



College of Information Studies

University of Maryland Hornbake Library Building College Park, MD 20742-4345

The Web

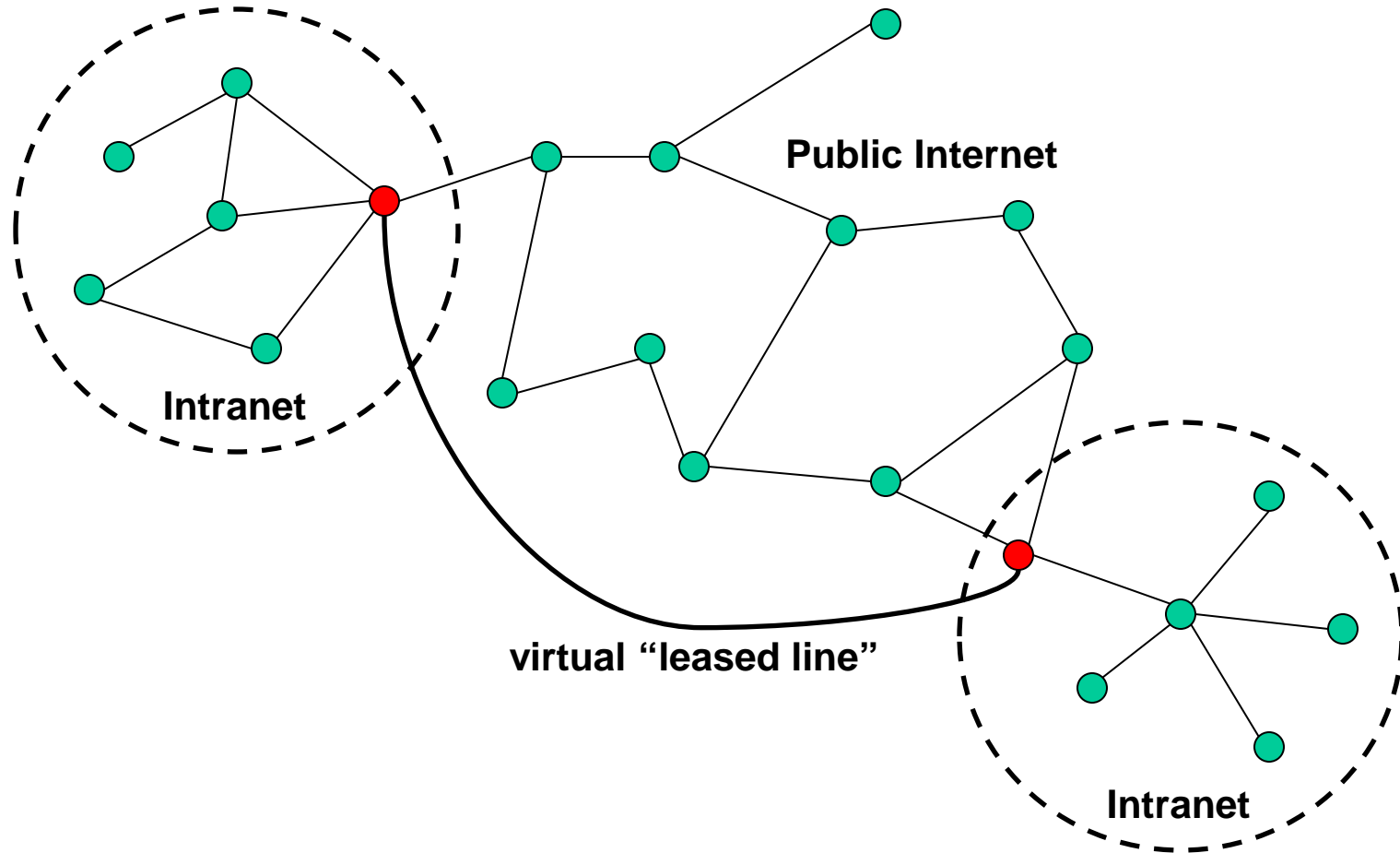
Week 10

LBSC 671

Creating Information Infrastructures

Virtual Private Networks

a secure private network over the public Internet



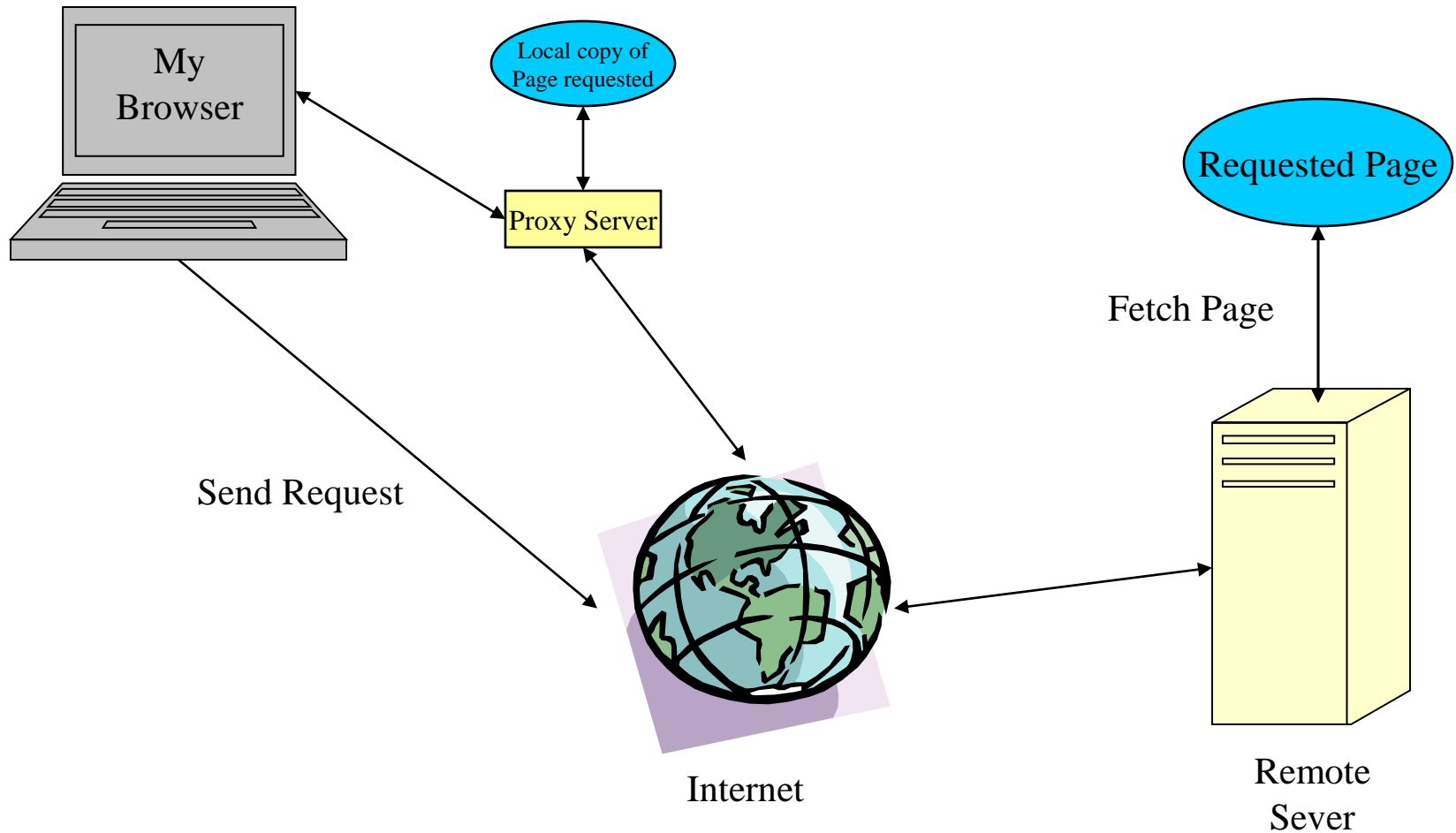
Tonight

- Learn to create a Web page
- Think about what the Web “is”
- Talk conceptually about databases

Internet \neq Web

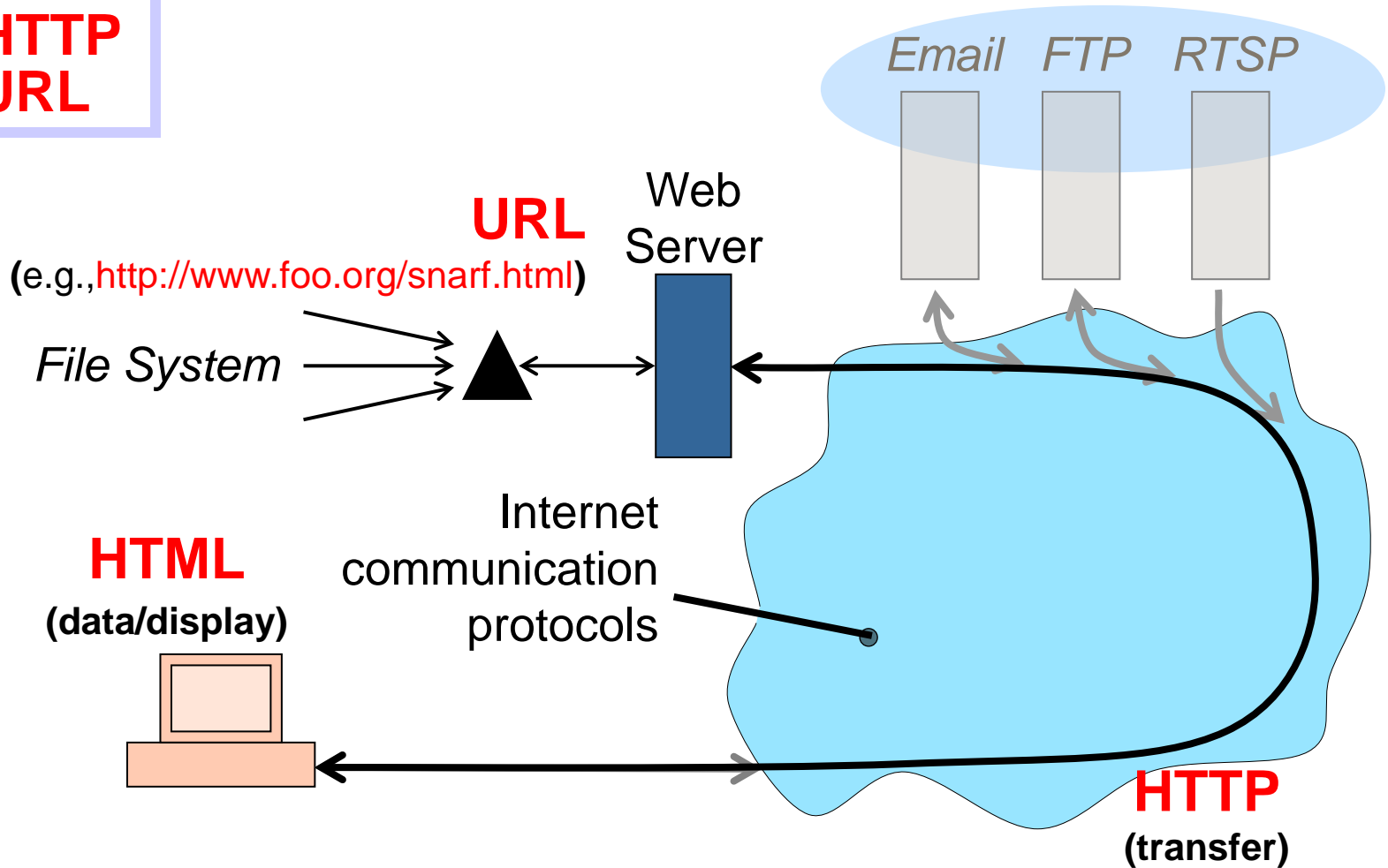
- Internet: collection of global networks
- Web: way of managing information exchange
- There are many other uses for the Internet
 - File transfer (FTP)
 - Email (SMTP, POP, IMAP)

The World-Wide Web



“The Web”

HTML
HTTP
URL

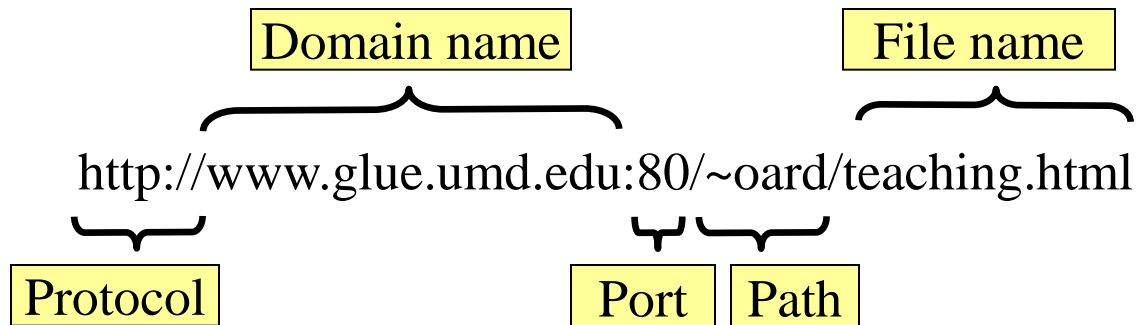


Web Standards

- HTML
 - How to write and interpret the information
- URL
 - Where to find it
- HTTP
 - How to get it

Uniform Resource Locator (URL)

- Uniquely identify Web pages



HyperText Markup Language (HTML)

- Simple document structure language for Web
- Advantages
 - Adapts easily to different display capabilities
 - Widely available display software (browsers)
- Disadvantages
 - Does not directly control layout

“Hello World” HTML

This is the header

```
<html>  
<head>  
<title>Hello World!</title>  
</head>
```

```
<body>  
  
<p>Hello world! This is my first webpage!</p>  
  
</body>  
</html>
```

This is the actual content of the HTML document

Hands On:

Learning HTML From Examples

- Use Internet Explorer to find a page you like
 - <http://terpconnect.umd.edu/~oard>
- On the “View” menu select “Source” (in IE)
 - Opens a notepad window with the source
- Compare HTML source with the Web page
 - Observe how each effect is achieved

Hands On: “Adopt” a Web Page

- Modify the HTML source using notepad
 - For example, change the page to yours
- Save the HTML source somewhere
 - In the “File” menu, select “Save As”
 - **Put the name in quotes (e.g., “test.html”)**
- FTP it to your ../pub directory on terpconnect
- View it
 - [http://terpconnect.umd.edu/~\(yourlogin\)/test.html](http://terpconnect.umd.edu/~(yourlogin)/test.html)

Tips

- Edit files on your own machine
 - Upload when you're happy
- Save early, save often!
- Reload browser to see changes
- File naming
 - **Don't use spaces**
 - Punctuation matters

What's a Document?

- Content
- Structure
 - Logical, Physical
- Appearance
 - Cascading Style Sheets
- Behavior
 - JavaScript

HTML Document Structure

- “Tags” mark structure
 - `<html>a document</html>`
 - `an ordered list`
 - `<i>something in italics</i>`
- Tag name in angle brackets `<>`
 - Not case sensitive (unlike XML)
- Open/Close pairs
 - Close tag is sometimes optional (unlike XML)

Logical Structure Tags

- Head
 - Title
- Body
 - Headers: `<h1>` `<h2>` `<h3>` `<h4>` `<h5>`
 - Lists: ``, `` (can be nested)
 - Paragraphs: `<p>`
 - Definitions: `<dt>``<dd>`
 - Tables: `<table>` `<tr>` `<td>` `</td>` `</tr>` `</table>`
 - Role: `<cite>`, `<address>`, ``, ...


Physical Structure Tags

- Bold: ``
- Italics: `<i></i>`
- Typeface: ``
- Size: ``
- Color: ``

(Hyper)Links

index.html

```
<html>
<head>
<title>Hello World!</title>
</head>
<body>
<p>Hello world! This is my first webpage!</p>
<p>Click <a href="test.html">here</a> for another page.</p>
</body>
</html>
```



test.html

```
<html>
<head>
<title>Another page</title>
</head>
<body>
<p>This is another page.</p>
</body>
</html>
```

Hypertext “Anchors”

- Internal anchors: somewhere on the same page
 - ` Students`
 - Links to: `Student Information`
- External anchors: to another page
 - `iSchool`
 - `email papers`
- URL may be complete, or relative to current page
 - `2`
- File name (in URL) is case sensitive (on Unix servers)
 - Protocol and domain name are not case sensitive

Images

- `` *or* ``
 - ``
 - SRC: can be url or path/file
 - ALT: a text string
 - ALIGN: position of the image
 - WIDTH and HEIGHT: size of the image
- Can use as anchor:
 - ``
- Example:
 - <http://www.umiacs.umd.edu/~daqingd/Image-Alignment.html>

Tables

eenie	mennie	miney
mo	catch	a tiger
by	the	toe

Table Example

```
<table align="center">
<caption align="right">The caption</caption>
<tr align="LEFT">
  <th> Header1 </th>
  <th> Header2</th>
</tr>
<tr><td>first row, first item </td>
  <td>first row, second item</td></tr>
<tr><td>second row, first item</td>
  <td>second row, second item</td></tr>
</table>
```

XHTML: Cleaning up HTML

```
<?xml version="1.0" encoding="iso-8859-1"?>
<html xmlns="http://www.w3.org/TR/xhtml1" >
<head>
  <title> Title of text XHTML Document </title>
</head>
<body>
<div class="myDiv">
  <h1> Heading of Page </h1>
  <p> here is a paragraph of text. I will include inside this paragraph
    a bunch of wonky text so that it looks fancy. </p>
  <p>Here is another paragraph with <em>inline emphasized</em>
    text, and <b> absolutely no</b> sense of humor. </p>
  <p>And another paragraph, this one with an  image, and a <br /> line break. </p>
</div>
</body></html>
```

Defining Blocks of Text

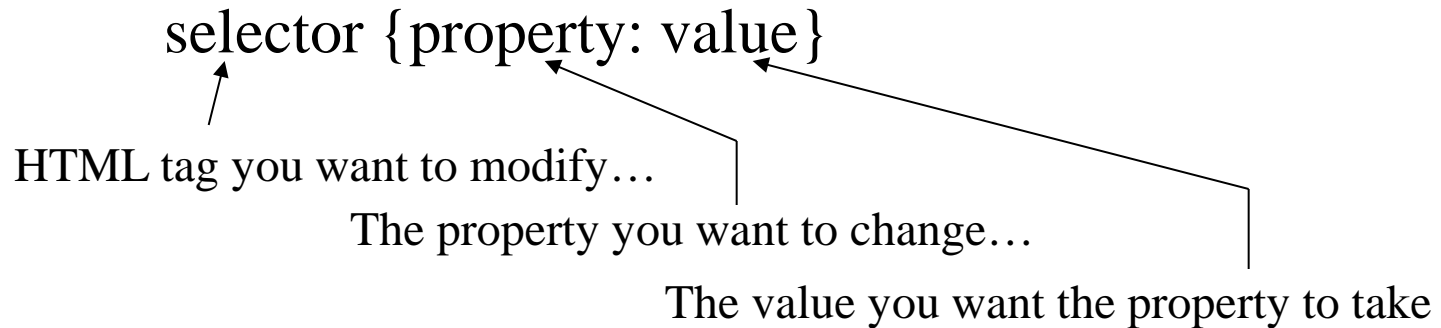
- `<div> ... </div>`
 - Named region
 - Implies a paragraph break,
 - Can include multiple paragraphs
- `<p> ... </p>`
 - Individual paragraph
- ` ... `
 - Any region
 - Does not create a paragraph break

Cascading Style Sheets (CSS)

- Separate content and structure from appearance
- Rules “cascade” from broad to narrow:
 - Browser default
 - External style sheet
 - Internal style sheet
 - Inline style

Basics of CSS

- Basic syntax:



- Example:

```
p { text-align: center;  
  color: black;  
  font-family: arial }
```

Causes

- Font to be center-aligned
- Font to be Arial and black

Different Ways of Using CSS

- Inline style:
 - Causes only this tag to have the desired properties

```
<p style="font-family:arial; color:blue">...</p>
```
- Internal stylesheet:
 - Causes *all* tags to have the desired properties

```
...  
<head>...  
<style type="text/css" >  
  p { font-family:arial; color:blue }  
</style>  
</head>  
<body>  
<p>...</p>  
...
```

Customizing Classes

- Ability to define customized styles for standard HTML tags:

```
...  
<head>...  
<style type="text/css">  
  p.style1 { font-family:arial; color:blue }  
  p.style2 { font-family:serif; color:red }  
</style>  
</head>  
<body>  
<p class="style1">...</p>  
<p class="style2">...</p>  
...
```

External Style Sheets

- Store formatting metadata in a separate file

mystyle.css

```
p.style1 { font-family:arial; color:blue }  
p.style2 { font-family:serif; color:red }
```

```
...  
<head>...  
<link rel="stylesheet" href="mystyle.css" type="text/css" />  
</head>  
<body>  
<p class="style1">...</p>  
<p class="style2">...</p>  
...
```

Programming for the Web

- JavaScript [Client-side]
 - Server embeds a program in HTML
 - Browser runs the program when it gets to it
- PHP “Common Gateway Interface” [Server-side]
 - HTML form sends field values to the server
 - Server passes field values to a program
 - Program generates a Web page as a response
- Ajax
 - Server sends browser a generic program to run
 - Browser and server programs exchange XML-encoded data

JavaScript

```
<HTML>
```

```
<HEAD>
```

```
  <TITLE>My first script</TITLE>
```

```
</HEAD>
```

```
<BODY BGCOLOR=WHITE>
```

```
<H1>
```

```
  <SCRIPT LANGUAGE=JAVASCRIPT TYPE="TEXT/JAVASCRIPT">
```

```
    document.write("Hello, world!")
```

```
  </SCRIPT>
```

```
</H1>
```

```
</BODY></HTML>
```

HTML Editors

- Several are available
 - Dreamweaver
 - Microsoft Word (File->”Edit with MS Word” in IE)
- You may still need to edit the HTML file
 - Some editors use browser-specific features
 - Some HTML features may be unavailable
 - File names may be butchered when you upload
- Verbose HTML can make hand-editing difficult

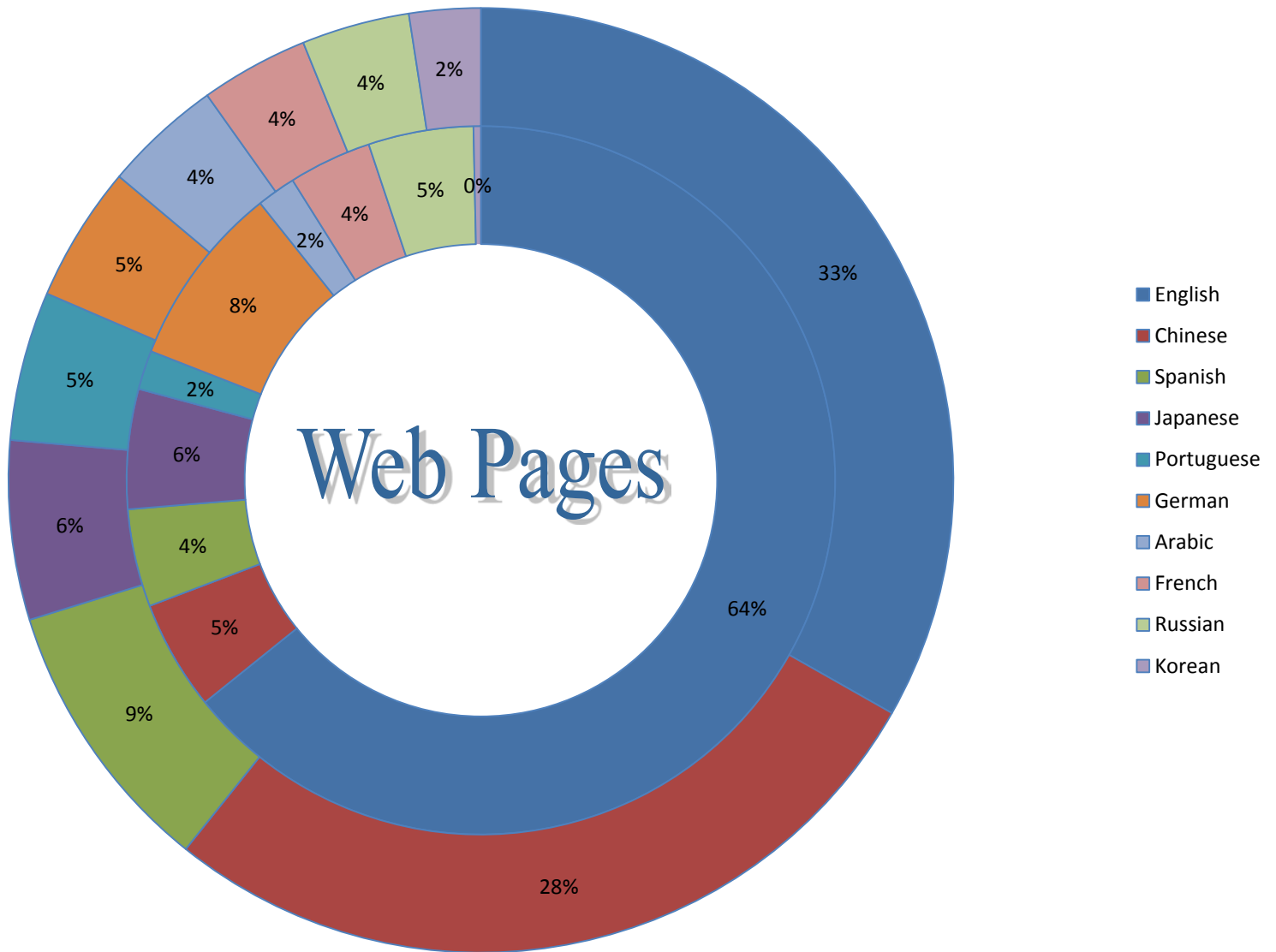
What is the Web?

- Protocols
 - HTTP, HTML, or URL?
- Perspective
 - Content or behavior?
- Content
 - Static, dynamic or streaming?
- Access
 - Public, protected, or internal?

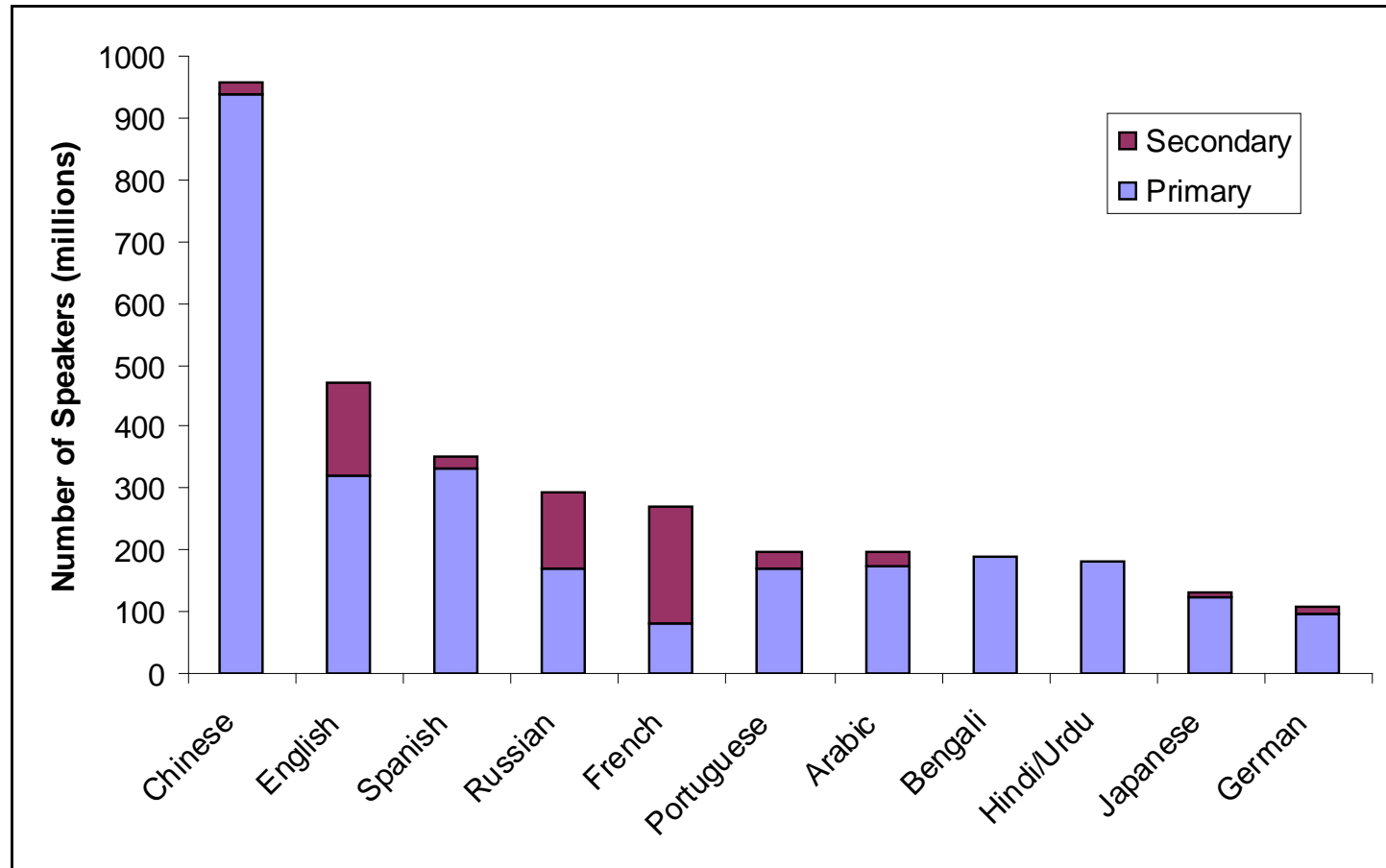
Why is there a Web?

- Affordable storage
 - 300,000 words/\$ in 1995
- Adequate backbone capacity
 - 25,000 simultaneous transfers in 1995
- Adequate “last mile” bandwidth
 - 1 second/screen in 1995
- Display capability
 - 10% of US population in 1995
- Effective search capabilities
 - Lycos and Yahoo were started in 1995

Global Internet Users

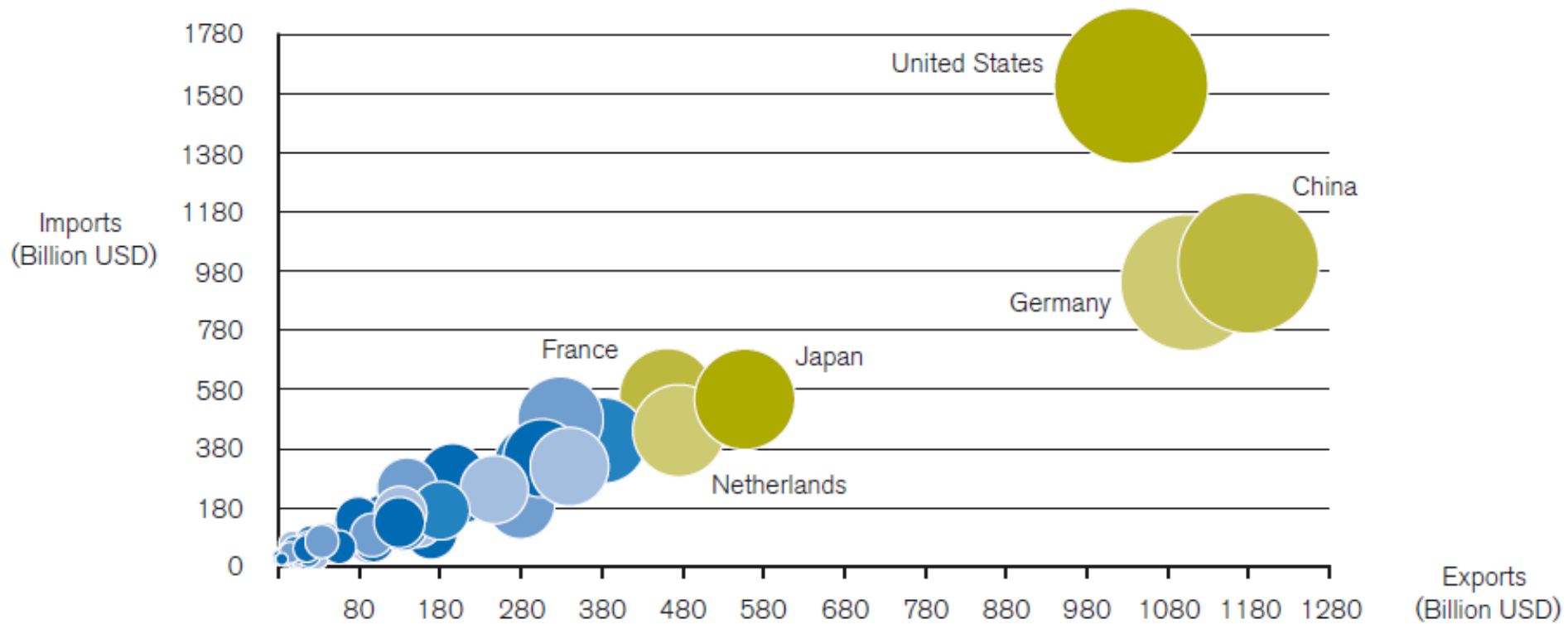


Most Widely-Spoken Languages



Global Trade

Leading economies of merchandise trade, 2009



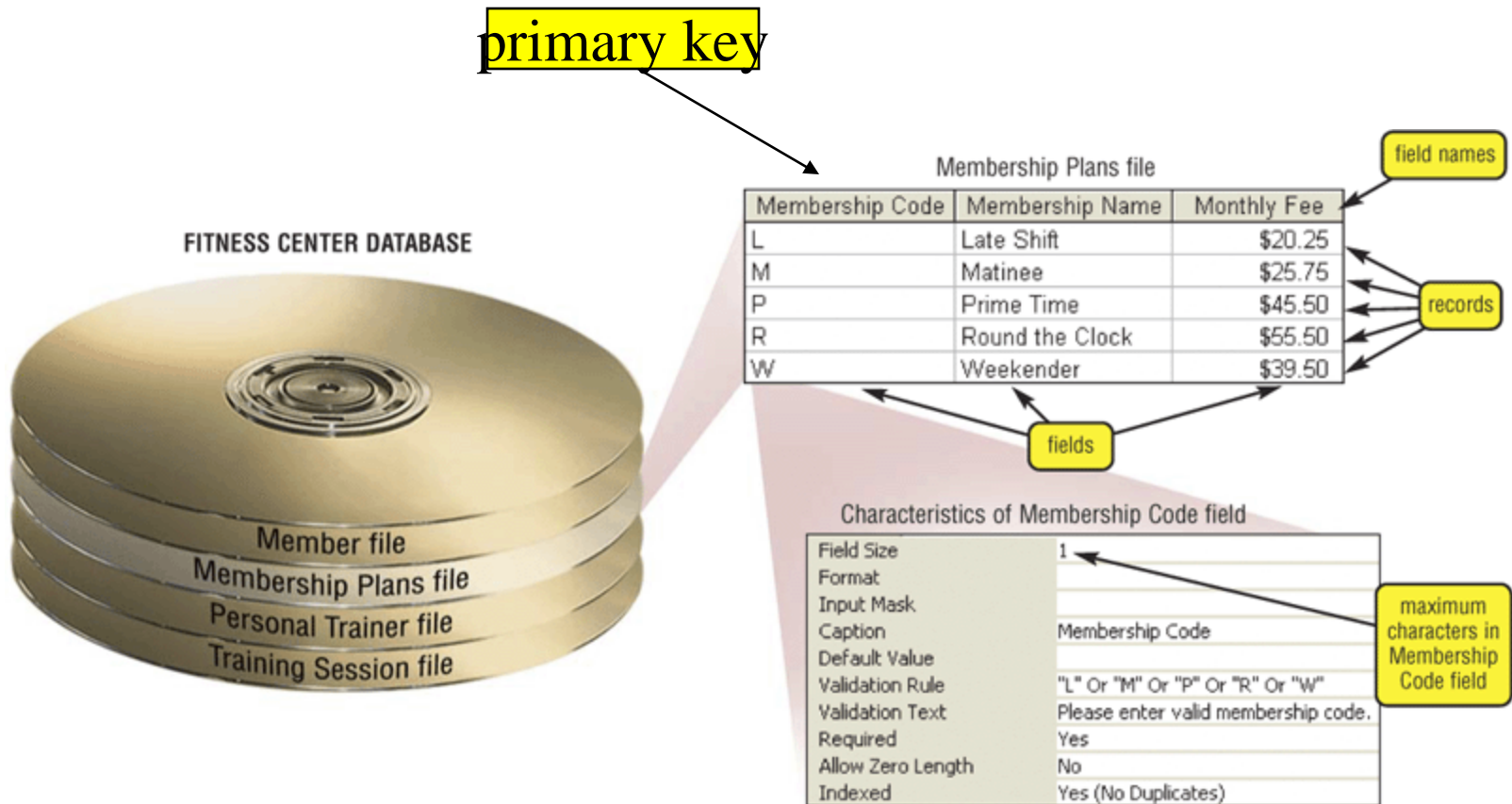
Databases

- Database
 - Collection of data, organized to support access
 - Models some aspects of reality
- DataBase Management System (DBMS)
 - Software to create and access databases
- Relational Algebra
 - Special-purpose programming language

Structured Information

- **Field** **An “atomic” unit of data**
 - number, string, true/false, ...
- **Record** **A collection of related fields**
- **Table** **A collection of related records**
 - Each record is one row in the table
 - Each field is one column in the table
- **Primary Key** **The field that identifies a record**
 - Values of a primary key must be unique
- **Database** **A collection of tables**

A Simple Example



Registrar Example

- Which students are in which courses?
- What do we need to know about the students?
 - first name, last name, email, department
- What do we need to know about the courses?
 - course ID, description, enrolled students, grades

A “Flat File” Solution

Student ID	Last Name	First Name	Department ID	Department	Course ID	Course description	Grades	email
1	Arrows	John	EE	EE	lpsc690	Information Technology	90	jarrows@wam
1	Arrows	John	EE	Elec Engin	ee750	Communication	95	ja_2002@yahoo
2	Peters	Kathy	HIST	HIST	lpsc690	Informatino Technology	95	kpeters2@wam
2	Peters	Kathy	HIST	history	hist405	American History	80	kpeters2@wma
3	Smith	Chris	HIST	history	hist405	American History	90	smith2002@glue
4	Smith	John	CLIS	Info Sci	lpsc690	Information Technology	98	js03@wam

Discussion Topic
Why is this a bad approach?

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

Relational Algebra

- Tables represent “relations”
 - Course, course description
 - Name, email address, department
- Named fields represent “attributes”
- Each row in the table is called a “tuple”
 - The order of the rows is not important
- Queries specify desired conditions
 - The DBMS then finds data that satisfies them

A Normalized Relational Database

Student Table

Student ID	Last Name	First Name	Department ID	email
1	Arrows	John	EE	jarrows@wam
2	Peters	Kathy	HIST	kpeters2@wam
3	Smith	Chris	HIST	smith2002@glue
4	Smith	John	CLIS	js03@wam

Department Table

Department ID	Department
EE	Electronic Engineering
HIST	History
CLIS	Information Studies

Course Table

Course ID	Course Description
lpsc690	Information Technology
ee750	Communication
hist405	American History

Enrollment Table

Student ID	Course ID	Grades
1	lpsc690	90
1	ee750	95
2	lpsc690	95
2	hist405	80
3	hist405	90
4	lpsc690	98

Approaches to Normalization

- For simple problems (like the homework)
 - Start with “binary relationships”
 - Pairs of fields that are related
 - Group together wherever possible
 - Add keys where necessary
- For more complicated problems
 - Entity relationship modeling (LBSC 670)

Example of Join

Student Table

Student ID	Last Name	First Name	Department ID	email
1	Arrows	John	EE	jarrows@wam
2	Peters	Kathy	HIST	kpeters2@wam
3	Smith	Chris	HIST	smith2002@glue
4	Smith	John	CLIS	js03@wam

Department Table

Department ID	Department
EE	Electronic Engineering
HIST	History
CLIS	Information Stuides

“Joined” Table

Student ID	Last Name	First Name	Department ID	Department	email
1	Arrows	John	EE	Electronic Engineering	jarrows@wam
2	Peters	Kathy	HIST	History	kpeters2@wam
3	Smith	Chris	HIST	History	smith2002@glue
4	Smith	John	CLIS	Information Stuides	js03@wam

Problems with Join

- Data modeling for join is complex
 - Useful to start with E-R modeling
- Join are expensive to compute
 - Both in time and storage space
- But it is joins that make databases relational
 - Projection and restriction also used in flat files

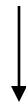
Some Lingo

- “Primary Key” uniquely identifies a record
 - e.g. student ID in the student table
- “Compound” primary key
 - Synthesize a primary key with a combination of fields
 - e.g., Student ID + Course ID in the enrollment table
- “Foreign Key” is primary key in the other table
 - Note: it need not be unique in this table

Project

New Table

Student ID	Last Name	First Name	Department ID	Department	email
1	Arrows	John	EE	Electronic Engineering	jarrows@wam
2	Peters	Kathy	HIST	History	kpeters2@wam
3	Smith	Chris	HIST	History	smith2002@glue
4	Smith	John	CLIS	Information Stuides	js03@wam



SELECT Student ID, Department

Student ID	Department
1	Electronic Engineering
2	History
3	History
4	Information Stuides

Restrict

New Table

Student ID	Last Name	First Name	Department ID	Department	email
1	Arrows	John	EE	Electronic Engineering	jarrows@wam
2	Peters	Kathy	HIST	History	kpeters2@wam
3	Smith	Chris	HIST	History	smith2002@glue
4	Smith	John	CLIS	Information Stuides	js03@wam

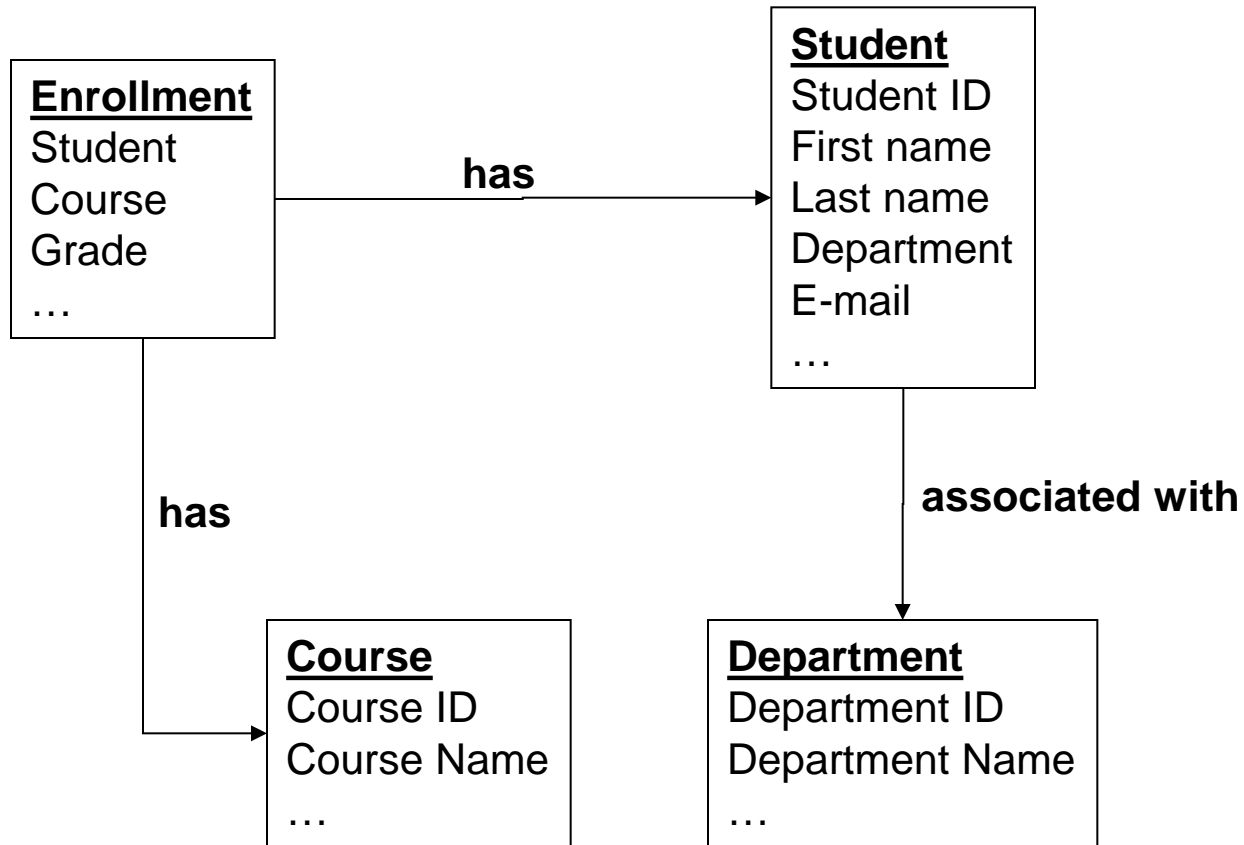
↓
WHERE Department ID = "HIST"

Student ID	Last Name	First Name	Department ID	Department	email
2	Peters	Kathy	HIST	History	kpeters2@wam
3	Smith	Chris	HIST	History	smith2002@glue

Entity-Relationship Diagrams

- Graphical visualization of the data model
- Entities are captured in boxes
- Relationships are captured using arrows

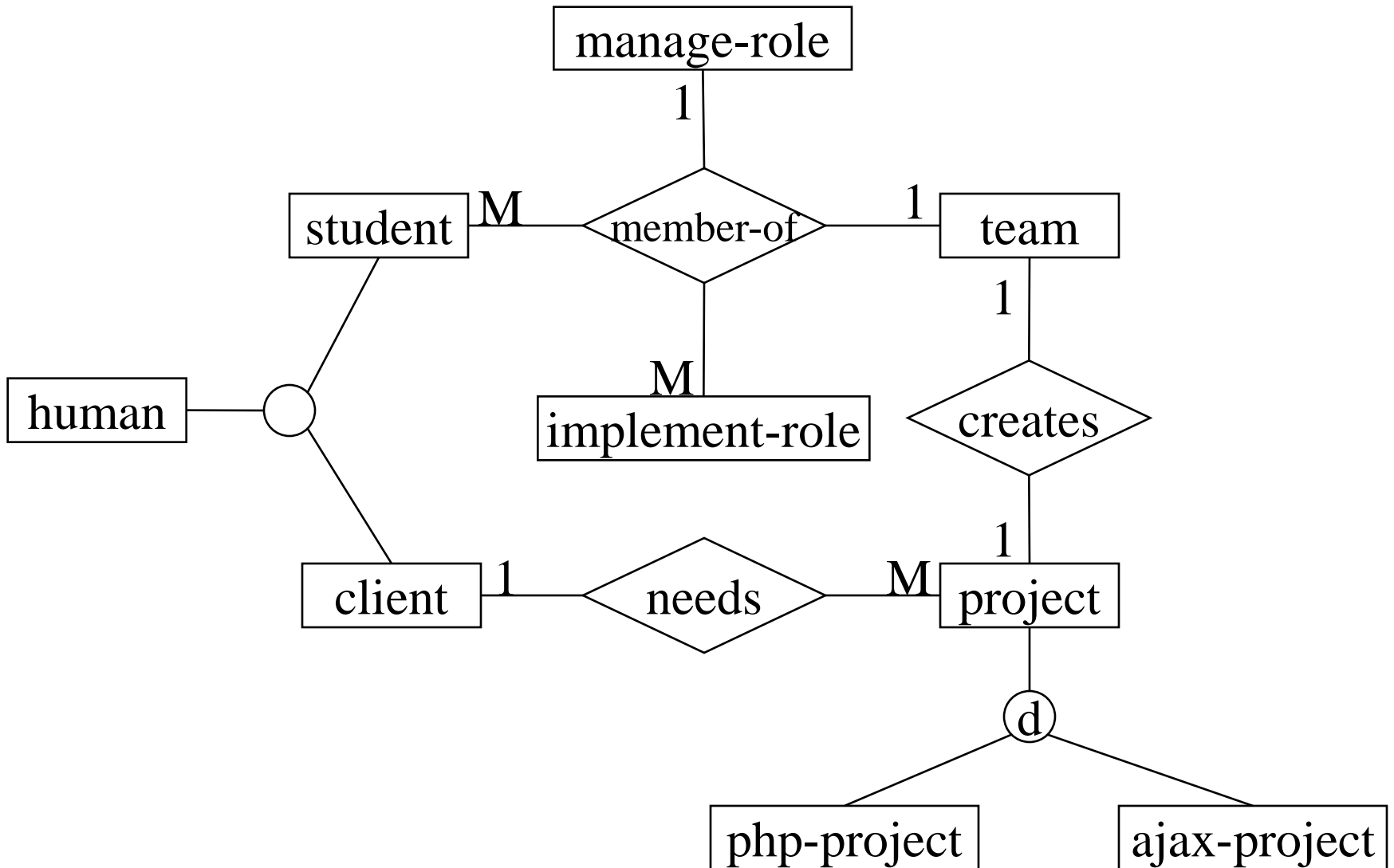
Registrar ER Diagram



Getting Started with E-R Modeling

- What **questions** must you answer?
- What **data** is needed to generate the answers?
 - Entities
 - Attributes of those entities
 - Relationships
 - Nature of those relationships
- How will the user interact with the system?
 - Relating the question to the available data
 - Expressing the answer in a useful form

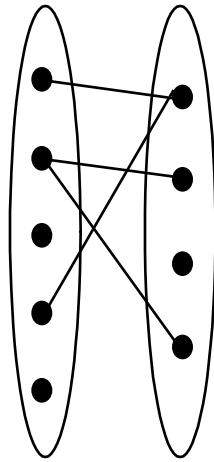
“Project Team” E-R Example



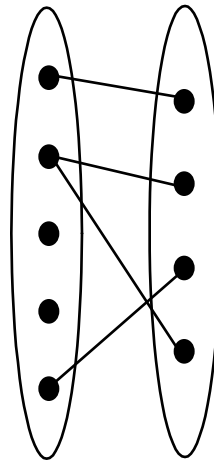
Components of E-R Diagrams

- Entities
 - Types
 - Subtypes (disjoint / overlapping)
 - Attributes
 - Mandatory / optional
 - Identifier
- Relationships
 - Cardinality
 - Existence
 - Degree

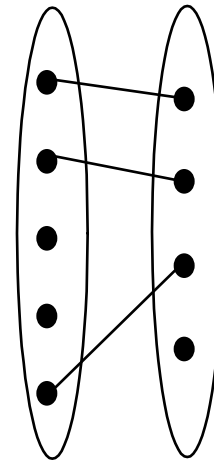
Types of Relationships



Many-to-Many



1-to-Many



1-to-1

Making Tables from E-R Diagrams

- Pick a primary key for each entity
- Build the tables
 - One per entity
 - Plus one per M:M relationship
 - Choose terse but memorable table and field names
- Check for parsimonious representation
 - Relational “normalization”
 - Redundant storage of computable values
- Implement using a DBMS

Normalization

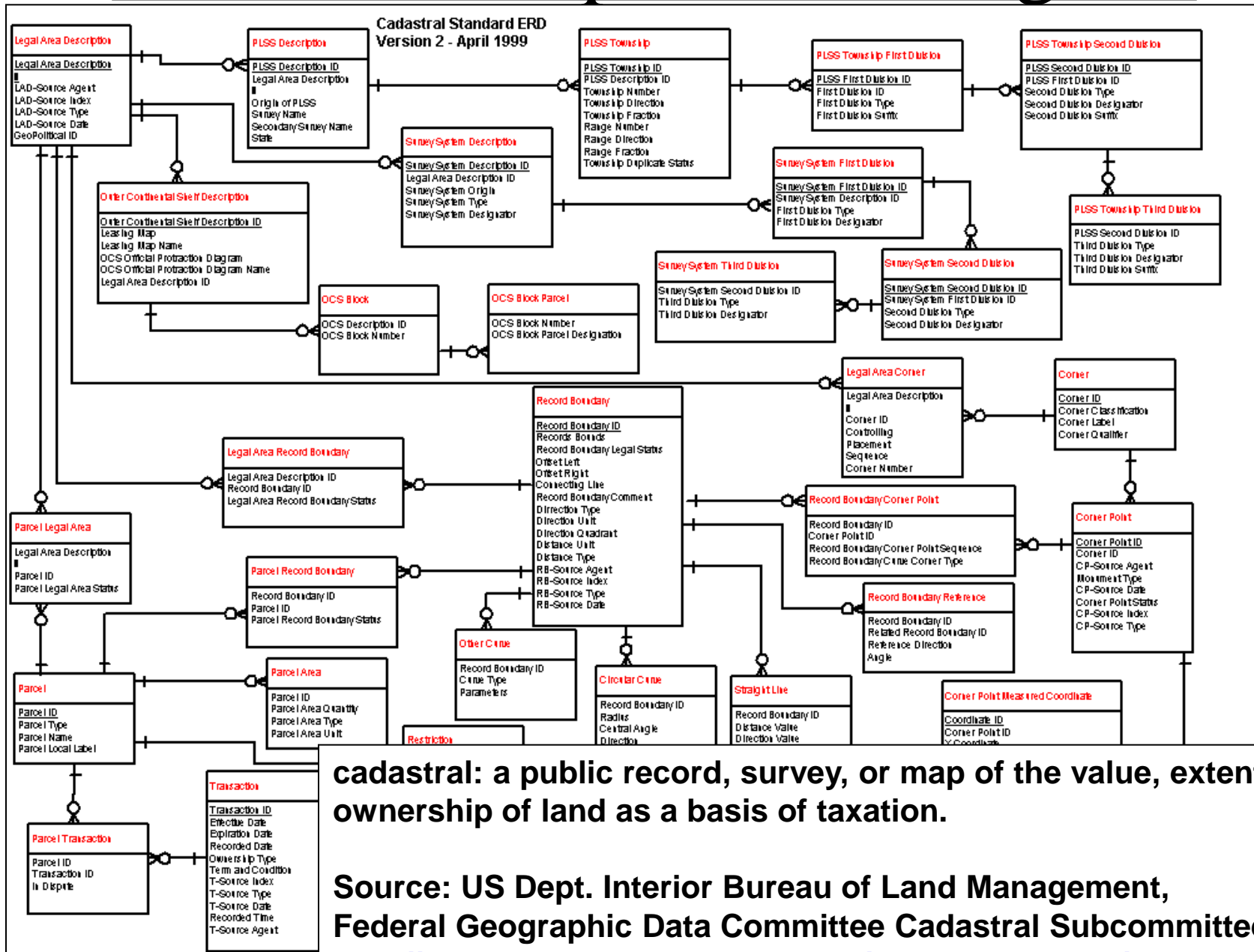
- 1NF: Single-valued indivisible (atomic) attributes
 - Split “Doug Oard” to two attributes as (“Doug”, “Oard”)
 - Model M:M implement-role relationship with a table
- 2NF: Attributes depend on complete primary key
 - (id, impl-role, name) \rightarrow (id, name) + (id, impl-role)
- 3NF: Attributes depend directly on primary key
 - (id, addr, city, state, zip) \rightarrow (id, addr, zip) + (zip, city, state)

- 4NF: Divide independent M:M tables
 - (id, role, courses) \rightarrow (id, role) + (id, courses)
- 5NF: Don't enumerate derivable combinations

Normalized Table Structure

- Persons: id, fname, lname, userid, password
- Contacts: id, ctype, cstring
- Ctlabels: ctype, string
- Students: id, team, mrole
- Iroles: id, irole
- Rlabels: role, string
- Projects: team, client, pstring

A More Complex ER Diagram



cadastral: a public record, survey, or map of the value, extent, and ownership of land as a basis of taxation.

**Source: US Dept. Interior Bureau of Land Management,
Federal Geographic Data Committee Cadastral Subcommittee**

<http://www.fairview-industries.com/standardmodule/cad-erd.htm>

Key Ideas

- Databases are a good choice when you have
 - Lots of data
 - A problem that contains inherent relationships
- Design before you implement
 - This is just another type of programming
 - The mythical person-month applies!
- Join is the most important concept
 - Project and restrict just remove undesired stuff

Before You Go

On a sheet of paper, answer the following (ungraded) question (no names, please):

What was the muddiest point in today's class?