ICS 101 Fall 2012

Introduction to Data Management

Asst. Prof. Lipyeow Lim
Information & Computer Science Department
University of Hawaii at Manoa
The Data Management Problem

Where is the photo I took last Christmas?
Where did I read about “Turing Machines”?
Where is the invoice for this computer?
Which product is the most profitable?
What is `data’? 

- **Data** are known facts that can be recorded and that have implicit meaning.
- Three broad categories of data
  - Structured data
  - Semi-structured data
  - Unstructured data
- `Structure’ of data refers to the organization within the data that is identifiable.
What is a database?

- A **database**: a collection of related data.
  - Represents some aspect of the real world (aka universe of discourse).
  - Logically coherent collection of data
  - Designed and built for specific purpose
- A **data model** is a collection of concepts for describing/organizing the data.
- A **schema** is a description of a particular collection of data, using the a given data model.
The Relational Data Model

• *Relational database*: a set of *relations*

• A *relation* is made up of 2 parts:
  
  – *Instance*: a *table*, with rows and columns. 
    #Rows = *cardinality*, #fields = *degree / arity*.
  
  – *Schema*: specifies name of relation, plus name and *domain/type* of each column or attribute.
    • E.G. Students(sid: string, name: string, login: string, age: integer, gpa: real).

• Can think of a relation as a *set* of rows or *tuples* (i.e., all rows are distinct).
Example Instance of Students Relation

<table>
<thead>
<tr>
<th>sid</th>
<th>name</th>
<th>login</th>
<th>age</th>
<th>gpa</th>
</tr>
</thead>
<tbody>
<tr>
<td>53666</td>
<td>Jones</td>
<td>jones@cs</td>
<td>18</td>
<td>3.4</td>
</tr>
<tr>
<td>53688</td>
<td>Smith</td>
<td>smith@eecs</td>
<td>18</td>
<td>3.2</td>
</tr>
<tr>
<td>53650</td>
<td>Smith</td>
<td>smith@math</td>
<td>19</td>
<td>3.8</td>
</tr>
</tbody>
</table>

- Cardinality = 3, degree=5, all rows distinct
Why is the relational model useful?

• Supports simple and powerful query capabilities!
• Structured Query Language (SQL)

```
SELECT S.sname
FROM Students S
WHERE S.gpa > 3.5
```

<table>
<thead>
<tr>
<th>sid</th>
<th>name</th>
<th>login</th>
<th>age</th>
<th>gpa</th>
</tr>
</thead>
<tbody>
<tr>
<td>53666</td>
<td>Jones</td>
<td>jones@cs</td>
<td>18</td>
<td>3.4</td>
</tr>
<tr>
<td>53688</td>
<td>Smith</td>
<td>smith@eecs</td>
<td>18</td>
<td>3.2</td>
</tr>
<tr>
<td>53650</td>
<td>Smith</td>
<td>smith@math</td>
<td>19</td>
<td>3.8</td>
</tr>
</tbody>
</table>
What is a DBMS?

• A **database management system (DBMS)** is a *collection of programs* that enables users to
  
  – **Create** new DBs and specify the structure using data definition language (DDL)
  
  – **Query** data using a query language or data manipulation language (DML)
  
  – **Store** very large amounts of data
  
  – **Support** durability in the face of failures, errors, misuse
  
  – **Control** concurrent access to data from many users
Types of Databases

- On-line Transaction Processing (OLTP)
  - Banking
  - Airline reservations
  - Corporate records

- On-line Analytical Processing (OLAP)
  - Data warehouses, data marts
  - Business intelligence (BI)

- Specialized databases
  - Multimedia

- XML
- Geographical Information Systems (GIS)
- Real-time databases (telecom industry)

- Special Applications
  - Customer Relationship Management (CRM)
  - Enterprise Resource Planning (ERP)

- Hosted DB Services
  - Amazon, Salesforce
A Bit of History

- 1970 Edgar F Codd (aka “Ted”) invented the relational model in the seminal paper “A Relational Model of Data for Large Shared Data Banks”
  - Main concept: relation = a table with rows and columns.
  - Every relation has a schema, which describes the columns.
- Prior 1970, no standard data model.
  - Network model used by Codasyl
  - Hierarchical model used by IMS
- After 1970, IBM built System R as proof-of-concept for relational model and used SQL as the query language. SQL eventually became a standard.
Why use a DBMS?

- Large datasets
- Concurrency/ multi-user
- Crash recovery
- Declarative query language
- No need to figure out what low level data structure
- Data independence and efficient access.
- Reduced application development time.
- Data integrity and security.
- Uniform data administration.
Transactions

• A *transaction* is the DBMS’s abstract view of a user program: a sequence of reads and writes.
  – Eg. User 1 views available seats and reserves seat 22A.

• A DBMS supports multiple users, ie, multiple transactions may be running concurrently.
  – Eg. User 2 views available seats and reserves seat 22A.
  – Eg. User 3 views available seats and reserves seat 23D.
ACID Properties of Transactions

- **Atomicity**: all-or-nothing execution of transactions
- **Consistency**: constraints on data elements is preserved
- **Isolation**: each transaction executes as if no other transaction is executing concurrently
- **Durability**: effect of an executed transaction must never be lost
Atomicity

• A transaction might commit after completing all its actions, or it could abort (or be aborted by the DBMS) after executing some actions.

• A very important property guaranteed by the DBMS for all transactions is that they are atomic. That is, a user can think of a Xact as always executing all its actions in one step, or not executing any actions at all.
  – DBMS logs all actions so that it can undo the actions of aborted transactions.
Example (Atomicity)

- The first transaction is transferring $100 from B’s account to A’s account.
- The second is crediting both accounts with a 6% interest payment.
- There is no guarantee that T1 will execute before T2 or vice-versa, if both are submitted together. However, the net effect must be equivalent to these two transactions running serially in some order.

T1: BEGIN
   A=A+100
   B=B-100
   END

T2: BEGIN
   A=1.06*A
   B=1.06*B
   END
Ensuring Isolation

- Scheduling concurrent transactions
- DBMS ensures that execution of \{T_1, \ldots, T_n\} is equivalent to some \textit{serial} execution \(T_1' \ldots T_n'\).
  - Idea: use \textit{locks} to serialize access to \textit{shared} objects
  - \textbf{Strict 2 Phase locking protocol:}
    - Before reading/writing an object, a transaction requests a lock on the object, and waits till the DBMS gives it the lock.
    - All locks are released at the end of the transaction.
    - What if \(T_j\) already has a lock on \(Y\) and \(T_i\) later requests a lock on \(Y\)? (\textbf{Deadlock!}) \(T_i\) or \(T_j\) is \textit{aborted} and restarted!
Files vs DBMS

- Swapping data between memory and files
- Difficult to add records to files
- Security & access control
- Do optimization manually
- Good for small data/files

- Run out of pointers (32bit)
- Code your own search algorithm
  - Search on different fields is difficult
- Must protect data from inconsistency due to concurrency
- Fault tolerance – crash recovery
Where is the photo I took last Christmas?
Where did I read about “Turing Machines”?
Where is the invoice for this computer?
Which product is the most profitable?
Unstructured Data

• What are some examples of unstructured data?
• How do we model unstructured data?
• How do we query unstructured data?
• How do we process queries on unstructured data?
• How do we index unstructured data?
Unstructured Text Data

• Field of “Information Retrieval”

• Data Model
  – Collection of documents
  – Each document is a **bag of words (aka terms)**

• Query Model
  – **Keyword** + Boolean Combinations
  – Eg. DBMS and SQL and tutorial

• Details:
  – Not all words are equal. **“Stop words”** (eg. “the”, “a”, “his” ...) are ignored.
  – **Stemming**: convert words to their basic form. Eg. “Surfing”, “surfied” becomes “surf”
Inverted Indexes

• Recall: an index is a mapping of search key to data entries
  – What is the search key?
  – What is the data entry?

• Inverted Index:
  – For each term store a list of postings
  – A posting consists of <docid, position> pairs

What is the data in an inverted index sorted on?
Lookups using Inverted Indexes

- Given a single keyword query “k” (eg. SQL)
  - Find k in the lexicon
  - Retrieve the posting list for k
  - Scan posting list for document IDs [and positions]
- What if the query is “k1 and k2”?
  - Retrieve document IDs for k1 and k2
  - Perform intersection
Too Many Matching Documents

• Rank the results by “relevance”!
• Vector-Space Model
  – **Documents** are **vectors** in high-dimensional space
  – Each dimension in the vector represents a term
  – **Queries** are represented as **vectors** similarly
  – **Vector distance** (dot product) between query vector and document vector gives ranking criteria
  – **Weights** can be used to tweak relevance
• PageRank (later)
Internet Search Engines

World Wide Web

Web Crawler

Web Page Repository

Search Engine Web Server

Indexer

Inverted Index

Keyword Query

Ranked Results

Snippets

Query

Doc IDs

Postings etc
Basic Web Search


<table>
<thead>
<tr>
<th>Query Expression</th>
<th>What it means</th>
</tr>
</thead>
<tbody>
<tr>
<td>Biking italy</td>
<td>Biking <strong>AND</strong> italy</td>
</tr>
<tr>
<td>Recycle steel OR iron</td>
<td>Recycle <strong>AND</strong> (steel <strong>OR</strong> iron)</td>
</tr>
<tr>
<td>“I have a dream”</td>
<td>“I have a dream” treated as one term</td>
</tr>
<tr>
<td>Salsa -dance</td>
<td>Salsa <strong>AND NOT</strong> dance</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Other nifty expressions</th>
<th>What it means</th>
</tr>
</thead>
<tbody>
<tr>
<td>12 + 34 - 56 * 7 / 8</td>
<td>Evaluates the arithmetic expression</td>
</tr>
<tr>
<td>300 Euros in USD</td>
<td>Converts 300 euros to US currency</td>
</tr>
</tbody>
</table>
Ranking Web Pages

- **Google’s PageRank**
  - Links in web pages provide clues to how important a webpage is.

- **Take a random walk**
  - Start at some webpage \( p \)
  - Randomly pick one of the links and go to that webpage
  - Repeat for all eternity

- **The number of times the walker visits a page is an indication of how important the page is.**

Vertices represent web pages. Edges represent web links.
## Semi-structured Search

Web pages are not really unstructured! Click “view source” to view HTML.

<table>
<thead>
<tr>
<th>Query Expression</th>
<th>What it means</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>define</strong>: imbroglio</td>
<td>Find definitions of “imbroglio”</td>
</tr>
<tr>
<td>Halloween <strong>site</strong>: <a href="http://www.census.gov">www.census.gov</a></td>
<td>Restrict search for “halloween” to US census website</td>
</tr>
<tr>
<td>Form 1098-T IRS <strong>filetype</strong>: pdf</td>
<td>Find the US tax form 1098-T in PDF format</td>
</tr>
<tr>
<td><strong>link</strong>: warriorlibrarian.com</td>
<td>Find pages that link to Warrior Librarian's website</td>
</tr>
<tr>
<td>Dan Shugar <strong>intext</strong>: Powerlight</td>
<td>Find pages mentioning Dan Shugar where his company, Powerlight, is included in the text of the page, i.e., less likely to be from the corporate website.</td>
</tr>
<tr>
<td>allintitle: Google Advanced Operators</td>
<td>Search for pages with titles containing &quot;Google,&quot; &quot;Advanced,&quot; and &quot;Operators&quot;</td>
</tr>
</tbody>
</table>
Summary

• Data Management Problem
  – How do we pose and answer queries on data?

• Structured data
  – Relational Data Model
  – SQL
  – Relational DBMS
  – Transactions

• Unstructured data
  – Bag of terms
  – Boolean combination of keyword queries
  – Inverted Indexes (Web Search Engines)

• Semi-structured data
  – Could use techniques from either structured or unstructured
  – More sophisticated keyword queries