Introduction to Database Systems

Lecture 1: Introduction

Data Management is Universal

- Managing data is at the core of most apps / services
  - whether they store small or large amounts of data
  - whether they are modern systems or older ones
- Hard problems even with small amounts of data
  - we'll see examples later on…
- Doing it right typically makes everything else easier

Motivation

- The world is drowning in data
  - affects almost every app / service
- Need professionals to help manage it
  - help domain scientists achieve new discoveries
  - help companies provide better services
  - help governments become more efficient
- Introduction to Database Systems
  - covers both *principles* and *tools*

Course Format

- Lectures: 20 sessions
  - Content: tutorials, individual exercises, group exercise
- Practice: 2 sessions
  - Lab room
- Grading:
  - Attendance: 10%
  - Exercise: 20%
  - Mid-term test: 10%
  - Final: 60%

Communications

- Web page/Classroom:
  - Syllabus is there
  - Lecture slides will be available there
  - Other announcements will be available there
  - Submitting exercise

  (class code: announce later)

- Mailing:
  - Email: phantha.72@gmail.com

Textbook

Main textbook, available at the bookstore:

- *Database Systems: The Complete Book*,
  Hector Garcia-Molina,
  Jeffrey Ullman,
  Jennifer Widom

  Second edition.

Covers most, but **not all**, of course content

Outline of Today's Lecture

- Overview of database mgmt systems
  - Why they are helpful
  - What are some of their key features
  - What are some of their key concepts

- Course content

Database

What is a database?

- A collection of files storing related data

Examples of databases

- Accounts database; payroll database; UW's students database; Amazon's products database; airline reservation database

Database Management System

What is a DBMS ?

- *A big program written by someone else that allows us to manage efficiently a large database and allows it to persist over long periods of time*

Examples of DBMSs

  - Oracle, IBM DB2, Microsoft SQL Server, Vertica, Teradata
  - Open source: MySQL (Sun/Oracle), PostgreSQL, AsterixDB
  - Open source library: SQLite

We will focus on relational DBMSs in most of the quarter

An Example: Online Bookseller

- What data do we need?
  - Data about books, customers, pending orders, order histories, trends, preferences, etc.
  - Data about sessions (clicks, pages, searches)
  - Note: data must be persistent! Outlive application
  - Also note that data is large… won't fit all in memory
- What capabilities on the data do we need?
  - Insert/remove books, find books by author/title/etc., analyze past order history, recommend books, …
  - Data must be accessed efficiently, by many users
  - Data must be safe from failures, malicious users, and bugs!

Multi-User Issues

- Jane and John both have ID number for gift certificate (credit) of $200 they got as a wedding gift
  - Jane @ her office orders "The Selfish Gene, R. Dawkins" ($80)
  - John @ his office orders "Guns and Steel, J. Diamond" ($100)

- Questions:
  - What is the ending credit?
  - What if second book costs $130?
  - What if the server crashes?
  - What if the data center goes offline?

Required Functionality for Data Management

1. Describe real-world entities in terms of stored data
2. Persistently store large datasets
3. Efficiently query & update
   - Must handle complex questions about data
   - Must handle sophisticated updates
   - Performance matters (users can feel 200ms latency)
4. Easily change structure (e.g., add attributes)
5. Enable simultaneous (đồng th) updates
6. Crash recovery
7. Security and integrity (tven)

DataBase Management System (DBMS)

- Very difficult to implement all these features inside the application (correctly)
- DBMS provides these features (and more)
- DBMS simplifies application development

Client-Server Architecture

- One *server* that stores the database (DBMS):
  - Usually a beefy system (Strong)
  - But can be your own desktop…
  - … or a huge cluster running a parallel DBMS
- Many *clients* run apps and connect to DBMS
  - E.g. Microsoft's SQL Server Management Studio
  - Or psql (for PostgreSQL)
  - Or some Java/C++ program (very typical)
- Clients "talk" to server using JDBC protocol
  - Often phone/browser <~> web server <~> DBMS

Key People

- **DB application developer**: writes programs that query and modify data
- **DB designer**: establishes schema
- **DB administrator**: loads data, tunes system, keeps whole thing running
- **Data analyst**: data mining, data integration
- **DBMS implementer**: builds the DBMS

Key Concepts

- **Data models**: how to describe real-world data
  - Relational, XML, JSon
- **Schema vs data**
- **Declarative query language**
  - Say what you want, not how to get it
- **Data independence**
  - Physical independence: Can change how data is stored on disk without affecting applications
  - Logical independence: can change schema w/o affecting apps
- **Query optimizer** and compiler
- **Transactions**: isolation and atomicity

What This Course Contains

- **Focus: Using DBMSs**
- Relational Data Model
  - SQL, Relational Algebra, Datalog
- Semistructured Data Model
  - JSon, NoSQL, AsterixDB
- Conceptual design
  - E/R diagrams, Views, and Database normalization
- Transactions
- Parallel databases, MapReduce, and Spark