XML and DTD

Introduction to Databases

CompSci 316 Spring 2019

DUKE
COMPUTER SCIENCE

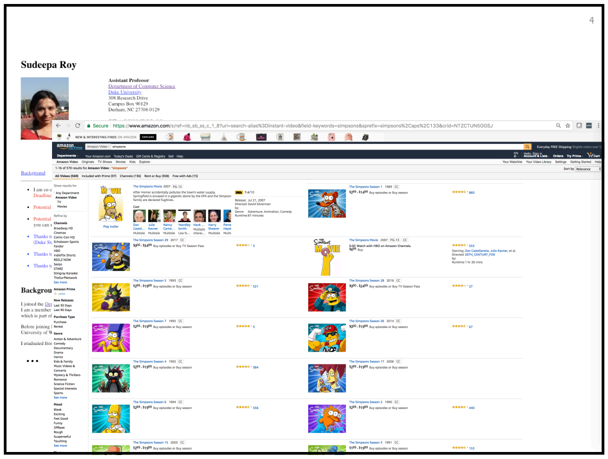
Announcements (Thu. Feb. 28)

- Homework #2 due today (except 1 & 2)
  - Late submissions with 5% penalty per hour
- Homework #3 to be assigned soon
- Project milestone #1 feedback to be emailed by next class

Structured vs. unstructured data

- Relational databases are highly structured
  - All data resides in tables
  - You must define schema before entering any data
  - Every row confirms to the table schema
  - Changing the schema is hard and may break many things
- Texts are highly unstructured
  - Data is free-form
  - There is no pre-defined schema, and it's hard to define any schema
  - Readers need to infer structures and meanings

What's in between these two extremes?



Sudeepa Roy

Assistant Professor
Department of Computer Science
Duke University
308 Research Drive
Campus Box 90129
Durham, NC 27708-0129

Background

- I am co-c… Deadline…
- Potential…
- Potential… you can s…
- Thanks t… (Duke St…
- Thanks t…
- Thanks t…

Backgrou…

I joined the Dep… I am a member… which is part of…

Before joining… University of W…

I graduated fro…

• • •

Secure https://www.amazon.com/s/ref=nb_sb_ss_c_1_8?url=search-alias%3Dinstant-video&field-keywords=simpsons&sprefix=simpsons%2Caps%2C133&crid=NTZCTUN5GGSJ

1-16 of 570 results for Amazon Video : "simpsons"

All Videos (569) Included with Prime (97) Channels (136) Rent or Buy (308) Free with Ads (15)

The Simpsons Movie 2007 PG-13 — After Homer accidentally pollutes the town's water supply, Springfield is encased in a gigantic dome by the EPA and the Simpson family are declared fugitives. — 7.4/10 — Release: Jul 21, 2007; Directed David Silverman; Genre: Adventure, Animation, Comedy; Runtime: 87 minutes

The Simpsons Season 1 1989 CC — $2.99 - $14.99 Buy episodes or Buy season — 860

The Simpsons Season 29 2017 CC — $0.00 - $34.99 Buy episodes or Buy TV Season Pass — 3

The Simpsons Movie 2007 PG-13 CC — 0.00 Watch with HBO on Amazon Channels. $6.99 Buy — 553 — Starring: Dan Castellaneta, Julie Kavner, et al. Directed 20TH_CENTURY_FOX; Runtime: 1 hr 26 mins

The Simpsons Season 5 1993 CC — $2.99 - $19.99 Buy episodes or Buy season — 321

The Simpsons Season 28 2016 CC — $0.00 - $24.99 Buy episodes or Buy TV Season Pass — 27

The Simpsons Season 7 1995 CC — $2.99 - $19.99 Buy episodes or Buy season — 3

The Simpsons Season 26 2014 CC — $0.00 - $19.99 Buy episodes or Buy season — 67

The Simpsons Season 4 1992 CC — $2.99 - $19.99 Buy episodes or Buy season — 384

The Simpsons Season 17 2006 CC — $2.99 - $19.99 Buy episodes or Buy season

The Simpsons Season 6 1994 CC — $2.99 - $19.99 Buy episodes or Buy season — 556

The Simpsons Season 2 1990 CC — $2.99 - $19.99 Buy episodes or Buy season — 440

The Simpsons Season 15 2003 CC — $2.99 - $19.99 Buy episodes or Buy season

The Simpsons Season 3 1991 CC — $1.99 - $19.99 Buy episodes or Buy season — 153

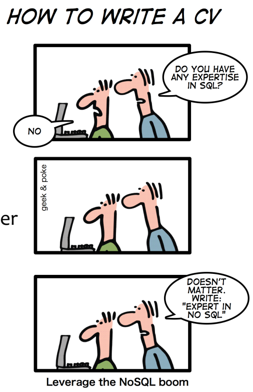
Departments
Your Amazon.com Today's Deals Gift Cards & Registry Sell Help
Amazon Video Originals TV Shows Movies Kids Explore
Your Watchlist Your Video Library Settings Getting Started Help
Sort by Relevance
Show results for
Any Department
Amazon Video
TV
Movies
Refine by
Channels
Amazon Prime
New Releases
Purchase Type
Genre
Mood
See more

Semi-structured data

- Observation: most data have some structure, e.g.:
  - Book: chapters, sections, titles, paragraphs, references, index, etc.
  - Item for sale: name, picture, price (range), ratings, promotions, etc.
  - Web page: HTML
- Ideas:
  - Ensure data is "well-formatted"
  - If needed, ensure data is also "well-structured"
    - But make it easy to define and extend this structure
  - Make data "self-describing"

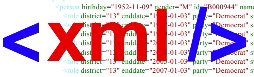
SQL vs. NoSQL

- SQL's rigidity in face of semi-structured data is one of the reasons behind the rise of (some) NoSQL systems
  - NoSQL has other motivations, which we hope to get to in a later part of this course



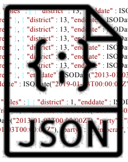
HOW TO WRITE A CV
DO YOU HAVE ANY EXPERTISE IN SQL?
NO
geek & poke
DOESN'T MATTER. WRITE: "EXPERT IN NO SQL"
Leverage the NoSQL boom

Our roadmap thru the NoSQL land



```
<congress>
  <people>
    <person birthday="1952-11-09" gender="M" id="B000944" name="Sherrod Brown">
      <role district="13" enddate="19[illegible]-01-03" pa[illegible]y="Democrat" startdate="1993-01-05" state="OH" type="rep"/>
      <r[illegible]dist[illegible]en[illegible]e="1[illegible]-01-03" p[illegible]Democrat" startdate="1995-01-04" state="OH" type="rep"/>
      [illegible]dist[illegible]="1[illegible]ndd[illegible]="1[illegible]-01-03" [illegible]ty=[illegible]rat" startdate="1997-01-07" state="OH" type="rep"/>
      [illegible]e dist[illegible]="1[illegible]endd[illegible]="2[illegible]-01-03" [illegible]rty="Demo[illegible]" startdate="1999-01-06" state="OH" type="rep"/>
      [illegible]dist[illegible]="1[illegible]endd[illegible]="2[illegible]-01-03"[illegible]arty=[illegible]ocrat" startdate="2001-01-03" state="OH" type="rep"/>
      <ro[illegible] district="13" enddate="200[illegible]-01-03" party="Democrat" startdate="2003-01-07" state="OH" type="rep"/>
      <role district="13" enddate="2007-01-03" party="Democrat" startdate="2005-01-04" state="OH" type="rep"/>
      <role enddate="2013-01-03" party="Democrat" startdate="2007-01-04" state="OH" type="sen"/>
      <role current="1" enddate="2019-01-03" party="Democrat" startdate="2013-01-03" state="OH" type="sen"/>
    </person>
    <person birthday="1958-10-13" gender="F" id="C000127" name="Maria Cantwell">
```

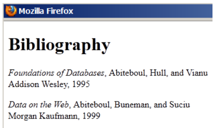
- But can’t relational databases do XML?



```
{ "_id" : "B000944", "birthday" : ISODate("1952-11-09T00:00:00Z"), "gender" : "M", "name" : "Sherrod Brown", "[illegible]les" : [ { "district" : 13, "[illegible]date" : ISODate("1995-01-
03T00:00:00Z"), "party" : "Democrat", "startdate" : ISODate("1993-01-05T00:00:00Z"), "state" : "OH", "type" : "re[illegible] }, { "district" : 13, "end[illegible]te" : ISODate("1997-01-
03T00:00:00Z"), "party" : "Democrat", "startdate" : ISODate("1995-01-04T00:00:00Z"), "state" : "OH", "type" : "re[illegible] }, { "district" : 13, "end[illegible]te" : [illegible]ODate("1999-01-
03T00:00:00Z"), "party" : "Democrat", "startdate" : ISODate("1997-01-07T00:00:00Z"), "state" : "OH", "type" : "re[illegible] }, { "dis[illegible]ict" : 13, "en[illegible]ate("2001-01-
03T00:00:00Z"), "party" : "Democrat", "startdate" : ISODate("1999-01-06T00:00:00Z"), "state" : "OH", "type" : "re[illegible] }, { "di[illegible]rict" : 13, "en[illegible]ate" : ISO[illegible]ate("2003-01-
03T00:00:00Z"), "party" : "Democrat", "startdate" : ISODate("2001-01-03T00:00:00Z"), "state" : "OH", "type" : "re[illegible] }, { "di[illegible]rict" : [illegible], "en[illegible]ate" : ISO[illegible]ate("2005-01-
03T00:00:00Z"), "party" : "Democrat", "startdate" : ISODate("2003-01-07T00:00:00Z"), "state" : "OH", "type" : "re[illegible] }, { [illegible]trict" : 13, "end[illegible]e" : ISO[illegible]ate("2007-01-
03T00:00:00Z"), "party" : "Democrat", "startdate" : ISODate("2005-01-04T00:00:00Z"), "state" : "OH", "type" : "re[illegible] }, { "en[illegible]date" [illegible]SODa[illegible]("2013-01[illegible]03T00:00:00Z"), "party" :
"Democrat", "startdate" : ISODate("2007-01-04T00:00:00Z"), "state" : "OH", "type" : "sen" }, { "current" : 1, "end[illegible]te" : ISO[illegible]te("2[illegible]19-01-[illegible]3T00:00:0[illegible]Z"), "party" : "Democrat",
"startdate" : ISODate("2013-01-03T00:00:00Z"), "state" : "OH", "type" : "sen" } ] }
{ "_id" : "C000127", "birthday" : ISODate("1958-10-13T00:00:00Z"), "gender" : "F", "name" : "Maria Cantwell", "[illegible]les" : [ { "district" : 1, "enddate" : I[illegible]ODate("1995-01-
03T00:00:00Z"), "party" : "Democrat", "startdate" : ISODate("1993-01-05T00:00:00Z"), "state" : "WA", "type" : "r[illegible]03T00:00:00Z"), "party" :
"Democrat", "startdate" : ISODate("2001-01-03T00:00:00Z"), "state" : "WA", "type" : "sen" }, { "enddate" : ISODa[illegible]("2013[illegible]01-0[illegible]T00:[illegible]Z"), "p[illegible]rty" [illegible] "Democrat", "startdate" :
ISODate("2007-01-04T00:00:00Z"), "state" : "WA", "type" : "sen" }, { "current" : 1, "enddate" : ISODate("2019-01[illegible]03T00:[illegible]Z"), [illegible]arty" : [illegible]ocra[illegible], "startdate" :
ISODate("2013-01-03T00:00:00Z"), "state" : "WA", "type" : "sen" } ] }
```

HTML: language of the Web

```
<h1>Bibliography</h1>
<p><i>Foundations of Databases</i>,
Abiteboul, Hull, and Vianu
<br>Addison Wesley, 1995
<p>…
```



Bibliography

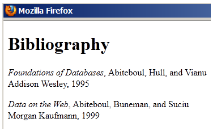
Foundations of Databases, Abiteboul, Hull, and Vianu
Addison Wesley, 1995

Data on the Web, Abiteboul, Buneman, and Suciu
Morgan Kaufmann, 1999

- It’s mostly a “formatting” language
- It mixes presentation and content
  - Hard to change presentation (say, for different displays)
  - Hard to extract content

XML: eXtensible Markup Language

```
<bibliography>
  <book>
    <title>Foundations of Databases</title>
    <author>Abiteboul</author>
    <author>Hull</author>
    <author>Vianu</author>
    <publisher>Addison Wesley</publisher>
    <year>1995</year>
  </book>
  <book>…</book>
</bibliography>
```



Bibliography

Foundations of Databases, Abiteboul, Hull, and Vianu
Addison Wesley, 1995

Data on the Web, Abiteboul, Buneman, and Suciu
Morgan Kaufmann, 1999

- Text-based
- Capture data (content), not presentation
- Data self-describes its structure
  - Names and nesting of tags have meanings!

Other nice features of XML

- Portability: Just like HTML, you can ship XML data across platforms
  - Relational data requires heavy-weight API’s
- Flexibility: You can represent any information (structured, semi-structured, documents, …)
  - Relational data is best suited for structured data
- Extensibility: Since data describes itself, you can change the schema easily
  - Relational schema is rigid and difficult to change

XML terminology

```
<bibliography>
  <book ISBN="ISBN-10" price="80.00">
    <title>Foundations of Databases</title>
    <author>Abiteboul</author>
    <author>Hull</author>
    <author>Vianu</author>
    <publisher>Addison Wesley</publisher>
    <year>1995</year>
  </book>…
</bibliography>
```

- Tag names: book, title, …
- Start tags: <book>, <title>, …
- End tags: </book>, </title>, …
- An element is enclosed by a pair of start and end tags: <book>…</book>
  - Elements can be nested: <book>…<title>…</title>…</book>
  - Empty elements: <is_textbook></is_textbook>
    - Can be abbreviated: <is_textbook/>
- Elements can also have attributes: <book ISBN="…" price="80.00">

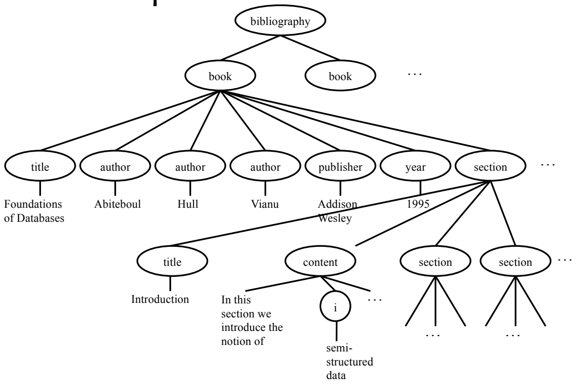
☞Ordering generally matters, except for attributes

Well-formed XML documents

A well-formed XML document

- Follows XML lexical conventions
  - Wrong: <section>We show that x < 0…</section>
  - Right: <section>We show that x &lt; 0…</section>
    - Other special entities: > becomes &gt; and & becomes &amp;
- Contains a single root element
- Has properly matched tags and properly nested elements
  - Right: <section>…<subsection>…</subsection>…</section>
  - Wrong: <section>…<subsection>…</section>…</subsection>

A tree representation



More XML features

- Processing instructions for apps: <? … ?>
  - An XML file typically starts with a version declaration using this syntax: <?xml version="1.0"?>
- Comments: <!-- Comments here -->
- CDATA section: <![CDATA[Tags: <book>,…]]>
- ID's and references
  - ID value must start with a non-digit

```
<person id="o12"><name>Homer</name>…</person>
<person id="o34"><name>Marge</name>…</person>
<person id="o56" father="o12" mother="o34">
  <name>Bart</name>…
</person>…
```

- Namespaces allow external schemas and qualified names

```
<myCitationStyle:book xmlns:myCitationStyle="http://…/mySchema">
    <myCitationStyle:title>…</myCitationStyle:title>
    <myCitationStyle:author>…</myCitationStyle:author>…
</book>
```

- And more…

Now for some more structure…

https://commons.wikimedia.org/wiki/File:Hundertwasser_04.jpg

Valid XML documents

- A valid XML document conforms to a Document Type Definition (DTD)
  - A DTD is optional
  - A DTD specifies a grammar for the document
    - Constraints on structures and values of elements, attributes, etc.
- Example

```
<!DOCTYPE bibliography [
        <!ELEMENT bibliography (book+)>
        <!ELEMENT book (title, author*, publisher?, year?, section*)>
        <!ATTLIST book ISBN ID #REQUIRED>
        <!ATTLIST book price CDATA #IMPLIED>
        <!ELEMENT title (#PCDATA)>
        <!ELEMENT author (#PCDATA)>
        <!ELEMENT publisher (#PCDATA)>
        <!ELEMENT year (#PCDATA)>
        <!ELEMENT i (#PCDATA)>
        <!ELEMENT content (#PCDATA|i)*>
        <!ELEMENT section (title, content?, section*)>
]>
```

DTD explained

<!DOCTYPE bibliography [

└→ bibliography is the root element of the document

<!ELEMENT bibliography (book+)>

→ One or more

└→ bibliography consists of a sequence of one or more book elements

<!ELEMENT book (title, author*, publisher?, year?, section*)>

→ Zero or one

→ Zero or more

└→ book consists of a title, zero or more authors, an optional publisher, and zero or more section's, in sequence

<!ATTLIST book ISBN ID #REQUIRED>

└→ book has a required ISBN attribute which is a unique identifier

<!ATTLIST book price CDATA #IMPLIED>

└→ book has an optional (#IMPLIED) price attribute which contains character data

Other attribute types include
IDREF (reference to an ID),
IDREFS (space-separated list of references),
enumerated list, etc.

```
<bibliography>
  <book ISBN="ISBN-10" price="80.00">
    <title>Foundations of Databases</title>
    <author>Abiteboul</author>
    <author>Hull</author>
    <author>Vianu</author>
    <publisher>Addison Wesley</publisher>
    <year>1995</year>
  </book>…
</bibliography>
```

DTD explained (cont'd)

<!ELEMENT title (#PCDATA)>
<!ELEMENT author (#PCDATA)>
<!ELEMENT publisher (#PCDATA)>
<!ELEMENT year (#PCDATA)>
<!ELEMENT i (#PCDATA)>

└→ author, publisher, year, and i contain parsed character data

<!ELEMENT content (#PCDATA|i)*>

└→ content contains mixed content: text optionally interspersed with i elements

<!ELEMENT section (title, content?, section*)>

└→ Recursive declaration:
Each section begins with a title,
followed by an optional content,
and then zero or more
(sub) section's

]>

PCDATA is text that will be parsed

- &lt; etc. will be parsed as entities
- Use a CDATA section to include text verbatim

```
<section><title>Introduction</title>
  <content>In this section we introduce
    the notion of <i>semi-structured data</i>…
  </content>
  <section><title>XML</title>
    <content>XML stands for…</content>
  </section>
  <section><title>DTD</title>
    <section><title>Definition</title>
      <content>DTD stands for…</content>
    </section>
    <section><title>Usage</title>
      <content>You can use DTD to…</content>
    </section>
  </section>
</section>
```

19

Using DTD

- DTD can be included in the XML source file

  -
    ```
    <?xml version="1.0"?>
    <!DOCTYPE bibliography [
      … …
    ]>
    <bibliography>
    … …
    </bibliography>
    ```
- DTD can be external

  -
    ```
    <?xml version="1.0"?>
    <!DOCTYPE bibliography SYSTEM "../dtds/bib.dtd">
    <bibliography>
    … …
    </bibliography>
    ```

  -
    ```
    <?xml version="1.0"?>
    <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
                   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
    <html>
    … …
    </html>
    ```

20

Annoyance: content grammar

- Consider this declaration:

```
<!ELEMENT pub-venue
  ( (name, address, month, year) |
    (name, volume, number, year) )>
```

  - "|" means "or"
- Syntactically legal, but won't work
  - Because of SGML compatibility issues
  - When looking at name, a parser would not know which way to go without looking further ahead
  - Requirement: content declaration must be "deterministic" (i.e., no look-ahead required)
  - Can we rewrite it into an equivalent, deterministic one?
- Also, you cannot nest mixed content declarations
  - Illegal: <!ELEMENT Section (title, (#PCDATA|i)*, section*)>

21

Annoyance: element name clash

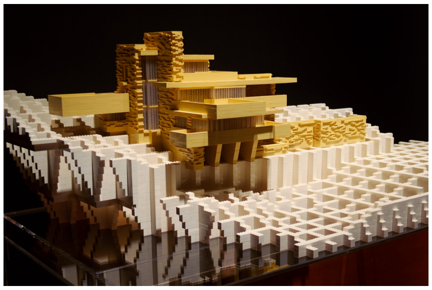
- Suppose we want to represent book titles and section titles differently
  - Book titles are pure text: (#PCDATA)
  - Section titles can have formatting tags: (#PCDATA|i|b|math)*
- But DTD only allows one title declaration!
- Workaround: rename as book-title and section-title?
  - Not nice—why can't we just infer a title's context?

22

Annoyance: lack of type support

- Too few attribute types: string (CDATA), token (e.g., ID, IDREF), enumeration (e.g., (red|green|blue))
  - What about integer, float, date, etc.?
- ID not typed
  - No two elements can have the same id, even if they have different types (e.g., book vs. section)
- Difficult to reuse complex structure definitions
  - E.g.: already defined element E1 as (blah, bleh, foo?, bar*, …); want to define E2 to have the same structure
  - Parameter entities in DTD provide a workaround
    - <!ENTITY % E.struct '(blah, bleh, foo?, bar*, …)'>
    - <!ELEMENT E1 %E.struct;>
    - <!ELEMENT E2 %E.struct;>
  - Something less "hacky"?

23



Want even more
structure support?

http://thenewsherald.com/content/articles/2012/11/10/entertainment/doc509deb9f9207c2452179065.jpg

24

XML Schema

- A more powerful way of defining the structure and constraining the contents of XML documents
  - Supports a rich set of types and user-defined types/structures
  - Supports notions of keys and foreign keys
- An XML Schema definition is itself an XML document
  - Typically stored as a standalone .xsd file
  - XML (data) documents refer to external .xsd files

```
<?xml version="1.0"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">

<xs:element name="bibliography">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="book" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>
  <xs:key name="bookKey">
    <xs:selector xpath="./book"/>
    <xs:field xpath="@ISBN"/>
  </xs:key>
  <xs:keyref name="bookForeignKey" refer="bookKey">
    <xs:selector xpath=".//book-ref"/>
    <xs:field xpath="@ISBN"/>
  </xs:keyref>
</xs:element>
```

```
<xs:element name="book">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="title" type="xs:string"/>
      <xs:element name="author" type="xs:string" minOccurs="0" maxOccurs="unbounded"/>
      <xs:element name="publisher" type="xs:string" minOccurs="0" maxOccurs="1"/>
      <xs:element name="year" type="xs:integer" minOccurs="0" maxOccurs="1"/>
      <xs:element ref="section" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
    <xs:attribute name="ISBN" type="xs:string" use="required"/>
    <xs:attribute name="price" type="xs:decimal" use="optional"/>
  </xs:complexType>
</xs:element>
```

```
<xs:element name="section">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="title" type="xs:string"/>
      <xs:element ref="content" minOccurs="0" maxOccurs="1"/>
      <xs:element ref="section" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

```
<xs:element name="content">
  <xs:complexType mixed="true">
    <xs:choice minOccurs="0" maxOccurs="unbounded">
      <xs:element name="i" type="xs:string"/>
      <xs:element name="b" type="xs:string"/>
      <xs:element name="book-ref">
        <xs:complexType><xs:attribute name="ISBN" type="xs:string"/></xs:complexType>
      </xs:element>
    </xs:choice>
  </xs:complexType>
</xs:element>

</xs:schema>
```

Why use DTD or XML Schema?

- Benefits of not using them
  - Unstructured data is easy to represent
  - Overhead of validation is avoided
- Benefits of using them
  - Serve as schema for the XML data
    - Guards against errors
    - Helps with processing
  - Facilitate information exchange
    - People can agree to use a common DTD or XML Schema to exchange data (e.g., XHTML)

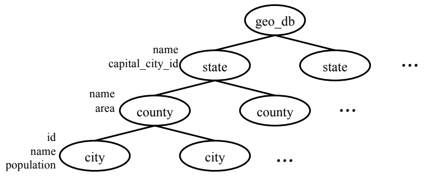
XML versus relational data

Relational data	XML data
• Schema is always fixed in advance and difficult to change	• Well-formed XML does not require predefined, fixed schema
• Simple, flat table structures	• Nested structure; ID/IDREF(S) permit arbitrary graphs
• Ordering of rows and columns is unimportant	• Ordering forced by document format; may or may not be important
• Exchange is problematic	• Designed for easy exchange
• "Native" support in all serious commercial DBMS	• Often implemented as an "add-on" on top of relations

Case study

- Design an XML document representing cities, counties, and states
  - For states, record name and capital (city)
  - For counties, record name, area, and location (state)
  - For cities, record name, population, and location (county and state)
- Assume the following:
  - Names of states are unique
  - Names of counties are only unique within a state
  - Names of cities are only unique within a county
  - A city is always located in a single county
  - A county is always located in a single state

A possible design



geo_db
name
capital_city_id
state
state
…
name
area
county
county
…
id
name
population
city
city
…