CMSC424: Database Design
Introduction/Overview

Professor: Pete Keleher
keleher@cs.umd.edu

Today

- Administrivia
  - Workload etc.

- Motivation: Why study databases? What are databases?

- Current Industry Outlook

- A typical DBMS at a glance
DBMSs to the Rescue

- Provide a systematic way to answer many of these questions...
- Aim is to allow easy management of high volumes of data
  - Storing, Updating, Querying, Analyzing...

- What is a Database?
  - A large, integrated collection of (mostly structured) data
  - Typically models and captures information about a real-world enterprise
    - Entities (e.g. courses, students)
    - Relationships (e.g. John is taking CMSC 424)
  - Usually also contains:
    - Knowledge of constraints on the data (e.g. course capacities)
    - Business logic (e.g. pre-requisite rules)
    - Encoded as part of the data model (preferable) or through external programs

DBMSs to the Rescue

- Massively successful for highly structured data
  - Why? Structure in the data (if any) can be exploited for ease of use and efficiency
    - If there is no structure in the data, hard to do much
    - Contrast managing emails vs managing photos
  - Much of the data we need to deal with is highly structured
  - Some data is semi-structured
    - E.g.: Resumes, Webpages, Blogs etc.
  - Some has complicated structure
    - E.g.: Social networks
  - Some has no structure
    - E.g.: Text data, Video/Image data etc.
Structured vs Unstructured Data

- A lot of the data we encounter is structured
  - Some have very simple structures
    - E.g. Data that can be represented in tabular forms
    - Significantly easier to deal with
    - We will focus on such data for much of the class

### Account
<table>
<thead>
<tr>
<th>bname</th>
<th>acct_no</th>
<th>balance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Downtown</td>
<td>A-101</td>
<td>500</td>
</tr>
<tr>
<td>Mianus</td>
<td>A-215</td>
<td>700</td>
</tr>
<tr>
<td>Perry</td>
<td>A-102</td>
<td>400</td>
</tr>
<tr>
<td>R.H</td>
<td>A-305</td>
<td>350</td>
</tr>
</tbody>
</table>

### Customer
<table>
<thead>
<tr>
<th>cname</th>
<th>cstreet</th>
<th>ccity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jones</td>
<td>Main</td>
<td>Harrison</td>
</tr>
<tr>
<td>Smith</td>
<td>North</td>
<td>Rye</td>
</tr>
<tr>
<td>Hayes</td>
<td>Main</td>
<td>Harrison</td>
</tr>
<tr>
<td>Curry</td>
<td>North</td>
<td>Rye</td>
</tr>
<tr>
<td>Lindsay</td>
<td>Park</td>
<td>Pittsfield</td>
</tr>
</tbody>
</table>

- Some data has a little more complicated structure
  - E.g. graph structures
    - Map data, social networks data, the web link structure etc
  - Can convert to tabular forms for storage, but may not be optimal
  - Queries often reason about graph structure
    - Find my “Erdos number”
    - Suggest friends based on current friends
  - Growing importance in recent years in a variety of domains: Biological, social networks, web...
Structured vs Unstructured Data

- Increasing amount of data in a semi-structured format
  - XML – Self-describing tags (HTML?)
  - Complicates a lot of things
  - We will discuss this toward the end

- A huge amount of data is unfortunately unstructured
  - Books, WWW
  - Amenable to pretty much only text search... so far
    - Information Retrieval research deals with this topic
  - What about Google search?
    - Google search is mainly successful because it uses the structure (in its original incarnation)

- Video? Music?
  - Can represent in DBMS’s, but can’t really operate on them

DBMSs to the Rescue

- Massively successful for highly structured data
  - Two Key Concepts:
    - **Data Modeling**: Allows reasoning about the data at a high level
      - e.g. “emails” have “sender”, “receiver”, “…”
      - Once we can describe the data, we can start “querying” it
    - **Data Abstraction/Independence**: Layer the system so that the users/applications are insulated from the low-level details
DBMSs to the Rescue: Data Modeling

- Data modeling
  - **Data model**: A collection of concepts that describes how data is represented and accessed
  - **Schema**: A description of a specific collection of data, using a given data model

- Some examples of data models that we will see
  - Relational, Entity-relationship model, XML, JSON...
  - Object-oriented, object-relational, semantic data model, RDF...

- Why so many models?
  - Tension between descriptive power and ease of use/efficiency
  - More powerful models → more data can be represented
  - More powerful models → harder to use, to query, and less efficient

DBMSs to the Rescue: Data Abstraction

- Probably **the** most important purpose of a DBMS
- Goal: Hiding *low-level details* from the users of the system
  - Alternatively: the principle that
    - *applications and users should be insulated from how data is structured and stored*
  - Also called *data independence*

- Through use of *logical abstractions*
Data Abstraction

What data users and application programs see?

What data is stored?
describe data properties such as data semantics, data relationships

How data is actually stored?
e.g. are we using disks? Which file system?

Logical Data Independence
Protection from logical changes to the schema

Physical Data Independence
Protection from changes to the physical structure of the data
Data Abstractions: Example

A View Schema

course_info(#registered, …)

View Level

View 1  View 2  …  View n

Logical Schema

students(sid, name, major, …)
courses(cid, name, …)
enrolled(sid, cid, …)

Logical Level

Physical Schema

all students in one file ordered by sid
courses split into multiple files by colleges

Physical Level

What about a Database System?

- A DBMS is a software system designed to store, manage, facilitate access to databases

- Provides:
  - Data Definition Language (DDL)
    - For defining and modifying the schemas
  - Data Manipulation Language (DML)
    - For retrieving, modifying, analyzing the data itself
  - Guarantees about correctness in presence of failures and concurrency, data semantics etc.

- Common use patterns
  - Handling transactions (e.g. ATM Transactions, flight reservations)
  - Archival (storing historical data)
  - Analytics (e.g. identifying trends, Data Mining)
Relational DBMS: SQL

- **SQL (sequel):** Structured Query Language

- **Data definition (DDL)**
  - `create table` `instructor` (  
    - `ID` `char(5)`,  
    - `name` `varchar(20)`,  
    - `dept_name` `varchar(20)`,  
    - `salary` `numeric(8,2)`)  

- **Data manipulation (DML)**
  - Example: Find the name of the instructor with ID 22222
    - `select` `name`  
    - `from` `instructor`  
    - `where` `instructor.ID = '22222'`

Current Industry Outlook

- **Relational DBMSs**
  - Oracle, IBM DB2, Microsoft SQL Server, Sybase

- **Open source alternatives**
  - MySQL, PostgreSQL, Apache Derby, BerkeleyDB (mainly a storage engine – no SQL), neo4j (graph data) ...

- **Data Warehousing Solutions**
  - Geared towards very large volumes of data and on analyzing them
  - Long list: Teradata, Oracle Exadata, Netezza (based on FPGAs), Aster Data (founded 2005), Vertica (column-based), Kickfire, Xtremedata (released 2009), Sybase IQ, Greenplum (eBay, Fox Networks use them)
  - Usually sell package/services and charge per TB of managed data
  - Many (especially recent ones) start with MySQL or PostgreSQL and make them parallel/faster etc..
Web Scale Data Management, Analysis

- Ongoing debate/issue
  - Cloud computing seems to eschew DBMSs in favor of homegrown solutions
  - E.g. Google, Facebook, Amazon etc...

- MapReduce: A paradigm for large-scale data analysis
  - Hadoop: An open source implementation
  - Apache Spark: a better open source implementation

- Why?
  - DBMSs can’t scale to the needs, not fault-tolerant enough
    - These apps don’t need things like transactions, that complicate DBMSs (? ? ?)
  - Mapreduce favors Unix-style programming, doesn’t require SQL
    - Try writing SVMs or decision trees in SQL
  - Cost
    - Companies like Teradata may charge $100,000 per TB of data managed

Current Industry Outlook

- Bigtable-like
  - Called “key-value stores”
  - Think highly distributed hash tables
  - Allow some transactional capabilities – still evolving area
  - PNUTS (Yahoo), Apache Cassandra (Facebook), Dynamo (Amazon), and many many others

- Mapreduce-like
  - Hadoop (open source), Pig (@Yahoo), Dryad (@Microsoft), Spark
  - Amazon EC2 Framework
  - Not really a database – but increasing declarative SQL-like capabilities are being added (e.g. HIVE at Facebook)

- Much ongoing research in industry and academia
DBMS at a glance

- **Data Models**
  - Conceptual representation of the data

- **Data Retrieval**
  - How to ask questions of the database
  - How to answer those questions

- **Data Storage**
  - How/where to store data, how to access it

- **Data Integrity**
  - Manage crashes, concurrency
  - Manage semantic inconsistencies

- Not fully disjoint categorization !!

---

What we will cover...

- We will mainly discuss structured data
  - That can be represented in tabular forms (called Relational data)
  - We will spend some time on XML
  - We will also spend some time on Mapreduce-like stuff
  - Structured is still the biggest and most important business (?)
    - Well defined problem with really good solutions that work
    - Contrast XQuery for XML vs SQL for relational
    - Solid technological foundations

- Many of basic techniques work for *unstructured* as well
  - E.g. reliable data storage etc.
  - Cf. Many recent attempts to add SQL-like capabilities, transactions to Mapreduce and related technologies
    - E.g., Spark DataFrames
What we will cover...

- representing information
  - data modeling
  - semantic constraints

- languages and systems for querying data
  - complex queries & query semantics
  - over massive data sets

- concurrency control for data manipulation
  - ensuring transactional semantics

- reliable data storage
  - maintain data semantics even if you pull the plug
  - fault tolerance

- representing information
  - data modeling: relational models, E/R models, XML/JSON
  - semantic constraints: integrity constraints, triggers

- languages and systems for querying data
  - complex queries & query semantics: SQL
  - over massive data sets: indexes, query processing, optimization, parallelization/cluster processing, streaming

- concurrency control for data manipulation
  - ensuring transactional semantics: ACID properties, distributed consistency

- reliable data storage
  - maintain data semantics even if you pull the plug: durability
  - fault tolerance: RAID
Summary

Why study databases?
- Shift from *computation* to *information*
  - Always true in *corporate* domains
  - Increasing true for *personal* and *scientific* domains
- Need has exploded in recent years
  - Data is growing at a very fast rate
- Solving the data management problems is going to be a key

Database Management Systems provide
- Data abstraction: Key in evolving systems
- Guarantees about data integrity
  - In presence of concurrent access, failures...
- Speed !!