Overview

- XML as Data
  - Data-Centric vs. Document-Centric
  - Relational vs. XML Models

- XML for Programming
  - SAX vs. DOM
### XML: Data vs. Document

<table>
<thead>
<tr>
<th>Data-Centric</th>
<th>Document-Centric</th>
</tr>
</thead>
<tbody>
<tr>
<td>Used as a data transport or exchange</td>
<td>SGML-like capabilities</td>
</tr>
<tr>
<td>Repeating, regular structure</td>
<td>Irregular structure</td>
</tr>
<tr>
<td>Eg,</td>
<td>Semi-structure</td>
</tr>
<tr>
<td>- Sales orders</td>
<td>- Eg,</td>
</tr>
<tr>
<td>- Patient records</td>
<td>- Forms</td>
</tr>
<tr>
<td>- Scientific data</td>
<td>- User’s manuals</td>
</tr>
<tr>
<td>- Dynamic web page</td>
<td>- Marketing brochure</td>
</tr>
<tr>
<td></td>
<td>- Static Web pages</td>
</tr>
</tbody>
</table>

### Eg, Data-Centric

```xml
<CATALOG>
  <CD>
    <TITLE>Empire Burlesque</TITLE>
    <ARTIST>Bob Dylan</ARTIST>
    <COUNTRY>USA</COUNTRY>
    <COMPANY>Columbia/COMPANY>
    <PRICE>10.90</PRICE>
    <YEAR>1985</YEAR>
  </CD>
  <CD>
    <TITLE>Hide your heart</TITLE>
    <ARTIST>Bonnie Tyler</ARTIST>
    <COUNTRY>UK</COUNTRY>
    <COMPANY>CBS Records/COMPANY>
    <PRICE>9.90</PRICE>
    <YEAR>1988</YEAR>
  </CD>
  <CD>
    <TITLE>Greatest Hits</TITLE>
    <ARTIST>Dolly Parton</ARTIST>
    <COUNTRY>USA</COUNTRY>
    <COMPANY>RC</COMPANY>
    <PRICE>9.90</PRICE>
    <YEAR>1982</YEAR>
  </CD>
</CATALOG>
```
Eg, Document-Centric

**Advisor Evaluation Form**

PhD Student Annual Evaluation – Advisor’s comments [CONFIDENTIAL]

Instructions: Complete AFTER meeting with the student to discuss their form.

<table>
<thead>
<tr>
<th>Student (print name)</th>
<th>Adviser (print name / list co-advisor if applicable)</th>
</tr>
</thead>
</table>

1. In what ways have you and this student collaborated this past year? What other collaborations has the student been involved in?

---

**Database Perspective**

- DB supports
  - Capture
  - Storage
  - Retrieval
  - Exchange

- XML originally as the language to exchange data over Web
  - Replacing EDI (Electronic Data Interchange)
RDBMS vs. XML Example

- AFV receives 100+ videos every week
- Build a DB app to answer the following queries:
  - Who sent which videos?
  - Show me all videos about Cat category
  - How many videos in a database since Jan 1, 2003?
  - Which is the video with the best rating for the 1st week of Jan?
  - Where does the sender James live? Phone? Gender?
  - How many videos does James send so far?
  - Show me all the ghost videos (ones without sender information)

ER Model

```
ER Model

Videos
  VID
  Category
  Date
  Rating

Sends

Owners
  Name
  Address
  Phone
  Gender
```


ER → RDBMS

**ER Diagram:**
- **VID**
- **Videos**
  - Category
  - Date
  - Rating
- **Sends**
- **Owners**
  - Phone
  - Gender
  - Address

**RDBMS:**

**Videos Table:**
<table>
<thead>
<tr>
<th>VID</th>
<th>Category</th>
<th>Date</th>
<th>Rating</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>Comedy</td>
<td>2005/1/1</td>
<td>5</td>
</tr>
<tr>
<td>200</td>
<td>Action</td>
<td>2005/1/10</td>
<td>4</td>
</tr>
<tr>
<td>300</td>
<td>SF</td>
<td>2004/12/31</td>
<td>5</td>
</tr>
</tbody>
</table>

**Sends Table:**
<table>
<thead>
<tr>
<th>Phone</th>
<th>VID</th>
</tr>
</thead>
<tbody>
<tr>
<td>564-3456</td>
<td>100</td>
</tr>
<tr>
<td>123-4567</td>
<td>200</td>
</tr>
<tr>
<td>123-4567</td>
<td>300</td>
</tr>
</tbody>
</table>

**Owners Table:**
<table>
<thead>
<tr>
<th>Name</th>
<th>Phone</th>
<th>Address</th>
<th>Gender</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jenny</td>
<td>564-3456</td>
<td>310 N. Atherton</td>
<td>F</td>
</tr>
<tr>
<td>Tom</td>
<td>123-4567</td>
<td>10 S. Beaver</td>
<td>M</td>
</tr>
</tbody>
</table>
Change #1

- VHS video → VHS, CD, DVD
- 100+ videos → 1 million videos

RDBMS

<table>
<thead>
<tr>
<th>VID</th>
<th>Category</th>
<th>Format</th>
<th>Date</th>
<th>Rating</th>
<th>Video</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>Comedy</td>
<td>VHS</td>
<td>2005/1/1</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1000000</td>
<td>SF</td>
<td>DVD</td>
<td>2004/12/31</td>
<td>5</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Phone</th>
<th>VID</th>
</tr>
</thead>
<tbody>
<tr>
<td>564-3456</td>
<td>100</td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
<tr>
<td>123-4567</td>
<td>1000000</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Name</th>
<th>Phone</th>
<th>Address</th>
<th>Gender</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jenny</td>
<td>564-3456</td>
<td>310 N. Atherton</td>
<td>F</td>
</tr>
<tr>
<td>Tom</td>
<td>123-4567</td>
<td>10 S. Beaver</td>
<td>M</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Change #2

- Arbitrary name formats for owners
  - Eg, J. Doe vs. Dr. “Jonny” John Jay Doe Jr
- 100+ different ways to capture owners’ information
  - “100 E. Foster Ave #200, State College, PA, 16801” vs.
  - adr1=“100 E. Foster #210”, adr2=“State College, PA, 16801”
  - street=“100 E. Foster, #200”, city=“State College”, state=“PA”, zip=“16801”
- 100+ different video formats with varying properties → 1000+ attributes for Videos

RDBMS: Finest Granularity

<table>
<thead>
<tr>
<th>Video</th>
<th>VID</th>
<th>Category</th>
<th>Date</th>
<th>Rating</th>
<th>Attr1</th>
<th>Attr2</th>
<th>...</th>
<th>att1000</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>100</td>
<td>Comedy</td>
<td>2005/1/1</td>
<td>5</td>
<td>10</td>
<td></td>
<td>...</td>
<td>T1</td>
</tr>
<tr>
<td></td>
<td>200</td>
<td>Action</td>
<td>2005/1/10</td>
<td>4</td>
<td>20</td>
<td></td>
<td>...</td>
<td></td>
</tr>
<tr>
<td></td>
<td>300</td>
<td>SF</td>
<td>2004/12/31</td>
<td>5</td>
<td></td>
<td></td>
<td>...</td>
<td>S20</td>
</tr>
</tbody>
</table>

Sends

<table>
<thead>
<tr>
<th>Phone</th>
<th>VID</th>
</tr>
</thead>
<tbody>
<tr>
<td>564-3456</td>
<td>100</td>
</tr>
<tr>
<td>123-4567</td>
<td>200</td>
</tr>
<tr>
<td>123-4567</td>
<td>300</td>
</tr>
</tbody>
</table>

Owners

<table>
<thead>
<tr>
<th>Prefix</th>
<th>NN</th>
<th>FN</th>
<th>MN</th>
<th>LN</th>
<th>Suffix</th>
<th>Phone</th>
<th>Street</th>
<th>City</th>
<th>State</th>
<th>Zip</th>
<th>Gender</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dr.</td>
<td>Jonny</td>
<td>John</td>
<td>Jay</td>
<td>Doe</td>
<td>Jr</td>
<td>123-4567</td>
<td>10 S. Beaver</td>
<td>PA</td>
<td>16801</td>
<td>M</td>
<td>M</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>RDDMS: Finest Granularity</th>
<th>Video</th>
<th>VID</th>
<th>Category</th>
<th>Date</th>
<th>Rating</th>
<th>Attr1</th>
<th>Attr2</th>
<th>...</th>
<th>att1000</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>100</td>
<td>Comedy</td>
<td>2005/1/1</td>
<td>5</td>
<td>10</td>
<td></td>
<td>...</td>
<td>T1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>200</td>
<td>Action</td>
<td>2005/1/10</td>
<td>4</td>
<td>20</td>
<td></td>
<td>...</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>300</td>
<td>SF</td>
<td>2004/12/31</td>
<td>5</td>
<td></td>
<td></td>
<td>...</td>
<td>S20</td>
</tr>
</tbody>
</table>

Sends

<table>
<thead>
<tr>
<th>Phone</th>
<th>VID</th>
</tr>
</thead>
<tbody>
<tr>
<td>564-3456</td>
<td>100</td>
</tr>
<tr>
<td>123-4567</td>
<td>200</td>
</tr>
<tr>
<td>123-4567</td>
<td>300</td>
</tr>
</tbody>
</table>

Owners

<table>
<thead>
<tr>
<th>Prefix</th>
<th>NN</th>
<th>FN</th>
<th>MN</th>
<th>LN</th>
<th>Suffix</th>
<th>Phone</th>
<th>Street</th>
<th>City</th>
<th>State</th>
<th>Zip</th>
<th>Gender</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dr.</td>
<td>Jonny</td>
<td>John</td>
<td>Jay</td>
<td>Doe</td>
<td>Jr</td>
<td>123-4567</td>
<td>10 S. Beaver</td>
<td>PA</td>
<td>16801</td>
<td>M</td>
<td>M</td>
</tr>
</tbody>
</table>
### RDBMS: Coarsest Granularity

<table>
<thead>
<tr>
<th>Video</th>
<th>VID</th>
<th>Category</th>
<th>Date</th>
<th>Rating</th>
<th>Att1to1000</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>100</td>
<td>Comedy</td>
<td>2005/1/1</td>
<td>5</td>
<td>10, T1</td>
</tr>
<tr>
<td></td>
<td>200</td>
<td>Action</td>
<td>2005/1/10</td>
<td>4</td>
<td>20</td>
</tr>
<tr>
<td></td>
<td>300</td>
<td>SF</td>
<td>2004/12/31</td>
<td>5</td>
<td>S20</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Sends</th>
<th>Phone</th>
<th>VID</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>564-3456</td>
<td>100</td>
</tr>
<tr>
<td></td>
<td>123-4567</td>
<td>200</td>
</tr>
<tr>
<td></td>
<td>123-4567</td>
<td>300</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Owners</th>
<th>Name</th>
<th>Phone</th>
<th>Address</th>
<th>Gender</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Jenny</td>
<td>564-3456</td>
<td>310 N. Atherton, State College, PA, 16801</td>
<td>F</td>
</tr>
<tr>
<td></td>
<td>Dr. &quot;Jonny&quot; John Jay Doe Jr</td>
<td>123-4567</td>
<td>10 S. Beaver, PA</td>
<td>M</td>
</tr>
</tbody>
</table>

### RDBMS: Ideal Case

<table>
<thead>
<tr>
<th>Video</th>
<th>VID</th>
<th>Category</th>
<th>Date</th>
<th>Rating</th>
<th>Att1</th>
<th>Att2</th>
<th>...</th>
<th>att1000</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>100</td>
<td>Comedy</td>
<td>2005/1/1</td>
<td>5</td>
<td>10</td>
<td>...</td>
<td>T1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>200</td>
<td>Action</td>
<td>2005/1/10</td>
<td>4</td>
<td>20</td>
<td>...</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>300</td>
<td>SF</td>
<td>2004/12/31</td>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td>S20</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Sends</th>
<th>Phone</th>
<th>VID</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>564-3456</td>
<td>100</td>
</tr>
<tr>
<td></td>
<td>123-4567</td>
<td>200</td>
</tr>
<tr>
<td></td>
<td>123-4567</td>
<td>300</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Owners</th>
<th>Name</th>
<th>Phone</th>
<th>Address</th>
<th>Gender</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Jenny</td>
<td>564-3456</td>
<td>310 N. Atherton, State College, PA, 16801</td>
<td>F</td>
</tr>
<tr>
<td></td>
<td>Dr. &quot;Jonny&quot; John Jay Doe Jr</td>
<td>123-4567</td>
<td>10 S. Beaver, PA</td>
<td>M</td>
</tr>
</tbody>
</table>
XML

<table>
<thead>
<tr>
<th>VID</th>
<th>Category</th>
<th>Date</th>
<th>Rating</th>
<th>Att1</th>
<th>Att2</th>
<th>…</th>
<th>att1000</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>Comedy</td>
<td>2005/1/1</td>
<td>5</td>
<td>10</td>
<td></td>
<td></td>
<td>T1</td>
</tr>
<tr>
<td>200</td>
<td>Action</td>
<td>2005/1/10</td>
<td>4</td>
<td>20</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>300</td>
<td>SF</td>
<td>2004/12/31</td>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td>S20</td>
</tr>
</tbody>
</table>

```xml
<VideoTable>
  <Video vid="100" category="comedy" date="2005/1/1" rating="5" att2="10" att1000="T1" />
  <Video vid="200" category="action" date="2005/1/10" rating="4" att1="20" />
  <Video vid="300" category="SF" date="2004/12/31" rating="5" att1000="S20" />
</VideoTable>
```

XML

<table>
<thead>
<tr>
<th>Name</th>
<th>Phone</th>
<th>Address</th>
<th>Gender</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jenny</td>
<td>564-3456</td>
<td>310 N. Atherton</td>
<td>F</td>
</tr>
<tr>
<td>Dr. Jonny John Jay Doe Jr</td>
<td>123-4567</td>
<td>10 S. Beaver</td>
<td>M</td>
</tr>
</tbody>
</table>

```xml
<OwnerTable>
  <Owner phone="564-3456" gender="F">
    <Name FN="Jenny" />
    <Address>310 N. Atherton</Address>
    <City>State College</City>
    <State>PA</State>
    <Zip>16801</Zip>
  </Owner>

  <Owner phone="123-4567" gender="M">
    <Name Prefix="Dr." FN="Jonny" MN="John" LN="Doe" Suffix="Jr." />
    <Address><Adr1>10 S. Beaver</Adr1></Address>
    <State>PA</State>
  </Owner>
</OwnerTable>
```
## RDBMS vs. XML

<table>
<thead>
<tr>
<th>RDBMS</th>
<th>XML</th>
</tr>
</thead>
<tbody>
<tr>
<td>Structured model</td>
<td>Unstructured or Semi-structured model</td>
</tr>
<tr>
<td>Large scale data</td>
<td>Small to Medium scale data</td>
</tr>
<tr>
<td>Limited semantics</td>
<td>Flexible and rich semantics</td>
</tr>
<tr>
<td>Focus: how to handle large size data efficiently?</td>
<td>Focus: how to handle large number of small size data with various formats efficiently?</td>
</tr>
</tbody>
</table>

## Where to Store XML Data?

- **Document-Centric XML**
  - Native XML Database
  - Content Management System
- **Data-Centric XML**
  - Well-structured → XML-Enabled Database
  - Semi-structured → Native XML Database
XML Storage Types

- **XML-Enabled Database**
  - Can transfer data between XML documents and storage
  - Major DBMS with XML support
  - Eg, Oracle, DB2, SQL Server, MySQL, …

- **Native XML Database**
  - Store XML in native form—ie, no transformation
  - Eg, Tamino, Xindice, Xyleme, Quip, …

- **Content Management System**
  - Built on top of native XML databases and/or the file system for content/document management
  - Eg, Astoria, Cascade Server, …

---

http://www.rpbourret.com/xml/XMLDatabaseProds.htm
Public SGML/XML Software

Publicly Available Software for SGML/XML/DSSSL

Introduction

Priority is given to "public" SGML/XML software in this document database since the scope of interest is mainly the internet, where the ethic of public gift is highly esteemed. The wealth of SGML software made freely available for public use is evidence of that ethic. As a supplement to the links and information provided on public SGML software below, readers should consult Steve Peppers' "Whirlwind Guide to SGML Tools and Vendors." See the main bibliographic entry for the Whirlwind Guide for a document abstract and detailed information about its contents.

See also the detailed software summary for 207 products extracted from the technical report of Elke Kuikka and Eija Nukinen [updated January 1998] (at the list bibliographic entry, or to the overview in the "Commercial SGML Software" page. NICE Technologies (November 1998) also has an online database of SGML vendors and products (local archive copy).

Primary sections in this document include the following -- however ineluctable the taxonomy for software categories. See the Contents listing to link directly to a particular description.

- SGML Processors
- SGML/XTL Editor, Writing, and Searching Tools
- SGML Data Conversion, Transformation, and Manipulation
- SGML Editor/Compilers
- DSSSL Software Tools
- XML/XSL/TLL Software Tools

SAX: Simple API for XML

- Event-based programming (stream model)
- XML document passes by until a pre-registered event triggers

```script
do forever: # the event loop
  get an event from the input stream
  if event type == EndOfDocument : quit # break out of event loop
  if event type == ... : call the appropriate handler subroutine, passing it event information as an argument
  elif event type == ... : call the appropriate handler subroutine, passing it event information as an argument
  else: # handle an unrecognized type of event
    ignore the event, or raise an exception
```

From http://Tutorial_EventDrivenProgramming.sourceforge.net
SAX: Simple API for XML

- NO need to read whole document into the memory → very efficient for handling huge XML document
- NO in-memory data structure to manipulate XML document → inconvenient to handle complex operations
  - Eg, IDREF validation (= integrity constraint validation in relational DB)

DOM: Document Object Model

- http://www.w3c.org/DOM/
- Created during “browser war” (late 1990s)
  - Programming for JavaScript vs. JScript
- DOM level 3
  - Well supported by major web browsers who support standard ECMAScript
DOM: Document Object Model

- Language-independent specification like IDL
- Tree-based programming
  - Reads the whole XML document into the memory, and build a Tree-shaped data structure
  - Thus, can't handle huge XML document well
- Very detailed APIs make it easy to program complex manipulation of XML nodes once loaded into the memory

DOM Illustration

```xml
<?xml version="1.0"?>
<books>
  <book>
    <author>Carson</author>
    <price format="dollar">31.95</price>
    <pubdate>05/01/2001</pubdate>
  </book>
</books>
```

### SAX Example

```java
// Application program
public static void main (String args[]) throws Exception {
    XMLReader xr = XMLReaderFactory.createXMLReader();
    MySAXApp handler = new MySAXApp();
    xr.setContentHandler(handler); // register my event handler

    xr.parse(new InputSource(new FileReader("foo.xml")));
}
```

```
// My event handler
public class MySAXApp extends DefaultHandler {
    public void startDocument () {…}
    public void endDocument () {…}
    public void startElement (String uri, String name,
                                String qName, Attributes atts) {…}
    public void endElement (String uri, String name, String qName) {…}
}
```

### DOM Example

```java
// DOM supports classes like Attr, Element, Node, Text, Document, …

import org.apache.xerces.parsers.*; // use Apache’s DOM implementation
import org.w3c.dom.*;

DOMParser parser = new DOMParser();
parser.parse("foo.xml");
Document d = parser.getDocument();

NodeList employeeList = d.getElementsByTagName("employee");
for (int i = 0; i < employeeList.getLength(); i++) {
    NodeList e = employeeList.item(i).getChildNodes();
}
```

```
```
## Conclusion

- XML is an important language that one should learn
- Plenty of research issues for Database Researchers
  - XML query language issue
  - Conversion issue btw XML and other (eg, relational) models
  - Storage issue for native XML database
  - Novel indexing issue
  - System design and implementation issue

## Further References

- World-Wide Web Consortium:  
  - [www.w3c.org/xml/](http://www.w3c.org/xml/)
- XML Cover Page: [www.oasis-open.org/cover/](http://www.oasis-open.org/cover/)
- XML Articles: [www.xml.com](http://www.xml.com)
- Latest XML News: [www.xmlhack.com](http://www.xmlhack.com)
- XML Tutorial:  
  - [www.w3schools.com/xml/default.asp](http://www.w3schools.com/xml/default.asp)
  - [www.brics.dk/~amoeller/XML](http://www.brics.dk/~amoeller/XML)
- XML WIKI:  
<End haveQuestion="Yes"/>
<Thanks/>
</End>