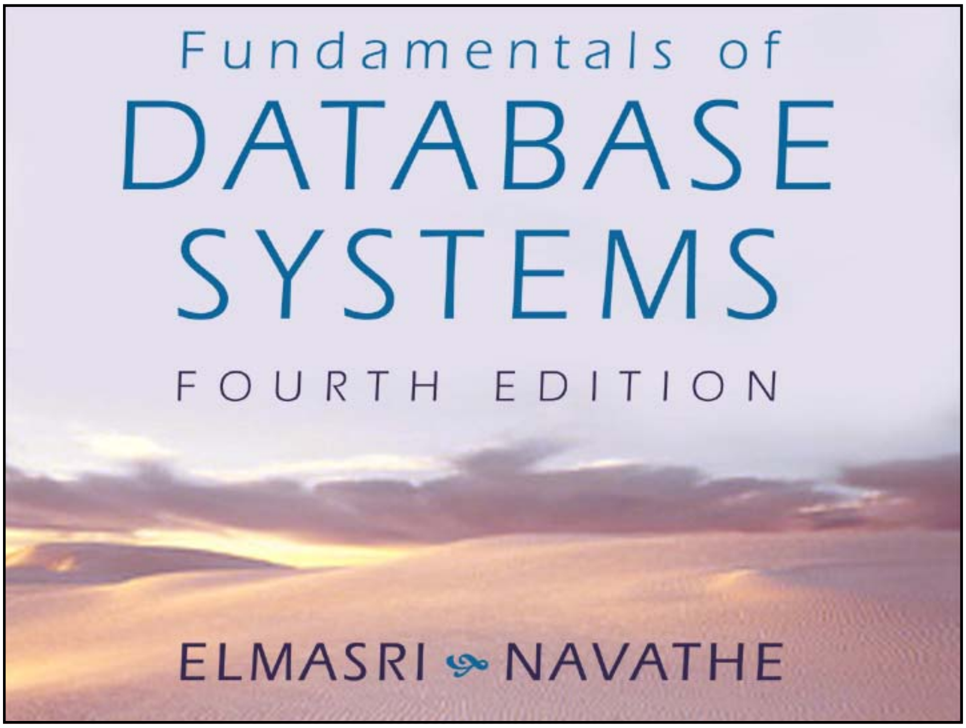

Chapter 1

Introduction and Conceptual Modeling

Types of Databases and Database Applications

- Numeric and Textual Databases
- Multimedia Databases
- Geographic Information Systems (GIS)
- Data Warehouses
- Real-time and Active Databases

A number of these databases and applications are described later in the book (see Chapters 24,28,29)

Basic Definitions

- **Database**: A collection of related data.
- **Data**: Known facts that can be recorded and have an implicit meaning.
- **Mini-world**: Some part of the real world about which data is stored in a database. For example, student grades and transcripts at a university.
- **Database Management System (DBMS)**: A software package/ system to facilitate the creation and maintenance of a computerized database.
- **Database System**: The DBMS software together with the data itself. Sometimes, the applications are also included.

Typical DBMS Functionality

- Define a database : in terms of data types, structures and constraints
- Construct or Load the Database on a secondary storage medium
- Manipulating the database : querying, generating reports, insertions, deletions and modifications to its content
- Concurrent Processing and Sharing by a set of users and programs – yet, keeping all data valid and consistent

Typical DBMS Functionality

Other features:

- Protection or Security measures to prevent unauthorized access
- “Active” processing to take internal actions on data
- Presentation and Visualization of data

Example of a Database (with a Conceptual Data Model)

- **Mini-world for the example**: Part of a UNIVERSITY environment.
- **Some mini-world *entities***:
  - STUDENTs
  - COURSEs
  - SECTIONs (of COURSEs)
  - (academic) DEPARTMENTs
  - INSTRUCTORs

Note: The above could be expressed in the ENTITY-RELATIONSHIP data model.

Example of a Database (with a Conceptual Data Model)

- **Some mini-world *relationships***:
  - SECTIONs *are of* specific COURSEs
  - STUDENTs *take* SECTIONs
  - COURSEs *have* prerequisite COURSEs
  - INSTRUCTORs *teach* SECTIONs
  - COURSEs *are offered by* DEPARTMENTs
  - STUDENTs *major in* DEPARTMENTs

Note: The above could be expressed in the *ENTITY-RELATIONSHIP* data model.

Main Characteristics of the Database Approach

- Self-describing nature of a database system: A DBMS **catalog** stores the *description* of the database. The description is called **meta-data**). This allows the DBMS software to work with different databases.
- Insulation between programs and data: Called **program-data independence**. Allows changing data storage structures and operations without having to change the DBMS access programs.

Main Characteristics of the Database Approach

- Data Abstraction: A **data model** is used to hide storage details and present the users with a *conceptual view* of the database.
- Support of multiple views of the data: Each user may see a different view of the database, which describes *only* the data of interest to that user.

Main Characteristics of the Database Approach

- Sharing of data and multiuser transaction processing : allowing a set of concurrent users to retrieve and to update the database. Concurrency control within the DBMS guarantees that each **transaction** is correctly executed or completely aborted. OLTP (Online Transaction Processing) is a major part of database applications.

Database Users

Users may be divided into those who actually use and control the content (called "Actors on the Scene") and those who enable the database to be developed and the DBMS software to be designed and implemented (called "Workers Behind the Scene").

Database Users

Actors on the scene

- **Database administrators:** responsible for authorizing access to the database, for co-ordinating and monitoring its use, acquiring software, and hardware resources, controlling its use and monitoring efficiency of operations.
- **Database Designers:** responsible to define the content, the structure, the constraints, and functions or transactions against the database. They must communicate with the end-users and understand their needs.
- **End-users:** they use the data for queries, reports and some of them actually update the database content.

Categories of End-users

- **Casual** : access database occasionally when needed
- **Naïve or Parametric** : they make up a large section of the end-user population. They use previously well-defined functions in the form of "canned transactions" against the database. Examples are bank-tellers or reservation clerks who do this activity for an entire shift of operations.

Categories of End-users

- **Sophisticated** : these include business analysts, scientists, engineers, others thoroughly familiar with the system capabilities. Many use tools in the form of software packages that work closely with the stored database.
- **Stand-alone** : mostly maintain personal databases using ready-to-use packaged applications. An example is a tax program user that creates his or her own internal database.

Advantages of Using the Database Approach

- Controlling redundancy in data storage and in development and maintenence efforts.
- Sharing of data among multiple users.
- Restricting unauthorized access to data.
- Providing persistent storage for program Objects (in Object-oriented DBMS's – see Chs. 20-22)
- Providing Storage Structures for efficient Query Processing

Advantages of Using the Database Approach

- Providing backup and recovery services.
- Providing multiple interfaces to different classes of users.
- Representing complex relationships among data.
- Enforcing integrity constraints on the database.
- Drawing Inferences and Actions using

Additional Implications of Using the Database Approach

- Potential for enforcing standards: this is very crucial for the success of database applications in large organizations Standards refer to data item names, display formats, screens, report structures, meta-data (description of data) etc.
- Reduced application development time: incremental time to add each new application is reduced.

Additional Implications of Using the Database Approach

- Flexibility to change data structures: database structure may evolve as new requirements are defined.
- Availability of up-to-date information – very important for on-line transaction systems such as airline, hotel, car reservations.
- Economies of scale: by consolidating data and applications across departments wasteful overlap of resources and personnel can be avoided.

Historical Development of Database Technology

- **Early Database Applications:** The Hierarchical and Network Models were introduced in mid 1960's and dominated during the seventies. A bulk of the worldwide database processing still occurs using these models.
- **Relational Model based Systems:** The model that was originally introduced in 1970 was heavily researched and experimented with in IBM and the universities. Relational DBMS Products emerged in the 1980's.

Historical Development of Database Technology

- **Object-oriented applications:** OODBMSs were introduced in late 1980's and early 1990's to cater to the need of complex data processing in CAD and other applications. Their use has not taken off much.
- **Data on the Web and E-commerce Applications:** Web contains data in HTML (Hypertext markup language) with links among pages. This has given rise to a new set of applications and E-commerce is using new standards like XML (eXtended Markup Language).

Extending Database Capabilities

- **New functionality is being added to DBMSs in the following areas:**
  - Scientific Applications
  - Image Storage and Management
  - Audio and Video data management
  - Data Mining
  - Spatial data management
  - Time Series and Historical Data Management

The above gives rise to new research and development in incorporating new data types, complex data structures, new operations and storage and indexing schemes in database systems.

When not to use a DBMS

- **Main inhibitors (costs) of using a DBMS**:
  - High initial investment and possible need for additional hardware.
  - Overhead for providing generality, security, concurrency control, recovery, and integrity functions.
- **When a DBMS may be unnecessary:**
  - If the database and applications are simple, well defined, and not expected to change.
  - If there are stringent real-time requirements that may not be met because of DBMS overhead.
  - If access to data by multiple users is not required.

When not to use a DBMS

- **When no DBMS may suffice:**
  - If the database system is not able to handle the complexity of data because of modeling limitations
  - If the database users need special operations not supported by the DBMS.