Course Outline

- What are the database systems
- Data models and data abstractions
- Query formulation
- Database design based on normal forms
- Physical organization: indexing and hashing
- Transactions: concurrency control and recovery
- Other topics: OODB, RTDB, data warehousing
Database Systems

- Need for information
  - information age and information explosion
  - information: structured and interpreted data
  - one of the major driving force on the growing use of computers individually and organizationally
  - plays a critical role in almost all areas where computers are used: business, engineering, education

- Database
  - a collection of self-describing integrated data
  - represents some aspects of the real world
  - a logically coherent collection
  - designed, built, and populated with data for a specific purpose (users and applications)
A collection of self-describing integrated data

- self-describing: file layouts and structures are kept in data dictionary; user need not know them
- integrated: data aggregation
  bits > bytes > attributes > tuples > relations

Representation of a mini-world

- database objects represent entities of interest whose properties (attributes) are represented by data values
- database operations (transactions) represent events of interest that add, delete, update objects
Database Management System (DBMS)

- Definition
  - a collection of programs that enables users to create and maintain a database
  - a general purpose software system that facilitates the process of defining, constructing, and manipulating databases for various applications

- Defining a database
  - specifying the data types, structures, and constraints for the data to be stored in the database

- Constructing a database
  - process of storing the data on some storage medium controlled by the DBMS

- Manipulating a database
  - retrieve specific data and update to reflect changes
Database Management System

- Primary goal
  - to provide an environment that is convenient and efficient to use in storing and retrieving data
  - efficiency should not impair convenience:
    data must be available to users (applications) in a form that can be used immediately
  - one major reason for data abstraction
  - factors for efficiency: data structure, access method, and query optimization.

- Query optimization
  
  retrieve (name = Emily) AND (age > 20):
  
  search for the name first or age first
Requirements for Database Management System

- Maintain the integrity of the database
  - a set of assertions (integrity constraints) must hold

- Maintain the security of the database
  - access control (not everyone can access all data)
  - multi-level security

- Synchronization
  - control concurrent access for improved performance
  - anomalies not allowed: lost update, inconsistent read

- Replication control
  - reduce duplicated information
  - control divergence
● Crash protection and recovery
  - logging, checkpointing, recovery management
  - redo and undo rules

● Data independence
  - application programs should be immune to the change of physical database and/or access methods, etc.
Data Dependence

- Traditional file processing
  user --- application program --- files
  user --- application program --- files
  - data isolation and duplication
  - definitions of file structures are built into every application program that uses it
  - a change to that structure requires changes to all those application programs
  - several applications may need the same data in different formats
  - having multiple copies of the file in different format: space and potential inconsistency among copies
Data Independence

- **DBMS approach**
  - real solution: data abstraction
  - it is the name of the game in database systems
  - one copy at one location of all data
  - access to the data only through DBMS:
    no application programs directly touch the data
    user --- application program ---
    DBMS -- files
  - user --- application program ---
  - DBMS offers a stable view of the data, which is not affected by reformatting or reorganization of data
  - many different views of the same data are supported
Logical and Physical Data Organization

- Logical organization
  - conceptual or logical format of the data
  (e.g., employee record has E#, Name, Address)

- Physical organization
  - actual structure of the data and all supporting access structures (e.g., index)
  (e.g., employee: E# 32 bits
  Name 30 bytes
  Address 50 bytes)

- Benefit
  - application programs must know the logical organization
  but the physical organization is an implementation detail
  they need not know
DBMS Architecture

- Different abstract levels
  - a widely accepted general architecture for a database
  - database described by three abstract levels
    - internal schema (physical database)
    - conceptual schema (conceptual database)
    - external schema (view)

- Objectives
  - insulation of application programs and data
  - support of multiple user views
  - use of schema to store the DB description (mete-data)
The Three Schema Architecture

- **External schema**
  - describes a subset of the database that a particular user group is interested in, according to the format the user wants, and hides the rest
  - may contain virtual data that is derived from the files, but is not explicitly stored

- **Conceptual schema**
  - hides the details of physical storage structures and concentrates on describing entities, data types, relationships, operations, and constraints.

- **Internal schema**
  - describes the physical storage structure of the DB
  - uses a low-level (physical) data model to describe the complete details of data storage and access paths
Three Schema Architecture

- Data and meta-data
  - three schemas are only meta-data (descriptions of data)
  - data actually exists only at the physical level

- Mapping
  - DBMS must transform a request specified on an external schema into a request against the conceptual schema, and then into the internal schema
  - requires information in meta-data on how to accomplish the mapping among various levels
  - overhead (time-consuming) leading to inefficiencies
  - few DBMSs have implemented the full three-schema architecture
Benefits of Three Schema Architecture

- **Logical data independence**
  - the capacity to change the conceptual schema without having to change external schema or application programs
  
  ex: Employee (E#, Name, Address, Salary)
  
  A view including only E# and Name is not affected by changes in any other attributes.

- **Physical data independence**
  - the capacity to change the internal schema without having to change the conceptual (or external) schema
  
  - internal schema may change to improve the performance (e.g., creating additional access structure)
  
  - easier to achieve logical data independence, because application programs are dependent on logical structures
Data Models

• Data abstraction
  - one fundamental characteristic of the database approach
  - hides details of data storage that are not needed by most
database users and applications

• Data model
  - a set of data structures and conceptual tools
    used to describe the structure of a database
    (data types, relationships, and constraints)
  - used in the definition of the conceptual,
    external, and internal schema
  - must provide means for DB designers to represent
    the real-world information completely and naturally
Data Models

- High-level (conceptual) data models
  - use concepts such as entities, attributes, relationships
  - object-based models: ER model, OO model

- Representational (implementation) data models
  - most frequently used in commercial DBMSs
  - record-based models: relational, hierarchical, network

- Low-level (physical) data models
  - to describe the details of how data is stored
  - captures aspects of database system implementation:
    record structures (fixed/variable length) and ordering,
    access paths (key indexing), etc.
Schemas and Instances

In any data model, it is important to distinguish between the description of the database and the database itself.

- **Database schema (meta-data)**
  - overall description of a database, specified by a set of definitions
  - specified during database design (not change frequently)
  - similar to the notion of *type definition* in programs

- **Database instance**
  - current contents of the database (actual data): DB state
  - may change frequently

- **Distinction between database schema and database state**
  - a database just specified (or defined) is in empty state
  - initial state would be achieved when the data is loaded
  - DBMS is responsible to ensure every database state is valid
Data Definition and Manipulation Languages

- Data definition language (DDL)
  - not a procedural language
  - notations for describing the types of entities and relationships among entities
  - DDL statements converted into data dictionary

- Data manipulation language (DML)
  - for accessing and modifying data
  - non-procedural: specifying "what" to access
  - procedural: specifying "what" and "how" to get
  - non-procedural languages could be easy to use but may not be efficient
DBMS Classification

• Criteria
  - data model on which DBMS is based
  - number of users supported by DBMS: single/multi user
  - number of sites: centralized vs distributed
  - homogeneity: homogeneous vs heterogeneous (federated)
  - general-purpose vs special-purpose
    <ex> airline reservation and telephone directory systems
    on-line transaction processing (OLTP) systems need to
    support large # of concurrent transactions w/o delays

• Data model
  - the main criterion for classification
  - entity-relationship (ER) model
  - object-oriented (OO) model
  - relational, network, hierarchical model
Data Models

- **ER model**
  - popular high-level conceptual model used in DB design
  - proposed by P. Chen in 1976 (ACM TODS)
  - perception of real-world consisting of a collection of entities and relationships among them

- **OO model**
  - DB is defined in terms of objects, their properties, and their operations (methods)

- **Relational model**
  - represents a DB as a collection of tables

- **Network model**
  - represents DB as record types and 1:N relationships

- **Hierarchical model**
  - represents data as hierarchical tree structures