XML Data

Semi structured Data

Dr. Shaheen Khatoon

Outline

- Structured, Semistructured, and Unstructured Data
- XML Hierarchical (Tree) Data Model
- Extracting XML Documents from Relational Databases
- XML Documents, DTD, and XML Schema
- XML Languages

Structured, Semistructured, and Unstructured Data

- **Structured data**
  - Represented in a strict format (schema)
  - Example: information stored in databases
- **Semi structured data**
  - Has a certain structure
  - Not all information collected will have identical structure
- **Unstructured data**
  - Limited indication of the of data document that contains information embedded within it

Examples

- **Structured:** Excel spreadsheets Comma-separated value file (.csv) Relational database tables
- **Semi-structured:** Hypertext Markup Language (HTML) files JavaScript Object Notation (JSON) files Extensible Markup Language (XML) files
- **Unstructured:** Audio, Video, Flat Text

Semistructured Data

- Schema information mixed in with data values
- **Self-describing data**
- May be displayed as a directed graph
  - **Labels** or **tags** on directed edges represent:
    - Schema names
    - Names of attributes
    - Object types (or entity types or classes)
    - Relationships

Semistructured Data (cont’d.)

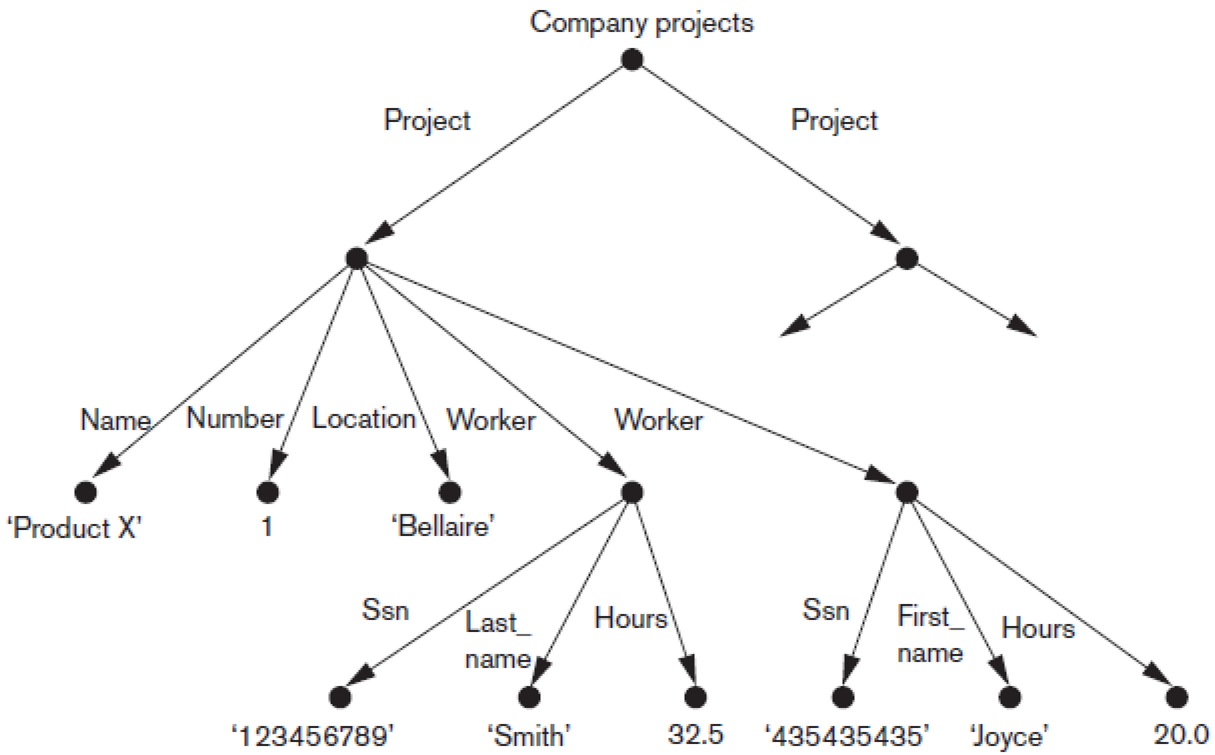


Figure 12.1
Representing semistructured data as a graph.

XML: Extensible Markup Language

- **Data sources**
  - Database storing data for Internet applications
  - Standard for data representation and exchange
- **Hypertext documents (HTML)**
  - Common method of specifying contents and formatting of Web pages
  - Tags describe content instead of formatting
- **XML data model**

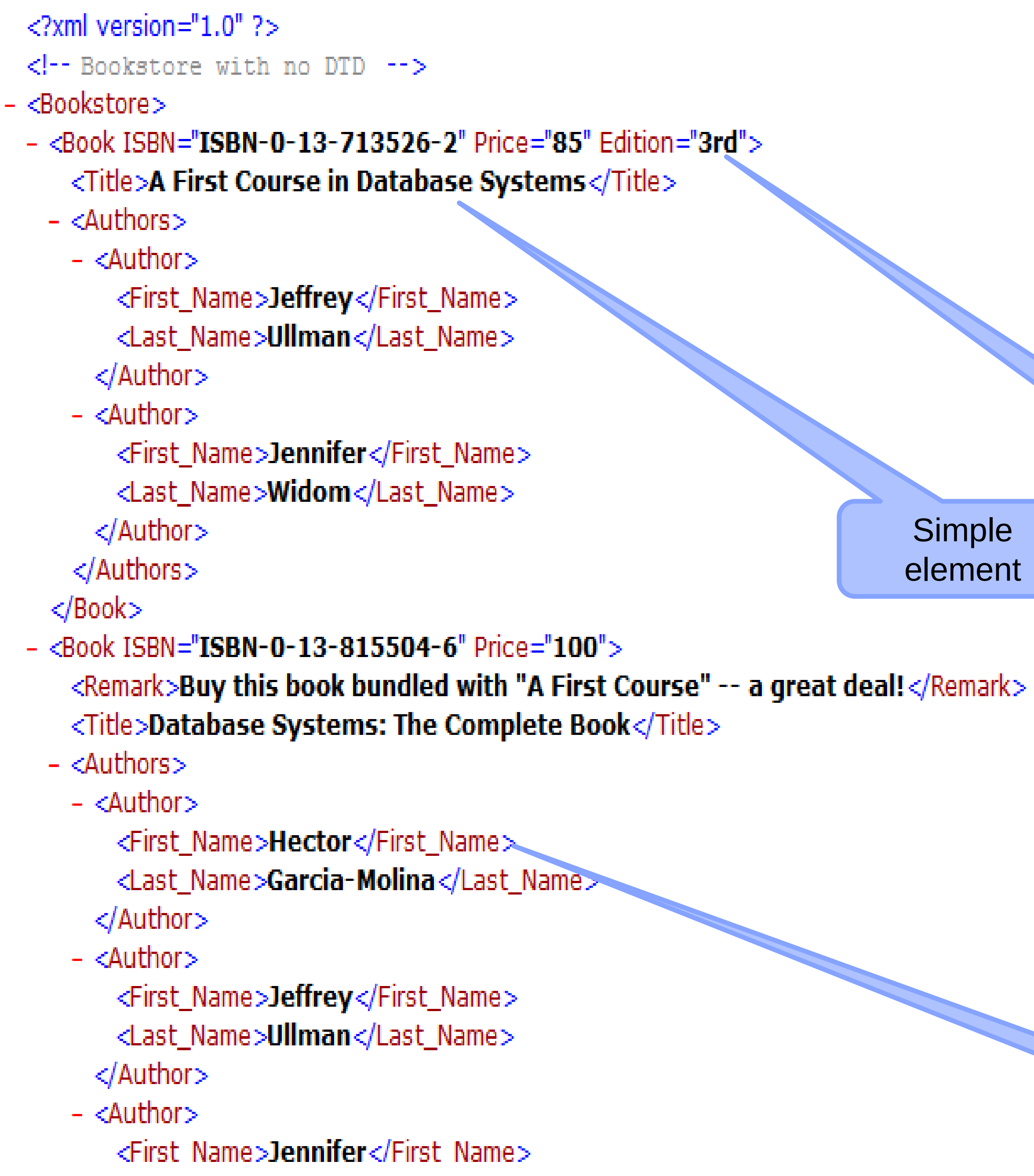
XML Hierarchical (Tree) Data Model

- **Elements** and **attributes**
  - Main structuring concepts used to construct an XML document
- **Complex elements**
  - Constructed from other elements hierarchically
- **Simple elements**
  - Contain data values
- **XML tag names**
  - Describe the meaning of the data elements in the document
  - Start tag: angled brackets: `<...>`, **`end tag`** with a slash: </ . . . >

Well-Formed XML

Basic constructs

- Tagged elements (nested)
- Attributes
- Text



```
  <?xml version="1.0" ?>
  <!-- Bookstore with no DTD  -->
- <Bookstore>
  - <Book ISBN="ISBN-0-13-713526-2" Price="85" Edition="3rd">
      <Title>A First Course in Database Systems</Title>
    - <Authors>
      - <Author>
          <First_Name>Jeffrey</First_Name>
          <Last_Name>Ullman</Last_Name>
        </Author>
      - <Author>
          <First_Name>Jennifer</First_Name>
          <Last_Name>Widom</Last_Name>
        </Author>
      </Authors>
    </Book>
  - <Book ISBN="ISBN-0-13-815504-6" Price="100">
      <Remark>Buy this book bundled with "A First Course" -- a great deal!</Remark>
      <Title>Database Systems: The Complete Book</Title>
    - <Authors>
      - <Author>
          <First_Name>Hector</First_Name>
          <Last_Name>Garcia-Molina</Last_Name>
        </Author>
      - <Author>
          <First_Name>Jeffrey</First_Name>
          <Last_Name>Ullman</Last_Name>
        </Author>
      - <Author>
          <First Name>Jennifer</First Name>
```

Simple element

Attributes

Complex element

Relational to XML Mapping

Company ER Model

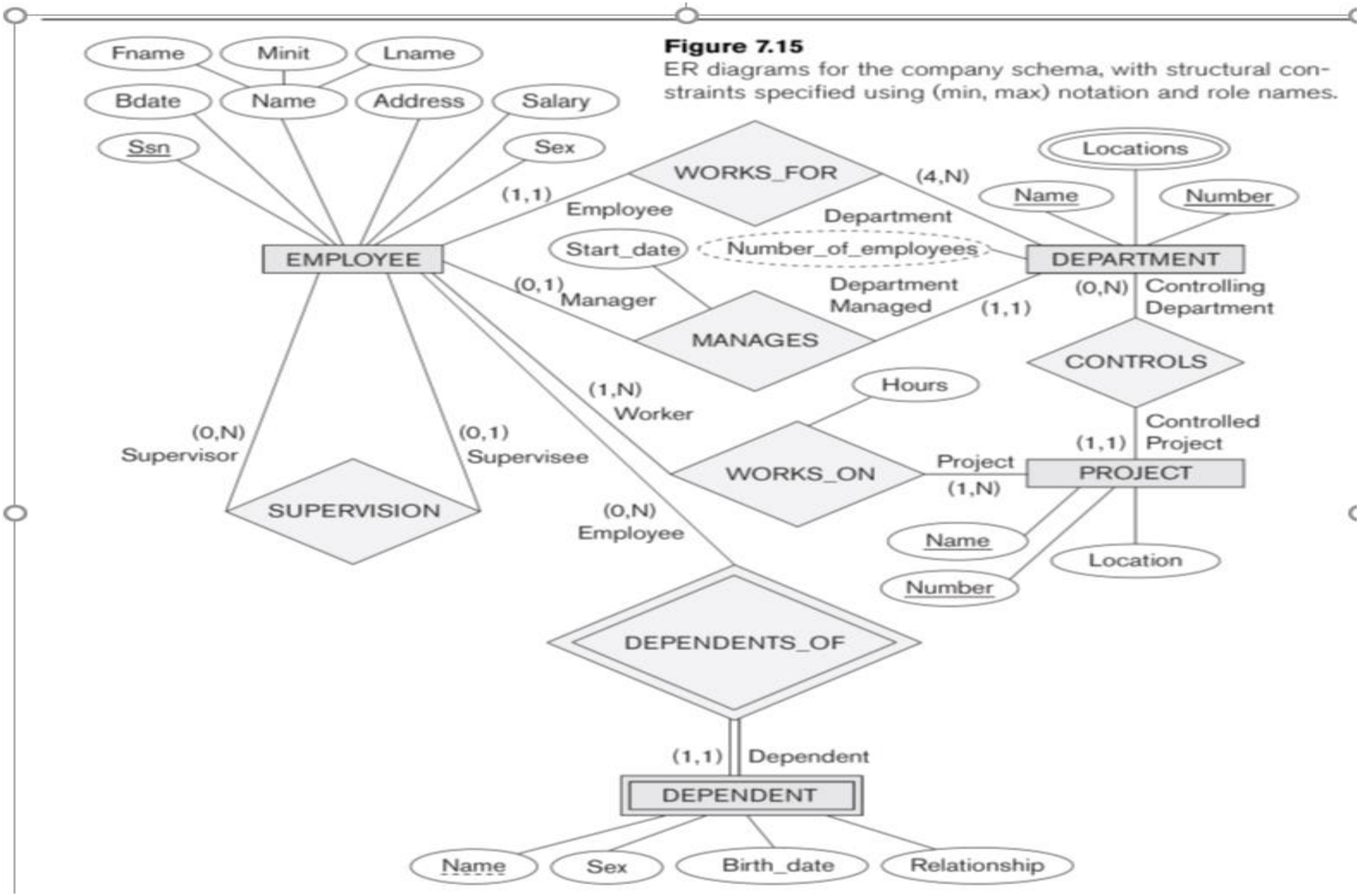
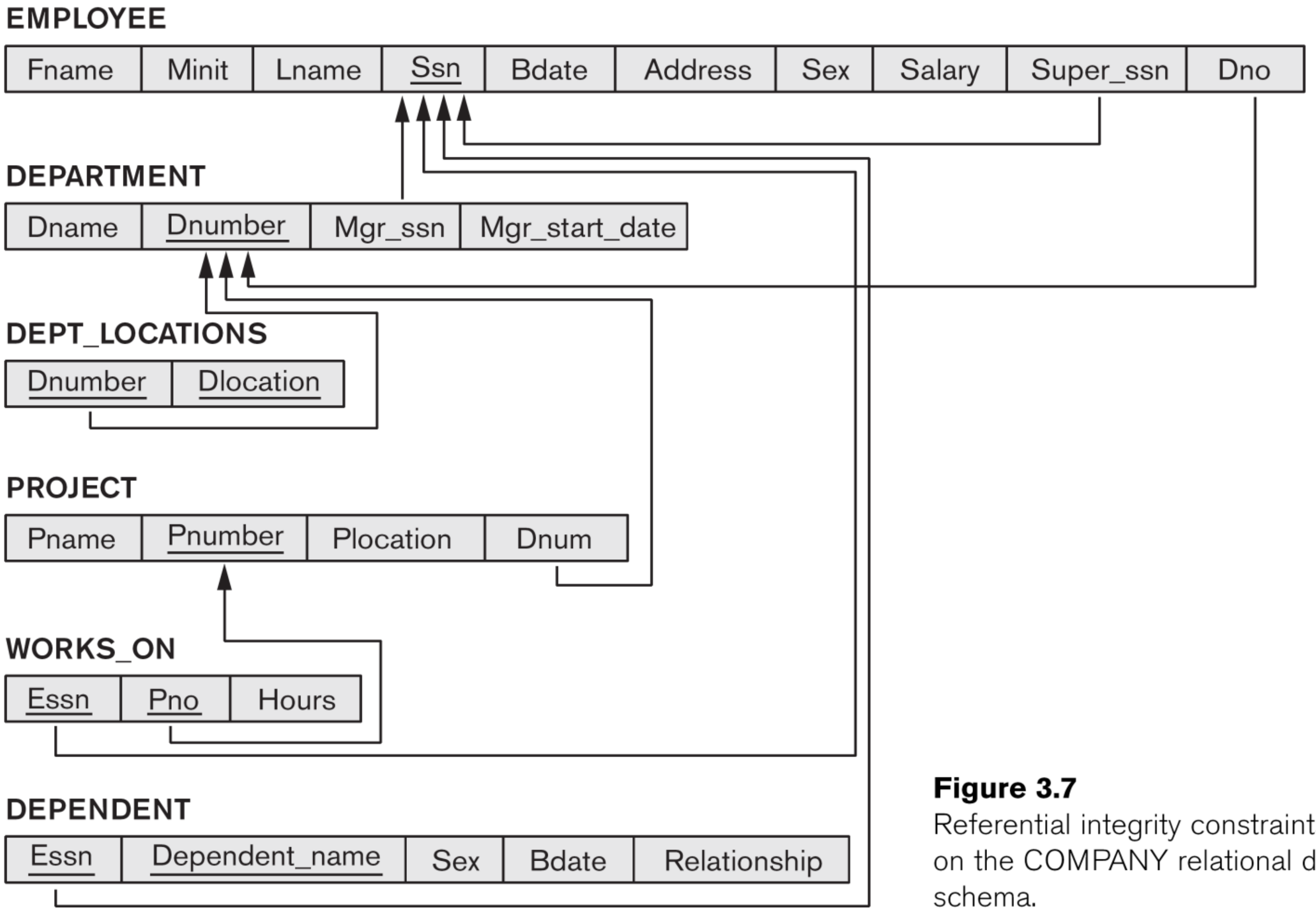


Figure 7.15
ER diagrams for the company schema, with structural constraints specified using (min, max) notation and role names.

Company Relational Model



EMPLOYEE

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	Dno

DEPARTMENT

Dname	Dnumber	Mgr_ssn	Mgr_start_date

DEPT_LOCATIONS

Dnumber	Dlocation

PROJECT

Pname	Pnumber	Plocation	Dnum

WORKS_ON

Essn	Pno	Hours

DEPENDENT

Essn	Dependent_name	Sex	Bdate	Relationship

Figure 3.7
Referential integrity constraints displayed on the COMPANY relational database schema.

Company Entities

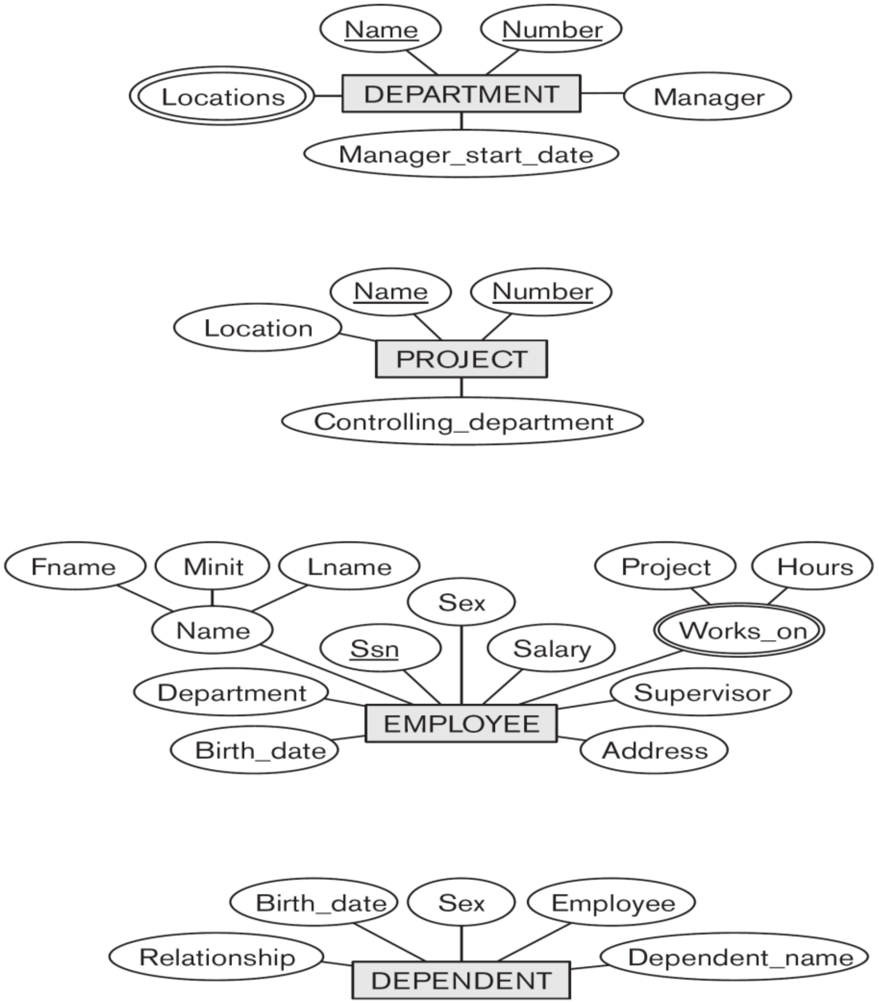


Figure 7.8
Preliminary design of entity types for the COMPANY database. Some of the shown attributes will be refined into relationships.

Relational to XML Mapping

```xml
<?xml version="1.0" standalone="yes" ?>
<Departments>
    <Department>
        <Dname>Research</Dname>
        <Dnumber>5</Dnumber>
        <Mgr_ssn>333445555</Mgr_ssn>
        <Mgr_start_date>1988-05-22</Mgr_start_date>
        <Dlocation>Bellaire</Dlocation>
        <Dlocation>Sugarland</Dlocation>
        <Dlocation>Houston</Dlocation>
    </Department>

    <Department>
        <Dname>Administration</Dname>
        <Dnumber>4</Dnumber>
        <Mgr_ssn>987654321</Mgr_ssn>
        <Mgr_start_date>1995-01-01</Mgr_start_date>
        <Dlocation>Stafford</Dlocation>
    </Department>
    ...
</Departments>
```

Please complete mapping from the homework exercise

Relational Model versus XML

	Relational	XML
Structure	Tables	Hierarchical Tree
Schema	Fixed in advance	Flexible “Self describing”
Queries	Simple (SQL)	Complex: Xpath, XQuery
Ordering	None – use order by clause	Implied ordering
Implementation	Native models of relational systems	Add-on

Knowledge Check

- You're creating a database to contain information about university records: students, courses, grades, etc. Should you use the relational model or XML?

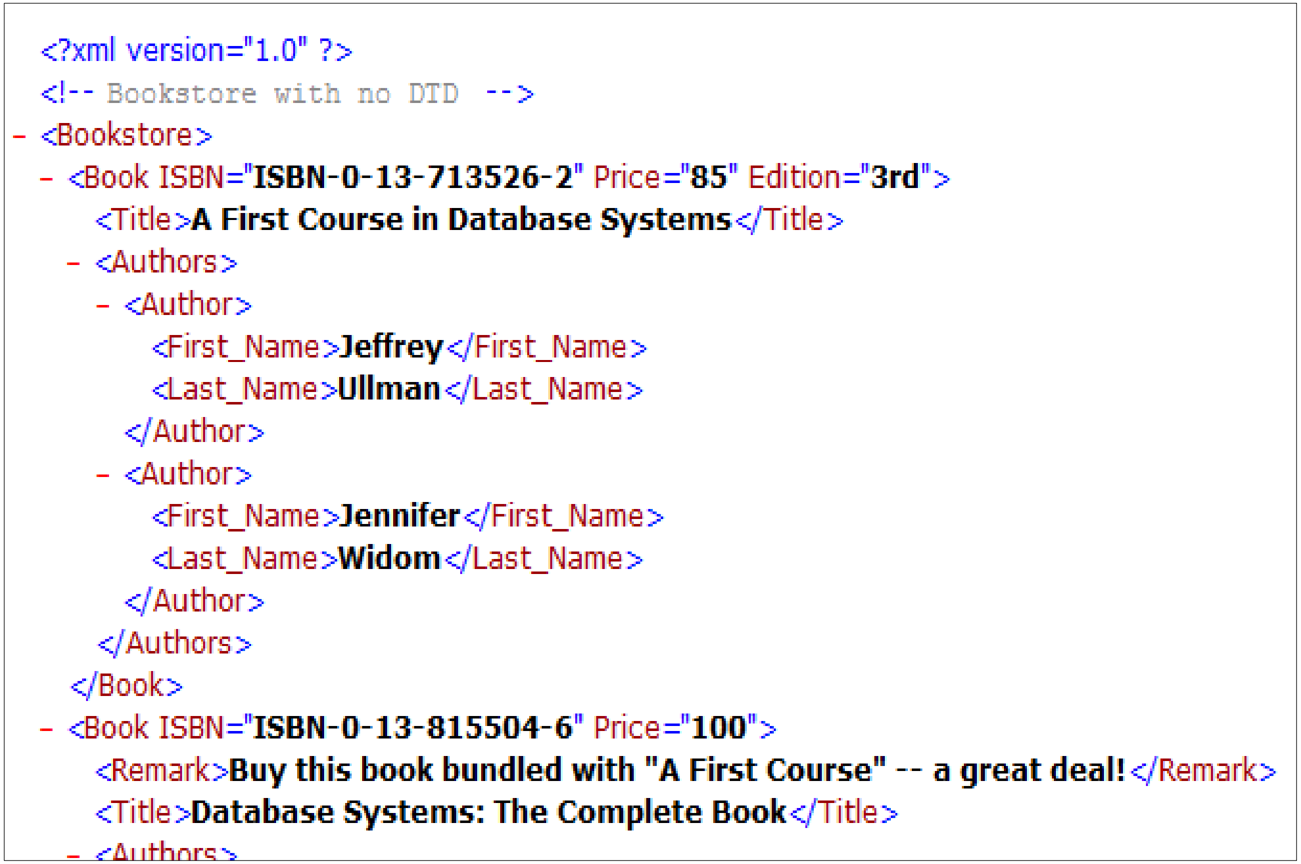
- You're creating a database to contain information for a university web site: news, academic announcements, admissions, events, research, etc. Should you use the relational model or XML?

- You're creating a database to contain information about family trees (ancestry). Should you use the relational model or XML?

“Well-Formed” XML

Adheres to basic structural requirements

- Single root element
- Matched tags, proper nesting
- Unique attributes within elements

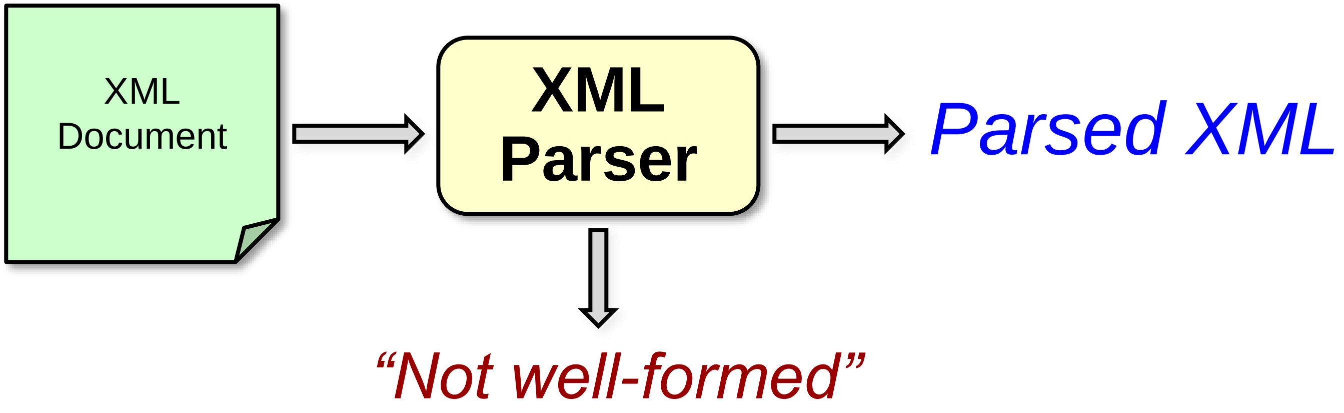


```
  <?xml version="1.0" ?>
  <!-- Bookstore with no DTD  -->
- <Bookstore>
  - <Book ISBN="ISBN-0-13-713526-2" Price="85" Edition="3rd">
      <Title>A First Course in Database Systems</Title>
    - <Authors>
      - <Author>
          <First_Name>Jeffrey</First_Name>
          <Last_Name>Ullman</Last_Name>
        </Author>
      - <Author>
          <First_Name>Jennifer</First_Name>
          <Last_Name>Widom</Last_Name>
        </Author>
      </Authors>
    </Book>
  - <Book ISBN="ISBN-0-13-815504-6" Price="100">
      <Remark>Buy this book bundled with "A First Course" -- a great deal!</Remark>
      <Title>Database Systems: The Complete Book</Title>
    - <Authors>
```

“Well-Formed” XML

Adheres to basic structural requirements

- Single root element
- Matched tags, proper nesting
- Unique attributes within elements

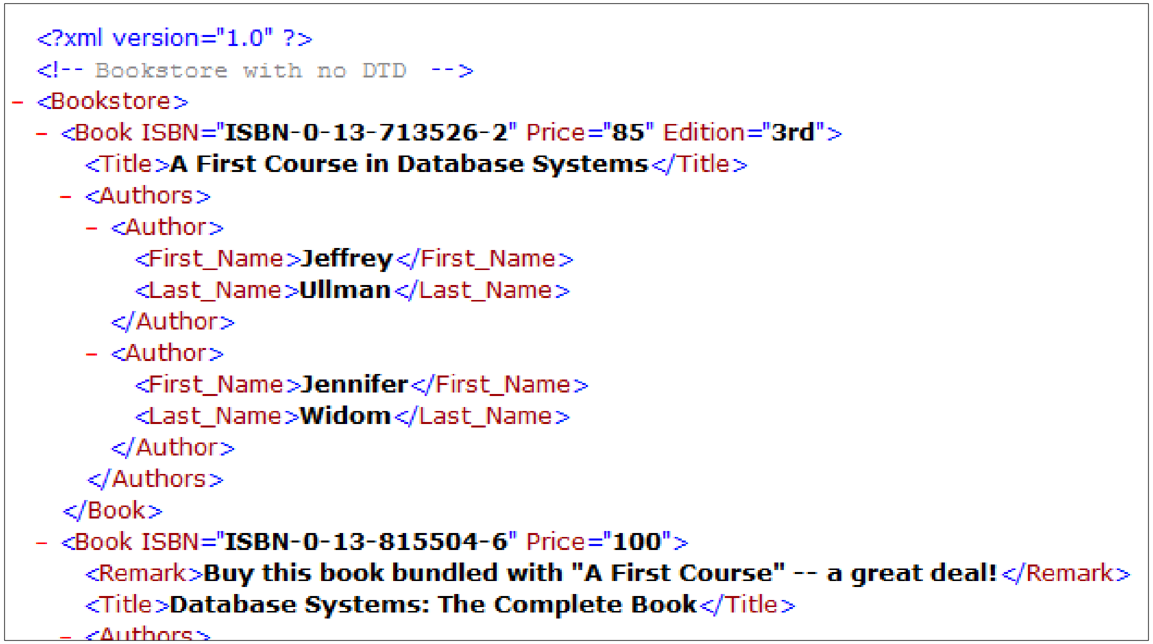


Well-Formed XML

Displaying XML

Use rule-based language to translate to HTML

- *Cascading stylesheets* (CSS)
- *Extensible stylesheet language* (XSL)

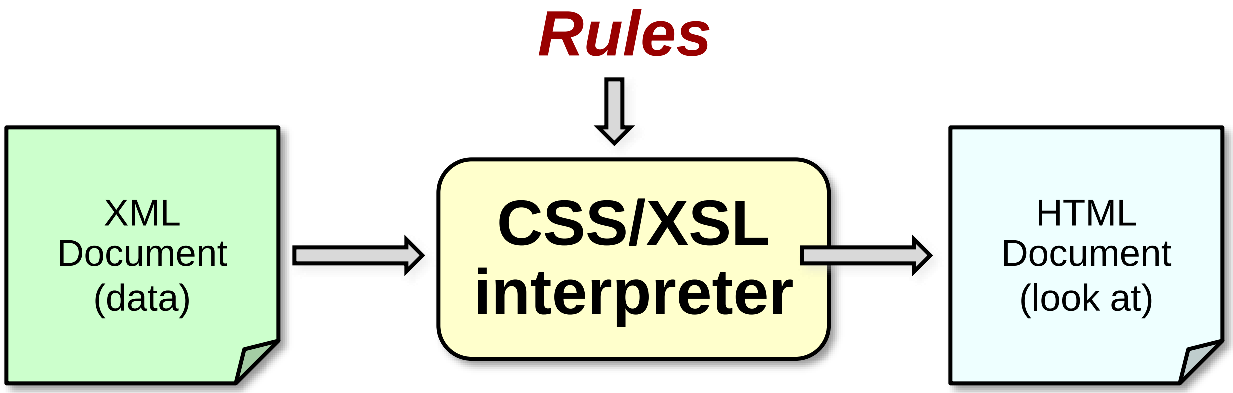


```
  <?xml version="1.0" ?>
  <!-- Bookstore with no DTD  -->
- <Bookstore>
  - <Book ISBN="ISBN-0-13-713526-2" Price="85" Edition="3rd">
      <Title>A First Course in Database Systems</Title>
    - <Authors>
      - <Author>
          <First_Name>Jeffrey</First_Name>
          <Last_Name>Ullman</Last_Name>
        </Author>
      - <Author>
          <First_Name>Jennifer</First_Name>
          <Last_Name>Widom</Last_Name>
        </Author>
      </Authors>
    </Book>
  - <Book ISBN="ISBN-0-13-815504-6" Price="100">
      <Remark>Buy this book bundled with "A First Course" -- a great deal!</Remark>
      <Title>Database Systems: The Complete Book</Title>
    - <Authors>
```

Displaying XML

Use rule-based language to translate to HTML

- *Cascading stylesheets* (CSS)
- *Extensible stylesheet language* (XSL)

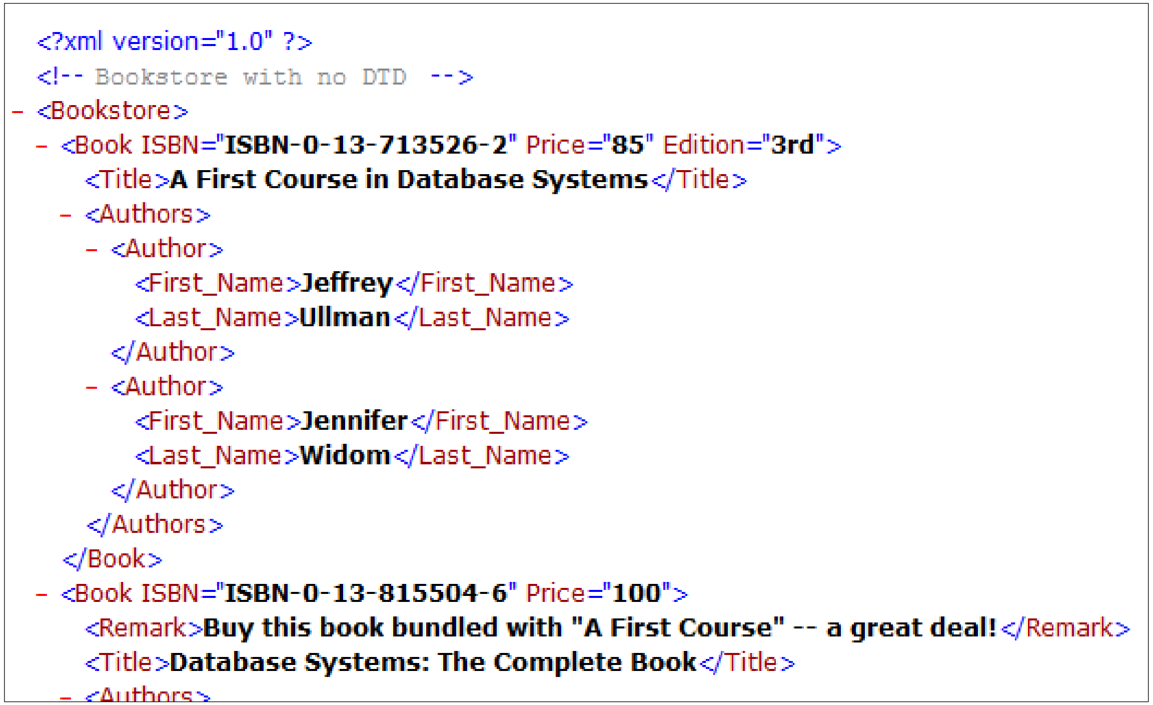


Well-Formed XML

Extensible Markup Language (XML)

- Standard for data representation and exchange

➢ Formal specification is enormous; we cover most important components



```
  <?xml version="1.0" ?>
  <!-- Bookstore with no DTD  -->
- <Bookstore>
  - <Book ISBN="ISBN-0-13-713526-2" Price="85" Edition="3rd">
      <Title>A First Course in Database Systems</Title>
    - <Authors>
      - <Author>
          <First_Name>Jeffrey</First_Name>
          <Last_Name>Ullman</Last_Name>
        </Author>
      - <Author>
          <First_Name>Jennifer</First_Name>
          <Last_Name>Widom</Last_Name>
        </Author>
      </Authors>
    </Book>
  - <Book ISBN="ISBN-0-13-815504-6" Price="100">
      <Remark>Buy this book bundled with "A First Course" -- a great deal!</Remark>
      <Title>Database Systems: The Complete Book</Title>
    - <Authors>
```

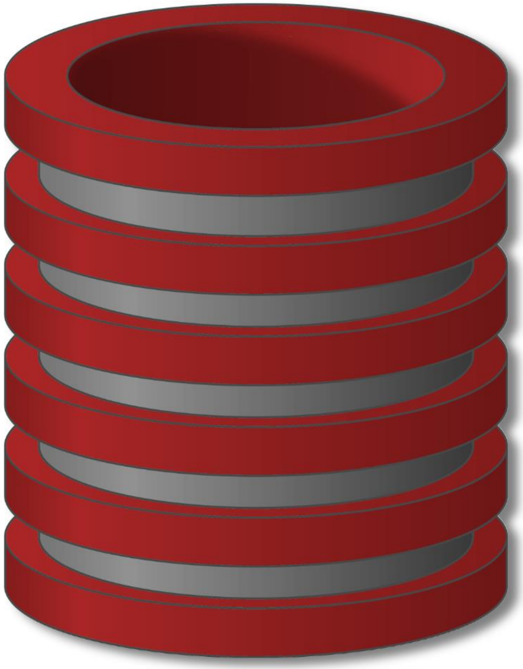
Practice: Identify Well Formed XML

```
<tasklist>
        <task name=eat/>
        <task name=drink/>
        <task name=play/>
</tasklist>
```

```
<tasklist>
        <task name="eat">
        <task name="drink">
        <task name="play">
</tasklist>
```

```
<tasklist>
        <task name="eat"/>
        <task name="drink"/>
        <task name="play"/>
</tasklist>
```

```
<tasklist>
        <task name="eat"/>
        <task name="drink"/>
        <task name="play"/>
<tasklist>
```



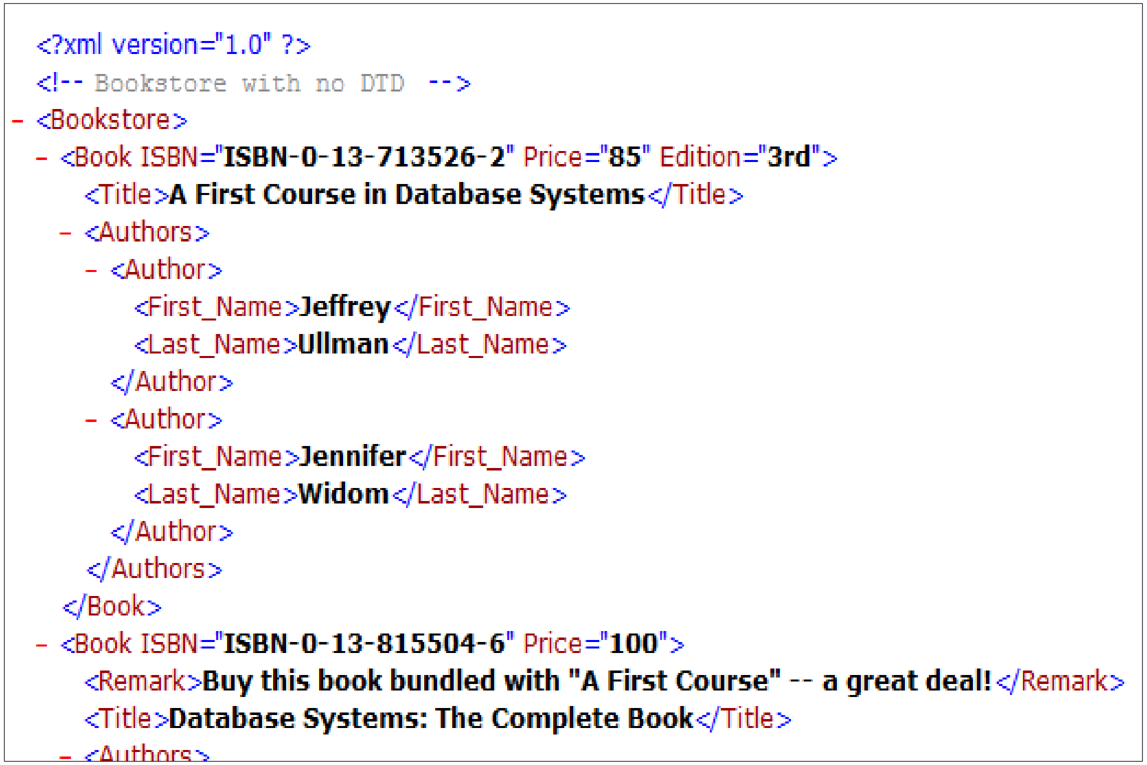
XML Data

DTDs

“Well-Formed” XML

Adheres to basic structural requirements

- Single root element
- Matched tags, proper nesting
- Unique attributes within elements

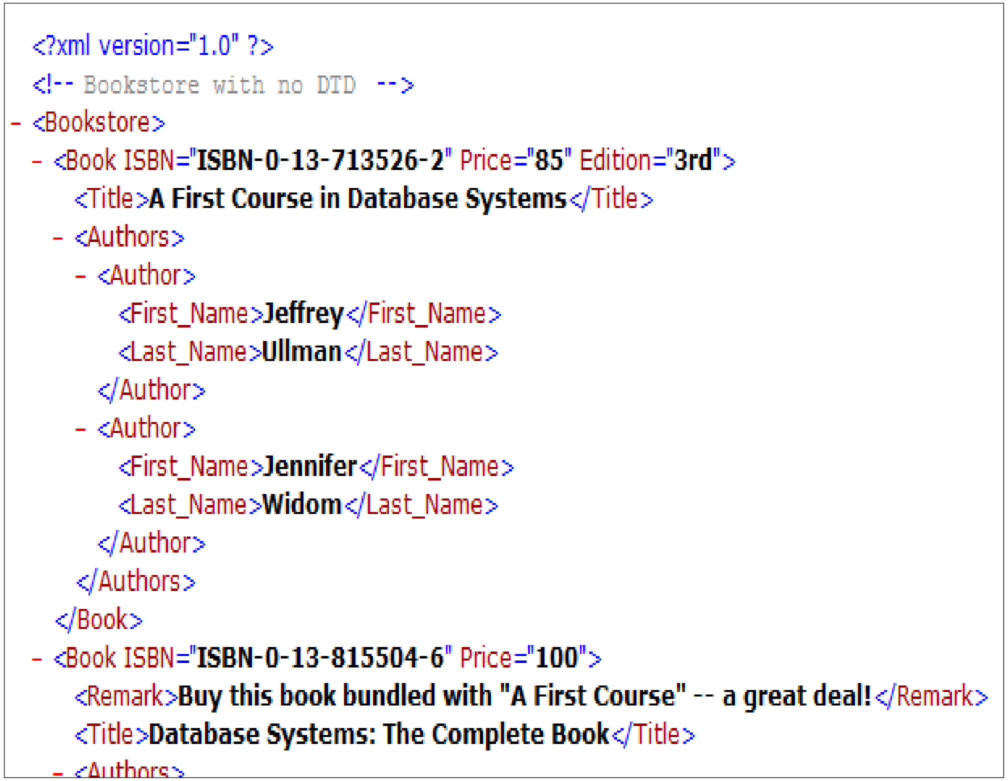


```
  <?xml version="1.0" ?>
  <!-- Bookstore with no DTD  -->
- <Bookstore>
  - <Book ISBN="ISBN-0-13-713526-2" Price="85" Edition="3rd">
      <Title>A First Course in Database Systems</Title>
    - <Authors>
      - <Author>
          <First_Name>Jeffrey</First_Name>
          <Last_Name>Ullman</Last_Name>
        </Author>
      - <Author>
          <First_Name>Jennifer</First_Name>
          <Last_Name>Widom</Last_Name>
        </Author>
      </Authors>
    </Book>
  - <Book ISBN="ISBN-0-13-815504-6" Price="100">
      <Remark>Buy this book bundled with "A First Course" -- a great deal!</Remark>
      <Title>Database Systems: The Complete Book</Title>
    - <Authors>
```

“Valid” XML

Adheres to basic structural requirements

- Also adheres to content-specific specification
  - *Document Type Descriptor* (DTD)
  - *XML Schema Description* (XSD)

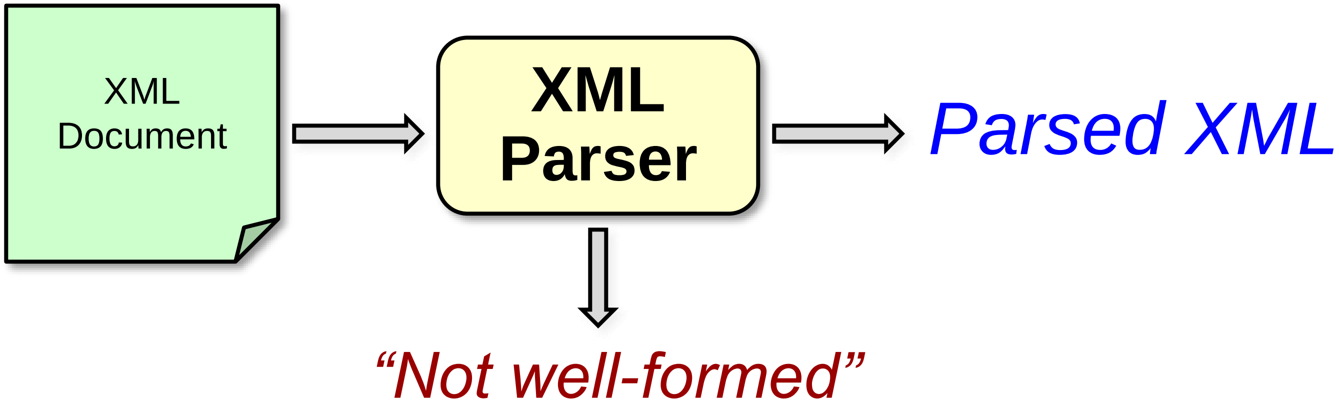


```
  <?xml version="1.0" ?>
  <!-- Bookstore with no DTD  -->
- <Bookstore>
  - <Book ISBN="ISBN-0-13-713526-2" Price="85" Edition="3rd">
      <Title>A First Course in Database Systems</Title>
    - <Authors>
      - <Author>
          <First_Name>Jeffrey</First_Name>
          <Last_Name>Ullman</Last_Name>
        </Author>
      - <Author>
          <First_Name>Jennifer</First_Name>
          <Last_Name>Widom</Last_Name>
        </Author>
      </Authors>
    </Book>
  - <Book ISBN="ISBN-0-13-815504-6" Price="100">
      <Remark>Buy this book bundled with "A First Course" -- a great deal!</Remark>
      <Title>Database Systems: The Complete Book</Title>
    - <Authors>
```

“Valid” XML

Adheres to basic structural requirements

- Also adheres to content-specific specification



XML Document

XML Parser

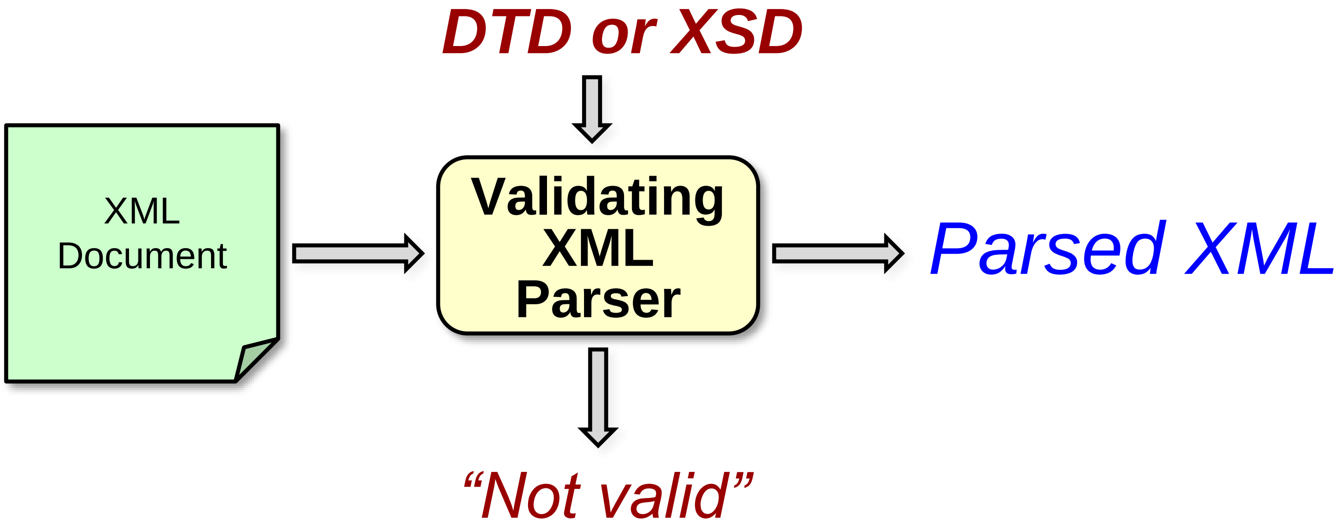
Parsed XML

“Not well-formed”

“Valid” XML

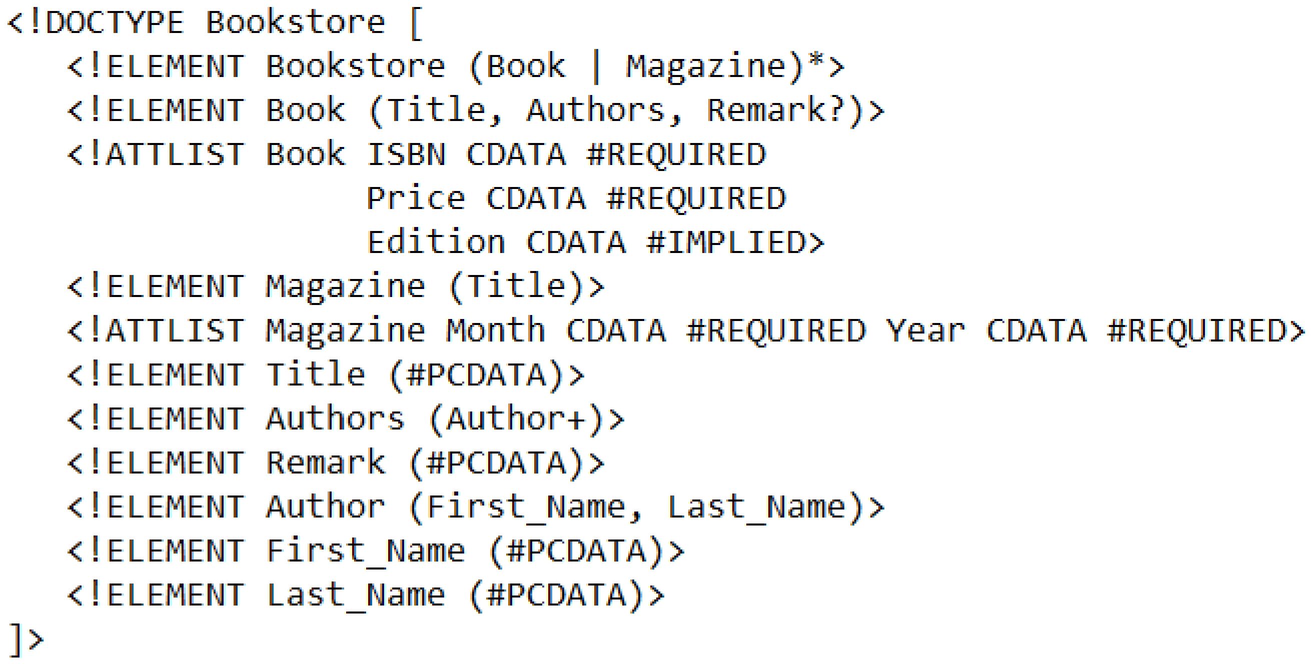
Adheres to basic structural requirements

- ➢Also adheres to content-specific specification



Document Type Descriptor (DTD)

- Grammar-like language for specifying elements, attributes, nesting, ordering, #occurrences



```
<!DOCTYPE Bookstore [
   <!ELEMENT Bookstore (Book | Magazine)*>
   <!ELEMENT Book (Title, Authors, Remark?)>
   <!ATTLIST Book ISBN CDATA #REQUIRED
                  Price CDATA #REQUIRED
                  Edition CDATA #IMPLIED>
   <!ELEMENT Magazine (Title)>
   <!ATTLIST Magazine Month CDATA #REQUIRED Year CDATA #REQUIRED>
   <!ELEMENT Title (#PCDATA)>
   <!ELEMENT Authors (Author+)>
   <!ELEMENT Remark (#PCDATA)>
   <!ELEMENT Author (First_Name, Last_Name)>
   <!ELEMENT First_Name (#PCDATA)>
   <!ELEMENT Last_Name (#PCDATA)>
]>
```

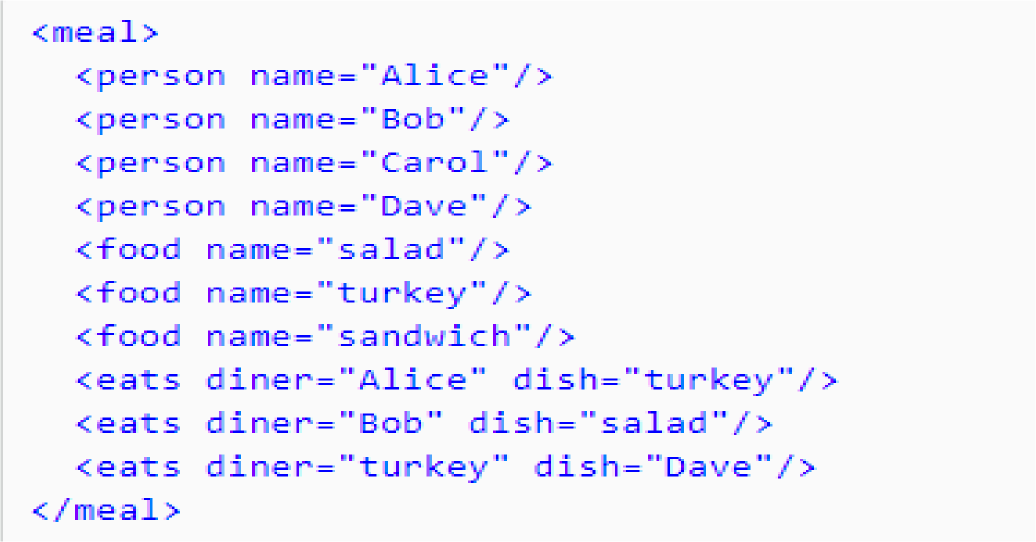
Please refer to below link for more detail

w3schools.com/xml/xml_dtd_attributes.asp

Here is an XML DTD

Create a XML documents that is valid with given DTD?

```
<!DOCTYPE meal
            [ <!ELEMENT meal (person*,food*,eats*)>       <!ELEMENT
person EMPTY>
            <!ELEMENT food EMPTY>
            <!ELEMENT eats EMPTY>
            <!ATTLIST person name ID #REQUIRED>
            <!ATTLIST food name ID #REQUIRED>
            <!ATTLIST eats diner IDREF #REQUIRED dish IDREF
#REQUIRED> ]>
```

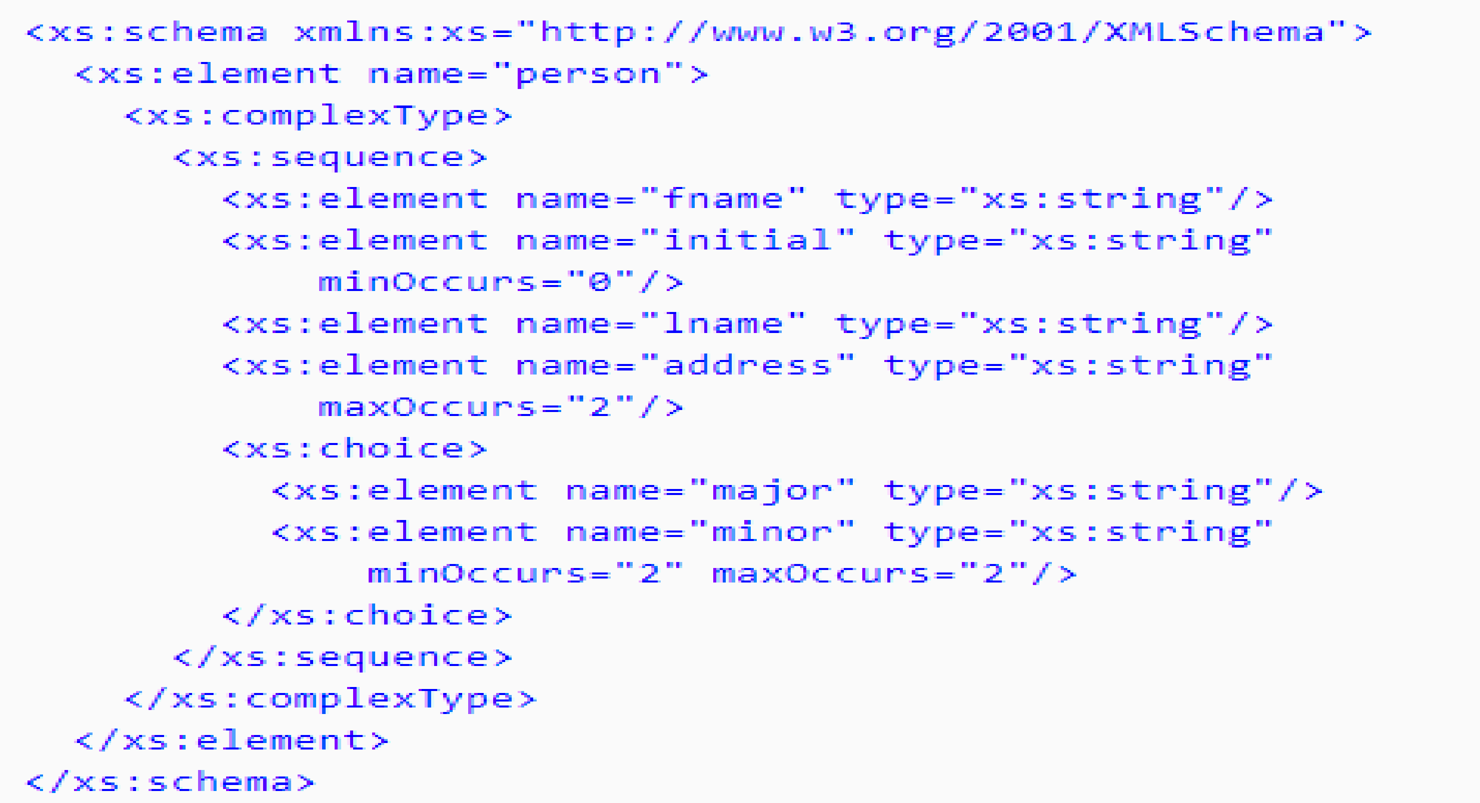


```
<meal>
  <person name="Alice"/>
  <person name="Bob"/>
  <person name="Carol"/>
  <person name="Dave"/>
  <food name="salad"/>
  <food name="turkey"/>
  <food name="sandwich"/>
  <eats diner="Alice" dish="turkey"/>
  <eats diner="Bob" dish="salad"/>
  <eats diner="turkey" dish="Dave"/>
</meal>
```

XML Schema (XSD)

- Extensive language
- Like DTDs, can specify elements, attributes, nesting, ordering, #occurrences
- Also data types, keys, (typed) pointers, and more

➢ **XSD is written in XML**



```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="person">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="fname" type="xs:string"/>
        <xs:element name="initial" type="xs:string"
            minOccurs="0"/>
        <xs:element name="lname" type="xs:string"/>
        <xs:element name="address" type="xs:string"
            maxOccurs="2"/>
        <xs:choice>
          <xs:element name="major" type="xs:string"/>
          <xs:element name="minor" type="xs:string"
              minOccurs="2" maxOccurs="2"/>
        </xs:choice>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

DTD/XSD versus none (well-formed)

+ DTD/XSD

- **Program can assume the structure**
- **CSS/XSL rules are simple when program has particular structure**
- **Specification language- DTD as a specification what the XML look like**
- **Documentation**
- **Strongly typed Data**

- DTD/XSD

- **Flexibility and ease of change is difficult**
- **DTD can be messy- irregular structure**
- **Benefits of no typing**