Some Logistics

• If you are auditing, please talk to us
• If you missed the first class, please talk to us

• Group formation and project selection is due a week from today, email us with:
  – Group members
  – Project selection

• Once a project is selected, please schedule a meeting with us during the office hour

• I will be away Feb 6th to Feb 20th
Today’s Outline

• **Introduction**
  – A Primer on Database Topics: from SQL to XML
  – A Primer on Web Topics: from Hypertext to Semantic Web

• **Semi-structured Search**
  – FlexPath
  – Timber
Database 101

• **What is a database (in the traditional sense)?**
  – A *large* collection of *structured* data

• **Structured:**
  – Data are stored according to the specified schema (i.e., tables)

<table>
<thead>
<tr>
<th>Student</th>
<th>Course</th>
<th>Semester</th>
<th>Grade</th>
</tr>
</thead>
<tbody>
<tr>
<td>John Doe</td>
<td>CS 6093</td>
<td>Spring 2011</td>
<td>A</td>
</tr>
<tr>
<td>Jane Smith</td>
<td>CS 6093</td>
<td>Spring 2011</td>
<td>A</td>
</tr>
</tbody>
</table>

• **Large:**
  – Modern databases contain thousands of tables and billions of records
Advanced Database Topics

• A crash course on topics that we don’t cover
  – Conceptual Model and Physical Storage Separation
  – Relational Data Model and Data Normalization
  – Relational Algebra and SQL
  – Storage and Indexing
  – Query Optimization
  – Transaction and Concurrency Control

• Topics that we will cover in this course, you will learn in details over the semester
  – Semi-structured Data and Usability
  – Data Warehousing and Mining
Abstraction: Logical & Physical

• Imagine a world without database, and you have to manage lots of data, what would you do?
  – Files: lots of them and/or very large ones
  – Problem:
    • To manipulate any piece of information, you have to write lots of code to search through the files and manage the intermediate results

• Database provides the conceptual and physical data independence
  – How you access the data is abstracted away from how the data is stored

<table>
<thead>
<tr>
<th>Student</th>
<th>Course</th>
<th>Semester</th>
<th>Grade</th>
</tr>
</thead>
<tbody>
<tr>
<td>John D</td>
<td>6093</td>
<td>S11</td>
<td>A</td>
</tr>
<tr>
<td>Jane S</td>
<td>6093</td>
<td>S11</td>
<td>A</td>
</tr>
</tbody>
</table>

File1:
“John D”, “6093”, “S11”, “A”
“Jane S”, “6093”, “S11”, “A”

File2:
“John D”, “Jane S”
“6093”, “6093”
“S11”, “S11”
“A”, “A”
Data Model

• How to represent real world information?
• Entity-Relationship (ER) model
  – Each conceptual entity is maintained using a **table**
  – 1-1 relationships are maintained using **key/foreign key**
  – 1-many and many-many relationships are maintained through tables

• **Data normalization**
  – Avoid update anomaly

<table>
<thead>
<tr>
<th>Student</th>
<th>CID</th>
<th>Grade</th>
</tr>
</thead>
<tbody>
<tr>
<td>John D</td>
<td>1</td>
<td>A</td>
</tr>
<tr>
<td>Jane S</td>
<td>1</td>
<td>A</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ID</th>
<th>Course</th>
<th>Semester</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>6093</td>
<td>S11</td>
</tr>
</tbody>
</table>
SQL: Structured Query Language

- **Key feature**
  - Declarative
    - Specifying what data one wants, without specifying how to get them

- **Example**

```sql
select s.student
from student s, course c
where s.grade = "A" and s.cid = c.id and c.semester = "S11"
```

<table>
<thead>
<tr>
<th>Student</th>
<th>CID</th>
<th>Grade</th>
</tr>
</thead>
<tbody>
<tr>
<td>John D</td>
<td>1</td>
<td>A</td>
</tr>
<tr>
<td>Jane S</td>
<td>1</td>
<td>A</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ID</th>
<th>Course</th>
<th>Semester</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>6093</td>
<td>S11</td>
</tr>
</tbody>
</table>

Selection, Project, Join (SPJ)
Indexing and Access Methods

• One of the most well studied core database system topics
  
select s.student
from student s, course c
where s.grade = “A” and s.cid = c.id and c.semester = “S11”

• Auxiliary data structure that can speed up record lookup
  – B-tree, R-tree
  – Hash

P1 1 A

P2 1 B

P3 1 B

…

Pn 1 A
Indexing, part 2

- **Clustered vs Unclustered**
  - Large impact due to the difference in sequential and random I/O
Query Optimization

- **Query plan**
  
  ```
  select s.student
  from student s, course c
  where s.grade = "A" and
    s.cid = c.id and
    c.semester = "S11"
  ```

- **Each SQL query can be written into many plans**

- **Cost model determines which plan is more efficient**
  - Scan
  - Index access
  - Join
Transaction and Concurrency Control

• A transaction is a series of database operations involving updates
  – Critically important in shopping/financial databases

• Atomicity: all or none
  Consistency: must leave the database in a consistent state
  Isolation: isolated from other transaction
  Durability: once committed, must be reflected in database

• Atomicity and durability are ensured through the use of a write log
  – If a transaction is aborted, log is used to undo the writes
  – If database crashes, log is used to redo the writes
Transaction, part 2

- Ensuring isolation is more challenging
- WR conflict
  - Reading uncommitted data
- RW conflict
  - Unrepeatable read
- WW conflict
  - Overwriting uncommitted data
- Strict Two Phase Lock (2PL)
  - Read lock can be shared
  - Write lock is exclusive
  - Transactions grab locks as they read and write data items, and release all locks after commit
Additional Advanced Database Topics

• Triggers
• Streaming
• Spatial
• Column database
• Privacy
• Distributed and parallel database
• And lots more …
Q: why isn’t there more databases?

- Setting up a database is non-trivial
- Evolving a database is hard
- All because of strict enforcement of structure!

<table>
<thead>
<tr>
<th>ID</th>
<th>Course</th>
<th>Semester</th>
</tr>
</thead>
<tbody>
<tr>
<td>John D</td>
<td>1</td>
<td>A</td>
</tr>
<tr>
<td>Jane S</td>
<td>1</td>
<td>A</td>
</tr>
</tbody>
</table>

| 1       | 6093    | S11      |
XML (eXtensible Markup Language)

- Designed to address this separation between schema and data (invented in 1996)
- Let the data be self-describable, i.e., semi-structured

```xml
<student>
  <name>John D</name>
  <course>
    <title>6093</title>
    <semester>S11</semester>
  </course>
  <grade>A</grade>
</student>
```
What is XML?

• **Documents that consist of elements indicated by begin & end tags**
  – Can be nested but cannot interleave each other
  • Can have arbitrary number of sub-elements or be empty

```xml
<chap title = "Introduction To XML">
  some free text
  <sect title = "What is XML?">
    <sect title = "Elements"> ... </sect>
  </sect>
  <sect title = "Why XML?"> ... </sect>
  ... possibly more free text
</chap>
```
Extensibility

• Users can easily define their own XML elements
• It contains very few rules and constraints
  – The only requirement is a document should be well formed (i.e., no interleaving elements and a begin tag must be matched with an end tag)
• This means that XML is a very general, multi-purpose format language
  – A strict version of HTML (called XHTML) is in fact a specialized subset of XML
Validity

• It is possible to associate XML documents with a schema (Document Type Definition or XML Schema), but not necessary
  – Typically used for applications where the schema can be agreed upon
  – For example, HTML documents have a schema that is understood by modern browsers

• An XML document is valid if it is well-formed and complies with its schema
XML Data versus Relational Data

• XML data is much easier to understand
• XML data is much larger to store

• XML has become the standard data exchange format
  – Applications use XML to exchange data with each other, but store them in compressed or relational format
  – RSS, SOAP, WSDL, JSON (inspired by XML), etc.

• Blobs of free text can be easily incorporated into XML data, therefore typical XML data is often a mixture of free texts and small doses of structures (hence the term “semi-structured”)

20 CS 6093 ©2011
XPath and XQuery

• **XPath**
  – Language for specifying navigation within an XML document.
  – An XPath expression’s syntax:
    • Path ::= /step1/step2/…
    • E.g., /films/film/writers/writer/Murray Burnett

• **XQuery**
  – A more complex language based on XPath.
  – Has all the functionalities of XPath.
  – Includes aggregation and join operations
Today’s Outline

• Introduction
  – A Primer on Database Topics: from SQL to XML
  – A Primer on Web Topics: from Hypertext to Semantic Web

• Semi-structured Search
  – FlexPath
  – Timber
The Invention of the Web

• Tim Berners-Lee
  – In 1990, implemented the first HTTP (HyperText Transfer Protocol) client (i.e., browser) and server via the Internet (TCP/IP)
  – info.cern.ch was the first-ever web site
  – He did not patent it!

• HyperText documents, a simple idea
  – documents with links to each other

• Client/Browser
  – Issues requests to the server for documents based on a URL
  – Renders documents received from server

• Server
  – Resolves document requests based on URLs
  – Send requested document back to the client
Documents are at the center of the Web

- Free text documents contributed greatly to the success of the Web
  - Easy to create, no schema to learn and conform to

- Initially, the Web is small enough so that a few people can categorize the whole Web
  - Yahoo! (Yet Another Hierarchical Officious Oracle)

- Slowly, the Web becomes large enough that browsing is no longer a valid option
  - Search engine becomes indispensible part of the Web
  - Before Google, there are AltaVista, Infoseek, Lycos, Ask Jeeves, etc.
Information Retrieval Queries

• Given a set of keywords, find the top most relevant documents to those keywords
  – The core technology behind search engine

• IR queries are very different from SQL queries

• Fernando will cover many aspects of IR in the upcoming lectures
Will Document and IR Queries still Dominate the Web of the Future?

• It’s an open question
• Limitations with document and IR queries?
  – Users with a complex task often have to search for relevant documents first, and then assemble information from those documents by themselves
  – E.g., Compare recent models of Nikon and Canon cameras
  – E.g., Find all Chinese restaurants in Brooklyn appeared in the Gourmet magazine

• And?
Semantic Web

• Sir Berners-Lee is a big proponent of Semantic Web
  – His article in Scientific American 2001 (first assignment) is a call to arms to researchers and programmers around the world
  – 10 years later, we are not there yet …

• Web 3.0 could be the Semantic Web 😊
What is Semantic Web

“The Semantic Web is an extension of the current web in which information is given well-defined meaning, better enabling computers and people to work in cooperation.”


• Semantic Web is a Web for the machines to understand, reason, and act
  – “book an appointment with Dr. House Tomorrow afternoon 3pm”
  – “send my medical records to Dr. House’s database once the appointment is confirmed”
  – “fill those prescriptions based on Dr. House’s instruction”
  – …
Main Components of Semantic Web

- XML
- RDF
- DAML+OIL, OWL
RDF: Resource Description Framework

• **Definitions:**
  - *Resource:* The resources being described by RDF are anything that can be named via a URI.
  - *Property:* A property is also a resource that has a name, for instance *Author* or *Title*.
  - *Statement:* A statement consists of the combination of a Resource, a Property, and an associated value.

*Alice* is the *creator* of the resource [http://www.cs.indiana.edu/~Alice](http://www.cs.indiana.edu/~Alice).

```
Alice = http://purl.org/dc/elements/1.1/creator
```

[Diagram of RDF concepts with labeled nodes and connections]
DAML+OIL and OWL

- DAML+OIL: DARPA Agent Markup Language & Ontology Inference Layer
- OWL: Web Ontology Language

- Both are extensions of RDF to better capture ontology with more expressive powers
  - such as negation, transitivity, etc.
A Teaser for Information Extraction

• **Semantic Web presents a grand vision, but faces a chicken-and-egg problem**
  – Until there are enough semantic/structured information on the Web, no serious application will be built to take advantage of Semantic Web
  – But without killer applications, there is no incentive for users to produce semantic/structured information

• **Information extraction to the rescue:**
  – IE techniques are expected to be used for bootstrapping Semantic Web
  – IE techniques can leverage small amount of semantic data and snowball them to more semantic data
10 min Break
Today’s Outline

• Introduction
  – A Primer on Database Topics: from SQL to XML
  – A Primer on Web Topics: from Hypertext to Semantic Web

• Semi-structured Search
  – FlexPath
  – Timber
An Emerging Query Paradigm

• XML data requires a different query paradigm from SQL query
  – Data heterogeneity
  – Free texts

• Semantic Web requires a different search paradigm from IR queries
  – Incorporating structural constraints

• Both point to the middle ground of “structured search”
  – A query paradigm that combines the flexibility of keywords searches and the additional control of structured constraints

• There are many studies, we look at one of them
  – FlexPath: Flexible Structure and Full-Text Querying for XML, SIGMOD 2004
XML Data can be Represented as Trees

<books>
  <book>
    <info>
      <author>Dickens</author>
      <title>Great Expectation</title>
    </info>
    <edition>paperback</edition>
  </book>
</books>
Tree Pattern Queries

- TPQ = (T, F, n)
  - T: a rooted tree structure, with PC and AD edges
  - F: Boolean combination of value predicates, including the FTExp expression
  - n ∈ T: the distinguished node that is the return unit

Q1:
$1.tag = \text{book} \&
$2.tag = \text{info} \&
$3.tag = \text{author} \&
$4.tag = \text{title} \&
\text{contains}(\text{title}, \text{"expectation"})
Tree Pattern Query – Logical Expression

- Representing the tree as a set of logical expressions
  - $pc(x, y)$ to denote parent/child edge
  - $ad(x, y)$ to denote ancestor/descendant edge

- Example
  - $pc(1, 2) \land pc(2, 3) \land pc(2, 3) \land \ldots$

```
<table>
<thead>
<tr>
<th>$1$</th>
<th>$2$</th>
<th>$3$</th>
<th>$4$</th>
</tr>
</thead>
</table>
```

$book$

$info$

$edition$

“paperback”

$author$

“Dickens”

$title$

“Great Expectations”
Challenge

• Structure of the XML data is very flexible (design goal!)
  – The exact format may be unknown at query time
  – Multiple formats may exist in the data

• Query relaxation!

```xml
document1:
  book
    info
      author
        "Dickens"
      title
        "Great Expectations"
      edition
        "paperback"

document2:
  book
    info
      author
        "Dickens"
      edition
        "paperback"
      title
        "Great Expectations"
```
Query Relaxation – Initial Ideas

- **Adding disjunction**
  - \( Q_1 \vee Q_2 \ldots \)
  - The returned results can be *arbitrarily* different from the original query

- **Relaxing contains operator**
  - Application and domain *specific*, cannot be part of a general approach

- **Dropping predicates**
  - We are getting there …
  - Need to be careful what kind of predicates we can drop
    - Need to maintain the return unit
    - Need to retain the contains condition
    - The resulting query should still be in TPQ
Inference Rules

- **PC edges can be relaxed to AD edges**
  - \( pc(x, y) \rightarrow ad(x, y) \)

- **Two connected AD edges can be combined**
  - \( ad(x, y) \land ad(y, z) \rightarrow ad(x, z) \)

- **The contains operator can be expanded to the ancestor node**
  - \( ad(x, y) \land contains(y, FTExp) \rightarrow contains(x, FTExp) \)
Query Closure and Core

- Given a TPQ, its closure can be computed by repeated applying inference rules until no more applies.

- Given a TPQ, its core can be computed by removing any predicate that can be derived from the remaining predicates based on the inference rules.
Query Closure – Example

• Structural relaxations
  – ad($1, $2), ad($2, $3), ad($2, $4)
  – ad($1, $3)
  – ad($1, $4)

• Content relaxations
  – contains($2, “expectation”)
  – contains($1, “expectation”)

Q1:
$1.tag = book &
$2.tag = info &
$3.tag = author &
$4.tag = title &
contains($4, “expectation”)
Query Relaxation

- Given a TPQ $Q$, compute its closure $Q_c$
- A valid relaxed TPQ $Q_r$ can be obtained by dropping one or more predicates from $Q_c$ as long as
  - The tree structure is maintained
  - The return unit is retained

$Q_1$: $1.tag = \text{book} \land 2.tag = \text{info} \land 3.tag = \text{author} \land 4.tag = \text{title} \land \text{contains}(4, \text{“expectation”}) \land \text{contains}(2, \text{“expectation”}) \land \text{contains}(1, \text{“expectation”})$

$Q_r$: $1.tag = \text{book} \land 2.tag = \text{info} \land 3.tag = \text{author} \land \text{contains}(2, \text{“expectation”})$
Question: what’s the most relaxed query for this example?

$Q_f:\quad \text{$1.tag = book \&
\text{contains($1, \text{"expectation"})$}$}
Principles in Ranking Relaxed TPQs

• **Top-k query processing**
  – Adopted from Information Retrieval, different from traditional database query processing
  – More suitable for flexible and heterogeneous data

• **Structural score ordering**
  – If Q1 is a relaxation of Q2, each answer returned by Q2 should have high structural score than any answer returned by Q1 but not Q2.

• **Order invariance**
  – The order in which the predicates are dropped should have no impact on the scores of the answers
Ranking Schemes

• **Penalty associated with dropping predicates**
  – Idea: the broader the context is relaxed, the higher the penalty

• **Integrated with the score from the evaluation of FTExp**
  – Leveraging IR style notion (see next)
Adaptation of *tf.idf* to XML Queries

<table>
<thead>
<tr>
<th>Document Collection (Information Retrieval)</th>
<th>XML Document (Structured Search)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Document</td>
<td>XML Node (result is a subtree rooted at a distinguished node, i.e., a node with a given label and structural properties)</td>
</tr>
<tr>
<td>Keyword(s)</td>
<td>Query Pattern</td>
</tr>
<tr>
<td><em>idf</em> (<em>inverse document frequency</em>) is a function of the fraction of documents that contain the keyword(s)</td>
<td><em>idf</em> is a function of the fraction of distinguished nodes that match the query pattern</td>
</tr>
<tr>
<td><em>tf</em> (<em>term frequency</em>) is a function of the number of occurrences of the keyword in the document</td>
<td><em>tf</em> is a function of the number of ways the query pattern matches the distinguished node</td>
</tr>
</tbody>
</table>
Algorithms

• **DPO**
  - Starts with evaluating the original query
  - If the returned results is fewer than needed, drop the predicate with smallest penalty and re-evaluate
  - Iterate until enough results are obtained
  - Key: predicates are dropped one at one iteration

• **SSO**
  - Similar to DPO, but uses selectivity estimation to estimate the result size of each query
  - Keeping dropping predicates until the estimated size is large enough
  - Key: multiple predicates can be dropped at one iteration

• **Hybrid**
Experiments

- Data set: XMark
- Query set: three benchmark queries
- Parameters:
  - $k$
  - document size
  - number of relaxation allowed
  - query complexity
    - number of query nodes
    - number of FTExp
  - document complexity
    - depth and fan-out
  - selectivity estimation accuracy
Example Experimental Result

Figure 15: Varying K (DocSize = 10MB)
Today’s Outline

• Introduction
  – A Primer on Database Topics: from SQL to XML
  – A Primer on Web Topics: from Hypertext to Semantic Web

• Semi-structured Search
  – FlexPath
  – Timber
TIMBER

- Tree-structured native XML database Implemented at the university of Michigan by Bright and Energetic Researchers

- A native XML database
  - Native: store XML data directly on the disk instead of shredding it into relational representation

- Core contributions
  - Tree Algebra for XML
  - A core set of XML query evaluation algorithms
    - Structural joins
    - GroupBy
  - XML query optimization techniques
    - Join order selection
    - Result size estimation
Fig. 2. TIMBER Architecture overview
Data Manager

- Stores XML data in its native format
  - Interval encoding
  - element → startkey: (startkey, endkey, level, attributes, value)

- Uses an object-oriented data store to physically store the data as key/object pairs
Query Evaluator

• Core access methods
  – Structural joins: parent/child and ancestor/descendant joins
  – Similar to traditional database, where join on key/foreign key is critical

• Interval encoding leads to the following observations
  – node1 = (s1, e1, l1)
  – node2 = (s2, e2, l2)
  – pc(node1, node2) = true iff (s1 < s2) & (e1 > e2) & (l1 = l2 – 1)
  – ad(node1, node2) = true iff (s1 < s2) & (e1 > e2) & (l1 < l2)

• Stack based algorithms lead to efficient implementation of structural joins

• Foundation for structured querying mechanisms
Structured Searching in Timber

- An active direction and essential component since 2003


- Cong Yu and H. V. Jagadish. Querying Complex Structured Databases. In VLDB, Vienna, Austria. 2007.
Impact of Timber

• First to introduce many novel native XML query evaluation and optimization techniques
  – And the Tree Algebra for XML (TAX) has become the standard logical algebra for XML

• Serving as the underlying system for many novel semi-structured database applications
  – Biology/Bioinformatics application
  – Database usability applications

• And, graduated a few students 😊
What We Learned Today

- A 10,000 feet overview of advanced database topics
- A structure and content hybrid query paradigm is emerging from both the database query perspective and the Web search perspective
- A detailed look at one of those attempts
- A brief overview of a system that underlies such attempts
Questions?