Adapted from Jimmy Lin’s Slides
Goals

- Metadata: XML
- Databases
Outline

1. The joys and sorrows of metadata
2. XML: a framework for data representation
3. New and interesting things
4. Relational Databases
5. Relational Algebra
Take-Away Messages

- Metadata makes data useful
- XML is a way to encode data and metadata
- XML allows computers to exchange information in new and interesting ways
### What's going on here? How do I use this?

<table>
<thead>
<tr>
<th>Date</th>
<th>OL</th>
<th>950</th>
<th>24.2</th>
<th>12.6</th>
<th>1</th>
<th>-0.1</th>
<th>33.1</th>
<th>27.8</th>
<th>53.3</th>
<th>5.92</th>
</tr>
</thead>
<tbody>
<tr>
<td>7/1/1988</td>
<td>OL</td>
<td>950</td>
<td>0.4</td>
<td>16.3</td>
<td>0.4</td>
<td>0.2</td>
<td>41</td>
<td>34.5</td>
<td>6.5</td>
<td>4.56</td>
</tr>
<tr>
<td>7/2/1988</td>
<td>OL</td>
<td>950</td>
<td>32.9</td>
<td>18.9</td>
<td>1.4</td>
<td>0.3</td>
<td>29.8</td>
<td>23.7</td>
<td>6.1</td>
<td>14.23</td>
</tr>
<tr>
<td>7/3/1988</td>
<td>OL</td>
<td>950</td>
<td>20.5</td>
<td>24.9</td>
<td>1.7</td>
<td>0.5</td>
<td>18.6</td>
<td>15.3</td>
<td>3.2</td>
<td>13.92</td>
</tr>
<tr>
<td>7/4/1988</td>
<td>OL</td>
<td>950</td>
<td>42.8</td>
<td>25.6</td>
<td>2.5</td>
<td>0.6</td>
<td>23.7</td>
<td>19.9</td>
<td>3.9</td>
<td>15.18</td>
</tr>
<tr>
<td>7/5/1988</td>
<td>OL</td>
<td>950</td>
<td>23.3</td>
<td>27.8</td>
<td>0.7</td>
<td>0.8</td>
<td>27.7</td>
<td>23.5</td>
<td>4.3</td>
<td>12.33</td>
</tr>
<tr>
<td>7/6/1988</td>
<td>OL</td>
<td>950</td>
<td>49.8</td>
<td>26.2</td>
<td>2.6</td>
<td>0.6</td>
<td>40.3</td>
<td>34</td>
<td>6.3</td>
<td>22.14</td>
</tr>
<tr>
<td>7/7/1988</td>
<td>OL</td>
<td>950</td>
<td>44.8</td>
<td>25.2</td>
<td>2.5</td>
<td>0.8</td>
<td>34</td>
<td>29.2</td>
<td>4.8</td>
<td>16.76</td>
</tr>
<tr>
<td>7/8/1988</td>
<td>OL</td>
<td>950</td>
<td>47.6</td>
<td>26.9</td>
<td>2.6</td>
<td>0.7</td>
<td>47.3</td>
<td>39.6</td>
<td>7.7</td>
<td>16.13</td>
</tr>
<tr>
<td>7/9/1988</td>
<td>OL</td>
<td>950</td>
<td>36.5</td>
<td>22.6</td>
<td>1.9</td>
<td>0.6</td>
<td>36.7</td>
<td>32.6</td>
<td>4</td>
<td>15.5</td>
</tr>
<tr>
<td>7/10/1988</td>
<td>OL</td>
<td>950</td>
<td>19.5</td>
<td>18.6</td>
<td>0.4</td>
<td>0.5</td>
<td>30.2</td>
<td>39.1</td>
<td>26.29</td>
<td>11.07</td>
</tr>
<tr>
<td>7/11/1988</td>
<td>OL</td>
<td>955</td>
<td>31.7</td>
<td>15.7</td>
<td>1.5</td>
<td>0.4</td>
<td>29.7</td>
<td>25</td>
<td>4.7</td>
<td>9.49</td>
</tr>
<tr>
<td>7/12/1988</td>
<td>OL</td>
<td>955</td>
<td>23.3</td>
<td>14.5</td>
<td>1.8</td>
<td>0.8</td>
<td>23.4</td>
<td>20.7</td>
<td>2.7</td>
<td>8.14</td>
</tr>
<tr>
<td>7/13/1988</td>
<td>OL</td>
<td>955</td>
<td>23.8</td>
<td>16.6</td>
<td>1.6</td>
<td>0.6</td>
<td>27.7</td>
<td>24.1</td>
<td>3.7</td>
<td>9.17</td>
</tr>
<tr>
<td>7/14/1988</td>
<td>OL</td>
<td>934</td>
<td>32.9</td>
<td>16.7</td>
<td>2.1</td>
<td>0.7</td>
<td>34</td>
<td>28.9</td>
<td>5.1</td>
<td>9.49</td>
</tr>
<tr>
<td>7/15/1988</td>
<td>OL</td>
<td>910</td>
<td>29.2</td>
<td>20.4</td>
<td>1.9</td>
<td>0.7</td>
<td>26</td>
<td>22.3</td>
<td>3.7</td>
<td>10.44</td>
</tr>
<tr>
<td>7/16/1988</td>
<td>OL</td>
<td>952</td>
<td>44.8</td>
<td>24.8</td>
<td>2.1</td>
<td>0.8</td>
<td>31.7</td>
<td>27.5</td>
<td>4.2</td>
<td>10.75</td>
</tr>
<tr>
<td>7/17/1988</td>
<td>OL</td>
<td>929</td>
<td>33.7</td>
<td>37.1</td>
<td>1.9</td>
<td>0.6</td>
<td>34.5</td>
<td>30.1</td>
<td>4.3</td>
<td>12.02</td>
</tr>
<tr>
<td>7/18/1988</td>
<td>OL</td>
<td>917</td>
<td>34.3</td>
<td>32.9</td>
<td>2</td>
<td>0.7</td>
<td>31.4</td>
<td>26.2</td>
<td>5.1</td>
<td>12.65</td>
</tr>
<tr>
<td>7/19/1988</td>
<td>OL</td>
<td>940</td>
<td>35.7</td>
<td>24.6</td>
<td>2</td>
<td>0.8</td>
<td>23.7</td>
<td>20.4</td>
<td>3.3</td>
<td>15.5</td>
</tr>
<tr>
<td>7/20/1988</td>
<td>OL</td>
<td>923</td>
<td>47.6</td>
<td>28.9</td>
<td>2.9</td>
<td>0.8</td>
<td>67.3</td>
<td>58.9</td>
<td>8.4</td>
<td>20.87</td>
</tr>
<tr>
<td>7/21/1988</td>
<td>OL</td>
<td>930</td>
<td>58.3</td>
<td>32.6</td>
<td>2.9</td>
<td>0.7</td>
<td>58</td>
<td>59.3</td>
<td>8.7</td>
<td>22.14</td>
</tr>
<tr>
<td>7/22/1988</td>
<td>OL</td>
<td>950</td>
<td>49.3</td>
<td>29.2</td>
<td>3.4</td>
<td>0.6</td>
<td>86</td>
<td>75.1</td>
<td>10.9</td>
<td>21.19</td>
</tr>
<tr>
<td>7/23/1988</td>
<td>OL</td>
<td>906</td>
<td>51.4</td>
<td>20.9</td>
<td>3.9</td>
<td>0.6</td>
<td>94</td>
<td>82.8</td>
<td>11.2</td>
<td>25.06</td>
</tr>
<tr>
<td>7/24/1988</td>
<td>OL</td>
<td>901</td>
<td>40.5</td>
<td>16.5</td>
<td>1.7</td>
<td>0.3</td>
<td>41</td>
<td>34.4</td>
<td>6.6</td>
<td>6.54</td>
</tr>
<tr>
<td>7/25/1988</td>
<td>OL</td>
<td>900</td>
<td>35.5</td>
<td>23.6</td>
<td>1.4</td>
<td>0.1</td>
<td>41</td>
<td>35.4</td>
<td>5.6</td>
<td>3.82</td>
</tr>
<tr>
<td>7/26/1988</td>
<td>OL</td>
<td>905</td>
<td>47.9</td>
<td>17.6</td>
<td>0.8</td>
<td>0.1</td>
<td>18.3</td>
<td>15.9</td>
<td>2.3</td>
<td>4.19</td>
</tr>
<tr>
<td>7/27/1988</td>
<td>OL</td>
<td>915</td>
<td>38</td>
<td>22.5</td>
<td>1.5</td>
<td>0.1</td>
<td>30</td>
<td>25.3</td>
<td>4.7</td>
<td>4.44</td>
</tr>
<tr>
<td>7/28/1988</td>
<td>OL</td>
<td>918</td>
<td>21.2</td>
<td>8.8</td>
<td>1.1</td>
<td>-0.1</td>
<td>24.7</td>
<td>21.1</td>
<td>3.6</td>
<td>4.81</td>
</tr>
<tr>
<td>7/29/1988</td>
<td>OL</td>
<td>904</td>
<td>38.5</td>
<td>22.8</td>
<td>2.1</td>
<td>0.3</td>
<td>54</td>
<td>46.8</td>
<td>7.2</td>
<td>9.8</td>
</tr>
<tr>
<td>7/30/1988</td>
<td>OL</td>
<td>911</td>
<td>94</td>
<td>32.6</td>
<td>2.1</td>
<td>0.3</td>
<td>45.5</td>
<td>38.9</td>
<td>6.6</td>
<td>9.49</td>
</tr>
<tr>
<td>7/31/1988</td>
<td>OL</td>
<td>955</td>
<td>58.3</td>
<td>43.1</td>
<td>2.5</td>
<td>1.1</td>
<td>41</td>
<td>33.1</td>
<td>7.9</td>
<td>9.8</td>
</tr>
<tr>
<td>8/1/1988</td>
<td>OL</td>
<td>951</td>
<td>55.8</td>
<td>42.2</td>
<td>2.1</td>
<td>0.8</td>
<td>38</td>
<td>31</td>
<td>7.1</td>
<td>8.86</td>
</tr>
</tbody>
</table>
Metadata

Literally “data about data”

“a set of data that describes and gives information about other data” – Oxford English Dictionary
Dublin Core
What is the Dublin Core?

- A metadata standard for describing digital resources
- An initiative to create a library card catalog for the Web
- Dublin Core fields:

  - Title
  - Description
  - Date
  - Identifier
  - Relation
  - Creator
  - Publisher
  - Source
  - Coverage
  - Subject
  - Contributor
  - Type
  - Format
  - Language
  - Rights
Encoding Metadata

- Language for encoding metadata should be:
  - Universal - so all can understand
  - Flexible - to incorporate different types
  - Extensible - flexible to custom types
  - Simple - to encourage adoption
  - Modular - so that schemes can be mixed, extended
How do we encode data for interoperability?

**Challenges**

January 31, 2001
31 janvier 2001
2001-01-31
01-31-2000
980942400
Outline

1. The joys and sorrows of metadata
2. XML: a framework for data representation
3. New and interesting things
4. Relational Databases
5. Relational Algebra
What is XML?

- XML = eXtensible Markup Language
- XML is a standard for exchanging structured data
  - Provides standardization at the syntactic level
  - Does not provide “meaning” for the tags
  - XML is a standard recommended by the W3C
Goals of XML

- Easy to use
- Easy to extend and adapt
- Easy to write programs that use XML
- Support a wide variety of applications
- Should be human legible
- Formal and concise
Attribute

\[ \text{Attribute} \]

\[<\text{person} \ attribute="28" />\]

Element

\[ \text{Element} \]

\[<\text{person}>
<\text{age}>28</\text{age}>
</\text{person}>\]
The Basic Rules

- XML is case sensitive
- All start tags must have end tags
- Elements must be properly nested
- XML declaration is the first statement
  ```xml
  <?xml version="1.0"?>
  ```
- Every document must contain a root element
- Attribute values must have quotation marks
  ```xml
  <item id="33905">
  ```
- Certain characters are reserved for parsing
  ```
  \&lt; = ‘<’
  ```
<rdf:RDF
 xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
 xmlns:dc="http://purl.org/dc/elements/1.1/"/>

<rdf:Description
 rdf:about="http://media.example.com/audio/guide.ra">

<dc:creator>Rose Bush</dc:creator>
<dc:description>Describes process for planting and nurturing different kinds of rose bushes.</dc:description>
<dc:date>2001-01-20</dc:date>
</rdf:Description>

</rdf:RDF>
What does XML do?
• What does XML do? … **nothing**
• Syntax vs. semantics
• XML vs HTML
Historic Perspective: Three Core Technologies

- HTTP - HyperText Transfer Protocol
  - A protocol for transferring data between machines on the Internet
- URL - Uniform Resource Locator
  - A scheme for referencing the specific location of a resource
- HTML - HyperText Markup Language
  - A markup language for encoding information to be read by humans

HTTP and URLs have stood the test of time.

But by 1996, HTML was already showing signs of age . . .
Started with very few tags

Language evolved as more tags were added:
- Forms
- Tables
- Fonts
- Frames
- ...
Problems with HTML

- I want personalized tags
  - HTML can’t be extended
- I want to incorporate other types of data
  - Mathematics, database entries, literary text, poems, purchase orders...
    - HTML can’t accommodate other types of data
- I want to process pages automatically with software
  - HTML is too messy and inconsistent
  - Browsers are too forgiving
Back to Basics

- HTML was defined using SGML
  - Standard Generalized Markup Language
  - A meta-language for defining languages
  - Complex, sophisticated, powerful . . .
Back to Basics

- HTML was defined using SGML
  - Standard Generalized Markup Language
  - A meta-language for defining languages
  - Complex, sophisticated, powerful ... too difficult to use
  - Idea: create a simpler version of SGML ...
Back to Basics

- HTML was defined using SGML
  - Standard Generalized Markup Language
  - A meta-language for defining languages
  - Complex, sophisticated, powerful . . . too difficult to use
  - Idea: create a simpler version of SGML . . . the birth of XML!
XML can be used to define other languages

Many XML languages, optimized for different roles

- XHTML: HTML by XML rules
- MathML: for mathematics
- EPUB: for creating eBooks
- RSS: for news feeds
- Civ IV: Create your own game
- SVG: Create graphics
What's new?

New preamble to tell us what’s here, and tags must have explicit ends.
An XML language for defining mathematic formulas

\[ x^2 + 4x + 4 = 0 \]
EPUB

- Format for putting books on mobile readers (except Kindles)
- Divide up a book into XHTML files
- Create two additional XML files
  - `opf` (open packaging format)
    - Metadata (using Dublin Core)
    - All the files needed
    - Linear reading order
  - `ncx` (navigation control file for XML)
    - Hierarchical organization of content (for easy navigation)
• RSS = Really Simple Syndication or Rich Site Summary
• An XML format for distributing news headlines on the Web
And Others . . .

- CML: chemical Markup Lang
- CellML: biological models
- BSML: bioinformatic sequences
- MAGE-ML: Microarray Gene Expression
- XSTAR: for archaeological research
- MARCXML: MARC in XML
- AML: astronomy markup language
- SportsML: for sharing sports data
- List goes on and on and on . . .
The XML Family Tree

![Diagram showing the XML family tree with layers from SGML to HTML, TEI, XHTML, MathML, RDF, SpeechML, SMIL, and XUL.](image-url)
Mixing XML Dialects

- XML is designed to support the integration of multiple standards
- Allows users to mix elements from different standards
  - Snapping together XML dialects like Lego pieces
  - Based on the notion of “namespaces”
Example

```xml
<?xml version="1.0" ?>
<rdf:RDF
 xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
 xmlns:rss="http://purl.org/rss/1.0/
 xmlns:dc="http://purl.org/dc/elements/1.1/>
/rss:channel rdf:about="http://www.xml.com/xml/news.rss">
  <rss:title>XML.com</rss:title>
  <rss:link>http://xml.com/pub</rss:link>
  <dc:description>
    XML.com features a rich mix of information and services for the XML community.
  </dc:description>
  <dc:subject>XML, RDF, metadata, information syndication services</dc:subject>
  <dc:identifier>http://www.xml.com</dc:identifier>
  <dc:publisher>O'Reilly & Associates, Inc.</dc:publisher>
  <dc:rights>Copyright 2000, O'Reilly & Associates, Inc.</dc:rights>
</rss:channel>
</rdf:RDF>
```
Another Example

```xml
<?xml version="1.0" encoding="iso-8859-1" ?>
<html xmlns="http://www.w3.org/TR/xhtml1" >
<head>
  <title> Title of XHTML Document </title>
</head>
<body>
  <div class="myDiv">
    <h1> Heading of Page </h1>
    <math xmlns="http://www.w3.org/1998/Math/MathML">
      ... MathML markup ...
    </math>
    <p> more html stuff goes here </p>
    <smil xmlns="http://www.w3.org/TR/smil1">
      ... SMIL markup ...
    </smil>
  </div>
</body>
</html>
```
1. The joys and sorrows of metadata

2. XML: a framework for data representation

3. New and interesting things

4. Relational Databases

5. Relational Algebra
Interoperability

- What does it mean and what’s the role of XML?
Interoperability

What does it mean and what's the role of XML?

**XML: universal format for data interchange**

Software exchanges data as XML-format messages

**Advantages?**
- Eliminates proprietary data formats
- Promotes interoperability
- Encourages cooperation
- Leverages lots of existing XML processing software
XML Messaging

Diagram:

- Factory
- Place order
- Response
- Supplier
- Supplier
- Supplier
XML Messaging
What’s in it for me?

- **Webapps**
  - Lower overhead
  - Richer data
  - More portability

- **Mashups**

- **Syntax vs. Semantics**
Mashups
Mashups
XML Schema

- Defines what a valid XML document should look like
  - Fields
  - Attributes
  - Number of entries
- Has filename extension “xsd”
- There are plenty of XML validators out there
- Won’t go into details . . . think of it like a rulebook
XSLT transforms one XML document into another
- Often used to display XML to a user
  - Webpage
  - Graphics
- Syntax varies, semantics are fixed
<?xml version="1.0"?>

<card type="simple">
   <name>John Doe</name>
   <title>CEO, Widget Inc.</title>
   <email>john.doe@widget.com</email>
   <phone>(202) 456-1414</phone>
</card>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0" xmlns="http://www.w3.org/1999/xhtml">
  <xsl:template match="card">
    <html>
      <head><title>business card</title></head>
      <body>
        <xsl:apply-templates select="name"/>
        <xsl:apply-templates select="title"/>
        <xsl:apply-templates select="email"/>
        <xsl:apply-templates select="phone"/>
      </body>
    </html>
  </xsl:template>
</xsl:stylesheet>
<xsl:stylesheet>
  <xsl:template match="name">
    <h1><xsl:value-of select="text()" /></h1>
  </xsl:template>

  <xsl:template match="title">
    <b>Title:</b> <xsl:value-of select="text()" /><br/>
  </xsl:template>

  <xsl:template match="email">
    <b>Email:</b> <a href="mailto:{text()}">
      <tt><xsl:value-of select="text()" /></tt>
    </a><br/>
  </xsl:template>

  <xsl:template match="phone">
    <b>Phone:</b> <xsl:value-of select="text()" /><br/>
  </xsl:template>
</xsl:stylesheet>
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="card_style1.xml"?>

<card type="simple">
  <name>John Doe</name>
  <title>CEO, Widget Inc.</title>
  <email>john.doe@widget.com</email>
  <phone>(202) 456-1414</phone>
</card>
XML isn’t all there is

- S-Expressions
  - Based on logical statements
  - Not used outside academia (not in it too much, either)

- Protocol Buffers
  - Blazingly fast
  - More constrained than XML (have to specify data types, ranges)

- JSON
  - Designed specifically for web applications
  - Lighter weight than XML
Take-Away Messages

- Metadata makes data useful
- XML is a way to encode data and metadata
- XML allows computers to exchange information in new and interesting ways
The joys and sorrows of metadata

XML: a framework for data representation

New and interesting things

Relational Databases

Relational Algebra
Take-Away Messages

- Databases are suitable for storing structured information
- Databases are important tools to organize, manipulate, and access structured information
- Databases are integral components of modern Web applications
Definitions

Structured Information
What you put in a database (e.g. from XML)

Database
What you put structured information in.

Database Management System (DBMS)
Software system designed to store, manage, and facilitate access to databases
What’s a database?

An integrated collection of data organized according to some model . . .
What’s a relational database?

An integrated collection of data organized according to a relational model...
Databases (try to) model reality.

- **Entities**: things in the world (Example: airlines, tickets, passengers)
- **Relationships**: how different things are related (Example: the tickets each passenger bought)
- **“Business Logic”**: rules about the world (Example: fare rules)
Components of a Relational Database

- **Field**: an “atomic” unit of data
- **Record**: a collection of related fields
- **Table**: a collection of related records
  - Each record is a row in the table
  - Each field is a column in the table
- **Database**: a collection of tables
## A Simple Example

<table>
<thead>
<tr>
<th>Name</th>
<th>DOB</th>
<th>SSN</th>
</tr>
</thead>
<tbody>
<tr>
<td>John Doe</td>
<td>04/15/1970</td>
<td>153-78-9082</td>
</tr>
<tr>
<td>Jane Smith</td>
<td>08/31/1985</td>
<td>768-91-2376</td>
</tr>
<tr>
<td>Mary Adams</td>
<td>11/05/1972</td>
<td>891-13-3057</td>
</tr>
</tbody>
</table>

Field Name

Record

Field

Primary Key
Components of a Relational Database

Why “Relational?”

View of the world in terms of entities and relations between them

- Tables represent “relations”
- Each row in the table is sometimes called a “tuple”
- Each tuple is “about” an entity
- Fields can be interpreted as “attributes” or “properties” of the entity

Data is manipulated by “relational algebra”:

- Defines things you can do with tuples
- Expressed in SQL (Structured Query Language, next week)
The Registrar Example

What do we need to know?

- Something about the students? (e.g., first name, last name, email, department)
- Something about the courses? (e.g., course ID, description, enrolled students, grades)
- Which students are in which courses

How do we capture these things?
## Put everything in a big table...

<table>
<thead>
<tr>
<th>Student ID</th>
<th>Last Name</th>
<th>First Name</th>
<th>Dept ID</th>
<th>Dept</th>
<th>Course ID</th>
<th>Course name</th>
<th>Grade</th>
<th>email</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Arrows</td>
<td>John</td>
<td>EE</td>
<td>EE</td>
<td>lbsc690</td>
<td>Information Technology</td>
<td>90</td>
<td>j.arrows@wam</td>
</tr>
<tr>
<td>1</td>
<td>Arrows</td>
<td>John</td>
<td>EE</td>
<td>Elec Engin</td>
<td>ee750</td>
<td>Communication</td>
<td>95</td>
<td>j.a.2002@yahoo</td>
</tr>
<tr>
<td>2</td>
<td>Peters</td>
<td>Kathy</td>
<td>HIST</td>
<td>HIST</td>
<td>lbsc690</td>
<td>Information Technology</td>
<td>95</td>
<td>k.peters2@wam</td>
</tr>
<tr>
<td>2</td>
<td>Peters</td>
<td>Kathy</td>
<td>HIST</td>
<td>history</td>
<td>hist405</td>
<td>American History</td>
<td>80</td>
<td>k.peters2@wma</td>
</tr>
<tr>
<td>3</td>
<td>Smith</td>
<td>Chris</td>
<td>HIST</td>
<td>history</td>
<td>hist405</td>
<td>American History</td>
<td>90</td>
<td>smith2002@glue</td>
</tr>
<tr>
<td>4</td>
<td>Smith</td>
<td>John</td>
<td>CLIS</td>
<td>Info Sci</td>
<td>lbsc690</td>
<td>Information Technology</td>
<td>98</td>
<td>js03@wam</td>
</tr>
</tbody>
</table>
A first stab...

What’s wrong with this?

<table>
<thead>
<tr>
<th>Student ID</th>
<th>Last Name</th>
<th>First Name</th>
<th>Dept ID</th>
<th>Dept</th>
<th>Course ID</th>
<th>Course name</th>
<th>Grade</th>
<th>email</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Arrows</td>
<td>John</td>
<td>EE</td>
<td>EE</td>
<td>lbsc690</td>
<td>Information Technology</td>
<td>90</td>
<td>jarrows@wam</td>
</tr>
<tr>
<td>1</td>
<td>Arrows</td>
<td>John</td>
<td>EE</td>
<td>Elec Engin</td>
<td>ee750</td>
<td>Communication</td>
<td>95</td>
<td>ja_2002@yahoo</td>
</tr>
<tr>
<td>2</td>
<td>Peters</td>
<td>Kathy</td>
<td>HIST</td>
<td>HIST</td>
<td>lbsc690</td>
<td>Informatico Technology</td>
<td>95</td>
<td>kpeters2@wam</td>
</tr>
<tr>
<td>2</td>
<td>Peters</td>
<td>Kathy</td>
<td>HIST</td>
<td>history</td>
<td>hist405</td>
<td>American History</td>
<td>80</td>
<td>kpeters2@wma</td>
</tr>
<tr>
<td>3</td>
<td>Smith</td>
<td>Chris</td>
<td>HIST</td>
<td>history</td>
<td>hist405</td>
<td>American History</td>
<td>90</td>
<td>smith2002@glue</td>
</tr>
<tr>
<td>4</td>
<td>Smith</td>
<td>John</td>
<td>CLIS</td>
<td>Info Sci</td>
<td>lbsc690</td>
<td>Information Technology</td>
<td>98</td>
<td>js03@wam</td>
</tr>
</tbody>
</table>
Goals of “Normalization”

- Save space
  - Save each fact only once
- More rapid updates
  - Every fact only needs to be updated once
- More rapid search
  - Finding something once is good enough
- Avoid inconsistency
  - Changing data once changes it everywhere
### Student Table

<table>
<thead>
<tr>
<th>Student ID</th>
<th>Last Name</th>
<th>First Name</th>
<th>Department</th>
<th>Email</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Arrows</td>
<td>John</td>
<td>EE</td>
<td>arrows@wam</td>
</tr>
<tr>
<td>2</td>
<td>Peters</td>
<td>Kathy</td>
<td>HIST</td>
<td>kpeters2@wam</td>
</tr>
<tr>
<td>3</td>
<td>Smith</td>
<td>Chris</td>
<td>HIST</td>
<td>smith2002@glue</td>
</tr>
<tr>
<td>4</td>
<td>Smith</td>
<td>John</td>
<td>CLIS</td>
<td>js03@wam</td>
</tr>
</tbody>
</table>

### Department Table

<table>
<thead>
<tr>
<th>Department ID</th>
<th>Department</th>
</tr>
</thead>
<tbody>
<tr>
<td>EE</td>
<td>Electrical Engineering</td>
</tr>
<tr>
<td>HIST</td>
<td>History</td>
</tr>
<tr>
<td>CLIS</td>
<td>Information Studies</td>
</tr>
</tbody>
</table>

### Course Table

<table>
<thead>
<tr>
<th>Course ID</th>
<th>Course Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>lbsc690</td>
<td>Information Technology</td>
</tr>
<tr>
<td>ee750</td>
<td>Communication</td>
</tr>
<tr>
<td>hist405</td>
<td>American History</td>
</tr>
</tbody>
</table>

### Enrollment Table

<table>
<thead>
<tr>
<th>Student ID</th>
<th>Course ID</th>
<th>Grade</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>lbsc690</td>
<td>90</td>
</tr>
<tr>
<td>1</td>
<td>ee750</td>
<td>95</td>
</tr>
<tr>
<td>2</td>
<td>lbsc690</td>
<td>95</td>
</tr>
<tr>
<td>2</td>
<td>hist405</td>
<td>80</td>
</tr>
<tr>
<td>3</td>
<td>hist405</td>
<td>90</td>
</tr>
<tr>
<td>4</td>
<td>lbsc690</td>
<td>98</td>
</tr>
</tbody>
</table>
## Updated Organization

### Student Table
<table>
<thead>
<tr>
<th>Student ID</th>
<th>Last Name</th>
<th>First Name</th>
<th>Department ID</th>
<th>Email</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Arrows</td>
<td>John</td>
<td>EE</td>
<td>arrows@wam</td>
</tr>
<tr>
<td>2</td>
<td>Peters</td>
<td>Kathy</td>
<td>HIST</td>
<td>kpeters2@wam</td>
</tr>
<tr>
<td>3</td>
<td>Smith</td>
<td>Chris</td>
<td>HIST</td>
<td>smith2002@glue</td>
</tr>
<tr>
<td>4</td>
<td>Smith</td>
<td>John</td>
<td>CLIS</td>
<td>s03@wam</td>
</tr>
</tbody>
</table>

### Department Table
<table>
<thead>
<tr>
<th>Department ID</th>
<th>Department</th>
</tr>
</thead>
<tbody>
<tr>
<td>EE</td>
<td>Electrical Engineering</td>
</tr>
<tr>
<td>HIST</td>
<td>History</td>
</tr>
<tr>
<td>CLIS</td>
<td>Information Studies</td>
</tr>
</tbody>
</table>

### Course Table
<table>
<thead>
<tr>
<th>Course ID</th>
<th>Course Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>lbsc690</td>
<td>Information Technology</td>
</tr>
<tr>
<td>ee750</td>
<td>Communication</td>
</tr>
<tr>
<td>hist405</td>
<td>American History</td>
</tr>
</tbody>
</table>

### Enrollment Table
<table>
<thead>
<tr>
<th>Student ID</th>
<th>Course ID</th>
<th>Grade</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>lbsc690</td>
<td>90</td>
</tr>
<tr>
<td>1</td>
<td>ee750</td>
<td>95</td>
</tr>
<tr>
<td>2</td>
<td>lbsc690</td>
<td>95</td>
</tr>
<tr>
<td>2</td>
<td>hist405</td>
<td>80</td>
</tr>
<tr>
<td>3</td>
<td>hist405</td>
<td>90</td>
</tr>
<tr>
<td>4</td>
<td>lbsc690</td>
<td>98</td>
</tr>
</tbody>
</table>
Keys

- “Primary Key” uniquely identifies a record
  - e.g., student ID in the student table
- “Foreign Key” is primary key in the other table
  - It need not be unique in this table
Approaches to Normalization

- For simple problems:
  - Start with the entities you’re trying to model
  - Group together fields that “belong together”
  - Add keys where necessary to connect entities in different tables

- For more complicated problems:
  - Entity-relationship modeling (LBSC 670)
Entity Relationship Modeling

- **Enrollment**
  - Student
  - Course
  - Grade

- **Student**
  - Student ID
  - First name
  - Last name
  - Department
  - E-mail

- **Course**
  - Course ID
  - Course Name

- **Department**
  - Department ID
  - Department Name

- **has**
- **associated with**
Entity Relationship Modeling

- Many-to-Many
- One-to-Many
- One-to-One
Database Integrity

- Registrar database must be internally consistent
  - All enrolled students must have an entry in the student table
  - All courses must have a name
  - Grades can’t be negative

- What happens:
  - When a student withdraws from the university?
  - When a course is taken off the books?
Integrity Constraints

- Conditions that must be true of the database at any time
  - Specified when the database is designed
  - Checked when the database is modified
- RDBMS ensures that integrity constraints are always kept
  - So that database contents remain faithful to the real world
  - Helps avoid data entry errors
- Where do integrity constraints come from?
1. The joys and sorrows of metadata
2. XML: a framework for data representation
3. New and interesting things
4. Relational Databases
5. Relational Algebra
# Relational Operations

## Student Table

<table>
<thead>
<tr>
<th>Student ID</th>
<th>Last Name</th>
<th>First Name</th>
<th>Department ID</th>
<th>Email</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Arrows</td>
<td>John</td>
<td>EE</td>
<td>j.arrows@wam</td>
</tr>
<tr>
<td>2</td>
<td>Peters</td>
<td>Kathy</td>
<td>HIST</td>
<td>k.peters@wam</td>
</tr>
<tr>
<td>3</td>
<td>Smith</td>
<td>Chris</td>
<td>HIST</td>
<td>smith2002@glue</td>
</tr>
<tr>
<td>4</td>
<td>Smith</td>
<td>John</td>
<td>CLIS</td>
<td>js03@wam</td>
</tr>
</tbody>
</table>

## Department Table

<table>
<thead>
<tr>
<th>Department ID</th>
<th>Department</th>
</tr>
</thead>
<tbody>
<tr>
<td>EE</td>
<td>Electrical Engineering</td>
</tr>
<tr>
<td>HIST</td>
<td>History</td>
</tr>
<tr>
<td>CLIS</td>
<td>Information Studies</td>
</tr>
</tbody>
</table>

## "Joined" Table

<table>
<thead>
<tr>
<th>Student ID</th>
<th>Last Name</th>
<th>First Name</th>
<th>Dept ID</th>
<th>Department</th>
<th>Email</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Arrows</td>
<td>John</td>
<td>EE</td>
<td>Electrical Engineering</td>
<td>j.arrows@wam</td>
</tr>
<tr>
<td>2</td>
<td>Peters</td>
<td>Kathy</td>
<td>HIST</td>
<td>History</td>
<td>k.peters@wam</td>
</tr>
<tr>
<td>3</td>
<td>Smith</td>
<td>Chris</td>
<td>HIST</td>
<td>History</td>
<td>smith2002@glue</td>
</tr>
<tr>
<td>4</td>
<td>Smith</td>
<td>John</td>
<td>CLIS</td>
<td>Information Studies</td>
<td>js03@wam</td>
</tr>
</tbody>
</table>
### Relational Operations

#### Representing and Storing Structured Data

**Table:**

<table>
<thead>
<tr>
<th>Student ID</th>
<th>Last Name</th>
<th>First Name</th>
<th>Dept ID</th>
<th>Department</th>
<th>email</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Arrows</td>
<td>John</td>
<td>EE</td>
<td>Electrical Engineering</td>
<td>arrows@wam</td>
</tr>
<tr>
<td>2</td>
<td>Peters</td>
<td>Kathy</td>
<td>HIST</td>
<td>History</td>
<td>ktpeters2@wam</td>
</tr>
<tr>
<td>3</td>
<td>Smith</td>
<td>Chris</td>
<td>HIST</td>
<td>History</td>
<td>smith2002@glue</td>
</tr>
<tr>
<td>4</td>
<td>Smith</td>
<td>John</td>
<td>CLIS</td>
<td>Information Studies</td>
<td>js03@wam</td>
</tr>
</tbody>
</table>

**Filtering with WHERE clause:**

```
WHERE Department ID = "HIST"
```
Relational Operations

```
<table>
<thead>
<tr>
<th>Student ID</th>
<th>Last Name</th>
<th>First Name</th>
<th>Dept ID</th>
<th>Department</th>
<th>email</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Arrows</td>
<td>John</td>
<td>EE</td>
<td>Electrical Engineering</td>
<td>jarrows@wam</td>
</tr>
<tr>
<td>2</td>
<td>Peters</td>
<td>Kathy</td>
<td>HIST</td>
<td>History</td>
<td>kpeters2@wam</td>
</tr>
<tr>
<td>3</td>
<td>Smith</td>
<td>Chris</td>
<td>HIST</td>
<td>History</td>
<td>smith2002@glue</td>
</tr>
<tr>
<td>4</td>
<td>Smith</td>
<td>John</td>
<td>CLIS</td>
<td>Information Studies</td>
<td>js03@wam</td>
</tr>
</tbody>
</table>
```

SELECT Student ID, Department

```
<table>
<thead>
<tr>
<th>Student ID</th>
<th>Department</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Electrical Engineering</td>
</tr>
<tr>
<td>2</td>
<td>History</td>
</tr>
<tr>
<td>3</td>
<td>History</td>
</tr>
<tr>
<td>4</td>
<td>Information Studies</td>
</tr>
</tbody>
</table>
```
Relational Operations

- Joining tables: JOIN
- Choosing columns: SELECT

Based on their labels (field names)

- Choosing rows: WHERE

Based on their contents

- These can be specified together
How is a database more than a spreadsheet?
Database in the “Real World”

Typical database applications:

- Banking (e.g., saving/checking accounts)
- Trading (e.g., stocks)
- Traveling (e.g., airline reservations)
- Networking (e.g., Facebook)

Characteristics:

- Lots of data
- Lots of concurrent operations
- Must be fast
- “Mission critical” (well . . . sometimes)
Operational Requirements

- Must hold a lot of data
  - Use lots of computers, each with a small slice
  - So which machine has your data?
- Must be reliable
  - Use lots of computers with duplicate copies
  - How do you keep copies consistent
- Must be fast
  - Use lots of computers
  - Share the load
- Must support concurrent operations
  - This is hard
  - But often not needed
Database Transactions

- **Transaction** = sequence of database actions grouped together
  - e.g., transfer $500 from checking to savings
- **ACID properties:**
  - **Atomicity**: all-or-nothing
  - **Consistency**: each transaction must take the DB between consistent states
  - **Isolation**: concurrent transactions must appear to run in isolation
  - **Durability**: results of transactions must survive even if systems crash
Idea: keep a log (history) of all actions carried out while executing transactions

- Before a change is made to the database, the corresponding log entry is forced to a safe location

Recovering from a crash:

- Effects of partially executed transactions are undone
- Effects of committed transactions are redone
- Trickier than it sounds!
**Discussion Question**

### RideFinder

Design a database to match drivers with passengers (e.g., for road trips)

- Drivers post available seats; they want to know about interested passengers
- Passengers call up looking for rides: they want to know about available rides (they don’t get to post “rides wanted” ads)
- These things happen in no particular order
Discussion Goals

- Design the tables you will need
  - First decide what information you need to keep track of
  - Then design tables to capture this information

- Design queries (using join, project, and restrict)
  - What happens when a passenger comes looking for a ride?
  - What happens when a driver comes to find out who his passengers are?

- Role play!
Exercise solution: tables

- **Ride**: Ride ID, Driver ID, Origin, Destination, Departure Time, Arrival Time, Available Seats
- **Passenger**: Passenger ID, Name, Address, Phone Number
- **Driver**: Driver ID, Name, Address, Phone Number
- **Booking**: Ride ID, Passenger ID
Exercise solution: queries

- **Passenger calls: Can I get a ride?**
  - Join: Ride, Driver
  - Project: Departure Time, Name, Phone Number
  - Restrict: Origin, Destination, Available Seats $> 0$

- **Driver calls: Who are my passengers?**
  - Join: Ride, Passenger, Booking
  - Project: Name, Phone Number
  - Restrict: (Driver) Name, Origin, Destination, Departure Time