often we already have our data in relations and need to design the XML schema to export them!

Generally orient the XML tree around the “central” objects

Big decision: element vs. attribute

- Element if it has its own properties, or if you *might* have more than one of them
- Attribute if it is a single property – or perhaps not!

Summary: XML as a Data Model

XML is a non-first-normal-form (NF^2) representation

- Can represent documents, data
- Standard data exchange format
- Several competing schema formats – esp., DTD and XML Schema – provide typing information