Why study databases?

- Interesting concepts and techniques.
- Spans computer science, including OS, languages, theory, AI, logic, etc.
- Databases have become increasingly important
  - shift from a focus on computation to information
  - data increases in volume and diversity.
- Jobs: In demand and well paid.
- Research: Many open problems.
Our first hour or so

- Some key concepts
- Examples to motivate the course
- Admin info
Databases and DBMSs

- Databases are everywhere, often behind the scenes.
- DBMS (Database Management System): “A powerful tool for creating and managing large amounts of data efficiently and allowing it to persist over long periods of time, safely.” [Ullman and Widom, FCDB]
- Database: a collection of data managed by a DBMS.
Data models

- Every DBMS is based on some data model: a notation for describing data, including
  - the structure of the data
  - constraints on the content of the data
  - operations on the data

- Some specific data models:
  - network & hierarchical data models — of historic interest
  - relational data model — the most widely used
  - semistructured data model
Comparing data models

Student job example
Mary (M) and Xiao (X) both work at Tim Hortons (T)
Jaspreet (J) works at both Bookstore (B) and Wind (W)

Hierarchical (tree)

Network (graph)

Relational (table)
Why the relational model?

- Matches how we think about data
- Real reason: *data independence*
- Earlier models tied to physical data layout
  - Procedural access to data (low-level, explicit access)
  - Relationships stored in data (linked lists, trees, etc.)
  - Change in data layout => application rewrite
- Relational model
  - Declarative access to data (system optimizes for you)
  - Relationships specified by queries (schemas help, too)
  - Develop, maintain apps and data layout separately

*Similar battle today with languages*
The relational data model

- Main concept is a “relation.”
  Based on the concept of relations in math.
- Can think of as tables of rows and columns.

<table>
<thead>
<tr>
<th>Teams</th>
<th>Name</th>
<th>Home Field</th>
<th>Coach</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Rangers</td>
<td>Runnymede CI</td>
<td>Tarvo Sinervo</td>
</tr>
<tr>
<td></td>
<td>Ducks</td>
<td>Humber Public</td>
<td>Maeve Mahar</td>
</tr>
<tr>
<td></td>
<td>Choppers</td>
<td>High Park</td>
<td>Tom Cole</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Games</th>
<th>Home team</th>
<th>Away team</th>
<th>Home goals</th>
<th>Away goals</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Rangers</td>
<td>Ducks</td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>Ducks</td>
<td>Choppers</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>Rangers</td>
<td>Choppers</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>Choppers</td>
<td>Ducks</td>
<td>0</td>
<td>5</td>
</tr>
</tbody>
</table>
Example ...

- A dataset scraped from Twitter
- Defining a schema that expresses its structure
- Creating an instance that contains the data
- Writing some queries on the data
What a DBMS provides

- Ability to specify the logical structure of the data
  - explicitly
  - and have it enforced
- Ability to query or modify the data.
- Good performance under heavy loads (huge data, many queries).
- Durability of the data.
- Concurrent access by multiple users/processes.
Overall architecture of a DBMS

- The DBMS sits between the data and the users or between the data and an application program.
- Within the DBMS are layers of software for:
  - parsing “queries”
  - implementing the fundamental operations
  - optimizing queries
  - maintaining indices on the data
  - accessing the files that store the data and indices
  - management of buffers
  - management of disk space
DBMS High-level Architecture

- Naïve users
- Application programmers
- Sophisticated users
- DB administrator

- Application interfaces
- Application programs
- Queries
- Database schema

- DBMS
  - Application programs
  - Object code
  - Embedded DML precompiler
  - Query evaluation engine
  - DML compiler
  - DDL interpreter
  - Transaction manager
  - Buffer manager
  - File manager

- Storage Manager
  - Disk Storage
  - Indices
  - Data files
  - Data dictionary
  - Statistical data
Advantages of a DBMS

- Data independence
- Efficient data access
- Data administration
- Data integrity & security
- Concurrent access, crash recovery
- Reduced application development time
Why not always use a DBMS?

- Expensive/complicated to set up & maintain
- Cost & complexity must be offset by need
- General-purpose, not suited for special-purpose tasks (e.g. text search!)
- The data must have a structure (schema)
A “semi-structured” example ...

- An xml dataset scraped from imdb.com
- No schema required, no instance made
- We can immediately write queries on the data
- A much looser approach
What this course is about

- **csc343** is about *using* DBMSs:
  - defining schemas and instances
  - writing queries
  - connecting to code written in a general-purpose language
  - rigorous underlying principles

- **csc443** is about implementation of the DBMS itself
  - database design, components, etc.