Today

- Introduction
  - About the class
  - Organization of this course

- Introduction to DBMS
Database Management Systems, Raghu Ramakrishnan, McGraw-Hill Education
The course web page will be updated regularly throughout the semester with lecture notes, announcements and important dates.

http://web.cs.hacettepe.edu.tr/~bbm371
Course Work and Grading

- **1 midterm exam (25 points)**
  - Closed book and notes

- **Pop-quizzes (22.5 points)**
  - Closed book
  - Will not be announced! Attend the lectures!

- **Pop-attendance (2.5 points)**
  - Attempting to create false attendance (e.g., signing in the attendance list on behalf of someone else) will be punished.

- **Final exam (50 points)**
## Course Overview

<table>
<thead>
<tr>
<th>Week</th>
<th>Date</th>
<th>Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>04-Oct</td>
<td>Introduction to DBMS</td>
</tr>
<tr>
<td>2</td>
<td>11-Oct</td>
<td>ER Diagram</td>
</tr>
<tr>
<td>3</td>
<td>18-Oct</td>
<td>Storage Devices</td>
</tr>
<tr>
<td>4</td>
<td>25-Oct</td>
<td>Basic File Concepts</td>
</tr>
<tr>
<td>5</td>
<td>1-Nov</td>
<td>Basic File Concepts – cont.</td>
</tr>
<tr>
<td>6</td>
<td>8-Nov</td>
<td>Index Types</td>
</tr>
<tr>
<td>7</td>
<td>15-Nov</td>
<td>Hash based file organization</td>
</tr>
<tr>
<td>8</td>
<td>22-Nov</td>
<td><strong>Midterm Exam 1</strong></td>
</tr>
<tr>
<td>9</td>
<td>29-Nov</td>
<td>Hash based file organization</td>
</tr>
<tr>
<td>10</td>
<td>6-Dec</td>
<td>Tree based indexing</td>
</tr>
<tr>
<td>11</td>
<td>13-Dec</td>
<td>Tree based indexing</td>
</tr>
<tr>
<td>12</td>
<td>20-Dec</td>
<td>External sorting</td>
</tr>
<tr>
<td>13</td>
<td>27-Dec</td>
<td>Spatial Indexes</td>
</tr>
<tr>
<td>14</td>
<td>3-Jan</td>
<td>Overview</td>
</tr>
</tbody>
</table>
What is a DBMS?

- A very large, integrated collection of data.
- Models real-world enterprise
- A Database Management System (DBMS) is a software package designed to store and manage databases
- Information about:
  - Entities: such as students, faculty, courses
  - Relationships: between entities for example a student is enrolled to a course
Data-Centric Applications

► Applications in which data plays an important role
  ► Airline reservation systems
    ► Data: aircrafts, flights, flight attendants, travelers, etc.

► Banking applications
  ► Data: clients, deposits, withdraws, etc.

► Hospital systems
  ► Data: patients, physicians, diagnosis, prescriptions, etc.

► University systems
  ► Data: students, teaching staff, courses, enrollments, etc.

Taken from the slides of Mohamed S. Hassan – Purdue University
History of DBMS

► Even from the early days of computers data must be stored for applications
► Late 1960 IBM’s Information Management System (IMS) for airline reservations.
► 1970s Edgar Codd proposed relational data model
► 1980s database query language SQL was standardized
► 1990s Data warehouses, consolidating data from multiple data stores for analysis
► 2000s Web applications
► Now – Even larger volumes of data NoSQL databases
Files vs. DBMS

- Imagine writing a program for a bank
  - Customers, Accounts, MoneyTransfers
  - More than 500 GB (does not fit in memory)

- Application must stage large datasets between main memory and secondary storage (500GB RAM is not still cheap!)

- Must protect data from inconsistency (update in ATM should be consistent with bank branch)

- Crash recovery

- Security and access control

- Concurrency (Transaction management)
Why Use a DBMS?

- Data independence and efficient access
- Reduced application and development time
- Data integrity and security
- Uniform data administration
- Concurrent access
- Recovery from crashes
Why Study Databases?

► Shift from computation to information
  ► Low-end users: Web Applications needs to organize information (a mess will not be effective)
  ► High-end users: Scientific applications now have data management problems!

► Datasets increasing in diversity and volume
  ► Digital libraries, interactive video, Human Genome project etc.

► DBMS encompasses most of CS
  ► OS, languages, AI, multimedia etc.
Data Models

- A **data model** is a collection of concepts for describing data. (high-level)
- A **schema** is a description of a particular collection of data, using the given data model
- The **relational model of data** is the most widely used model today.
  - Main concept: **relation**, basically a table with rows and columns
  - Every relation has a **schema**, which describes the columns, or fields.
  - Schema is defined by: name of schema, the name of each **field** (or attribute or column) and type of each field
    - e.g. **Students**(sid: string, name: string, login: string, age: integer, gpa: real)
## Entity: Student

**Students** (sid: string, name: string, login: string, age: integer, gpa: real)

<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@ee</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>

### Using age as a field is not a good idea, why?

- **Integrity Constraints:** We can define the field sid to be unique or age to be larger than 0. Rules for records to satisfy

**Record**
Unlike programmers of early systems, programmer of relational system does not need to implement lower level details. Many views, single conceptual (logical) schema and physical schema.

- Views describe how users see the data.
- Conceptual schema defines logical structure
- Physical schema describes the files and indexes used
### External Level

#### Dept-Table

<table>
<thead>
<tr>
<th>deptno</th>
<th>dNam</th>
<th>dAdr</th>
<th>dChair</th>
</tr>
</thead>
<tbody>
<tr>
<td>356</td>
<td>BilM</td>
<td>mmm</td>
<td>2</td>
</tr>
<tr>
<td>357</td>
<td>EleM</td>
<td>ggg</td>
<td>4</td>
</tr>
</tbody>
</table>

#### Staff-Table

<table>
<thead>
<tr>
<th>sld</th>
<th>sNam</th>
<th>sBranc</th>
<th>sPos</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Ebru</td>
<td>Inf.Ret</td>
<td>Assoc</td>
</tr>
<tr>
<td>2</td>
<td>HS</td>
<td>Inf.Ret</td>
<td>Prof</td>
</tr>
</tbody>
</table>

### Conceptual Layer

#### Dept-File

356BilMmmm2357EleMggg4

#### Staff-File

1EbruInf.Ret.Assoc2HSInf.Ret.Prof
Physical View

- The DBMS must know
  - exact physical location
  - precise physical structure

Employee record

A.B.C. De Silva       |222, Galle Road, Colombo       |
Name (20 characters)  Address (40 characters)

650370690V|Senior Lecturer

NID (10 char) Designation (15 char)
The conceptual model is a logical representation of the entire contents of the database.

The conceptual model is made up of base tables.

Base tables are “real” in that they contain physical records.
External View

- The user/application see
  - authorised data
  - own format
External View cont.

- **External Views Allows to**
  - hide unauthorised data
    - e.g. salary, dob
  - provide user view
    - e.g. view employee name, designation, department data taken from employee and department files
  - derive new attributes
    - e.g. age derived from dob or nid
Example: University Database

- Conceptual schema:
  - Students(sid:string, name:string, login:string, age:integer, gpa:real)
  - Courses(cid:string, cname:string, credits:integer)
  - Enrolled(sid:string, cid:string, grade:string)

- Physical schema:
  - Relations stored as unordered files
  - Index on first column of Students

- External Schema (View):
  - Course_info(cid:string, enrollment:integer)
Data Independence

► Applications insulated from how data is structured and stored.

► **Logical data independence:** Protection from changes in logical structure of data.

► **Physical data independence:** Protection from changes in physical structure of data.

* One of the most important benefits of using a DBMS!
Concurrent execution of user programs is essential for good DBMS performance.

- Because disk accesses are frequent and relatively slow, it is important to keep the CPU humming by working on several user programs concurrently.

Interleaving actions of different user programs can lead to inconsistency: e.g., check is cleared while account balance is being computed.

DBMS ensures such problems don’t arise: users can pretend they are using a single-user system.
Structure of a DBMS

- A typical DBMS has a layered architecture.
- The figure does not show the concurrency control and recovery components.
- This is one of several possible architectures; each system has its own variations.

Database Management Systems 3ed, R. Ramakrishnan and J. Gehrke
An overview of Database Concepts

Querying
- Relational Algebra/Calculus
- Structured Query Language (SQL)
- Entity Relationship (ER) Diagrams

Database Design
- Normalization / Functional Dependencies
- Data Definition Language (DDL)

Transaction Management
- Locking Mechanisms (Avoiding Deadlocks)
- Crash Recovery

Physical Storage
- Utilizing the Memory Hierarchy (Buffering)
- How to Store Data in Files
- Finding data fast; Indexing Structures
- External Sorting

Covered in BBM 471
 Covered in this Course
Querying

- Relational Algebra: Operators defined over Relations; union, intersection, difference, selection, projection, products, joins

\[ \pi_{\text{name}}((\sigma_{\text{bid}=103} \text{Reserves}) \bowtie \text{Sailors}) \]

- Structured Query Language: A declarative programming language for Relational Algebra. Can you understand the query below?
  - SELECT name FROM students WHERE surname='ERCAN'

- SQL corresponding to above Relational Algebra Query
  
  ```sql
  SELECT S.sname
  FROM Sailors S
  WHERE EXISTS (SELECT *
                    FROM Reserves R
                    WHERE R.bid=103 AND S.sid=R.sid)
  ```
Database Design

- ER Diagrams

- Functional Dependencies and Normal Forms:
  - Mathematical model to get rid of redundancy in the database design
  - Redundancy makes updates more difficult or results in inconsistent data
  - This model expresses desired relational design in a domain independent way!
Two users performing operations on a joint account at the same time.

If one reads before the other writes back, the first to write will be cancelled.

It will work ok if read and insert is atomic (not interrupted).

To make sure, we can lock the account.
A prepaid mobile phone user will transfer 10 credit to User 2.

This operation needs two steps

If trying to remove 10 credits from User 1 fails for some reason, we have added 10 credits to U2 out of the blue

If we perform the operation in a transaction, we can roll-back the changes.
Databases make these folks happy...

- End users and DBMS vendors
- DP application programmers
  - E.g. smart webmasters
- Database administrator
  - Design logical / physical schemas
  - Handles security and authorization
  - Data availability, crash recovery
End of the first lecture...