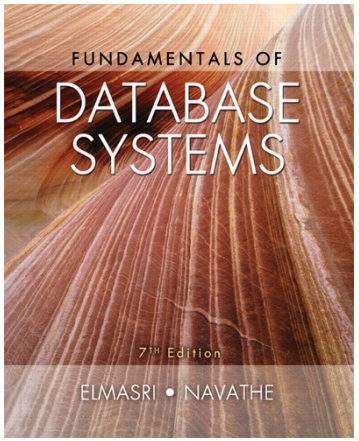

CHAPTER 1

Databases and Database Users

Outline

- Types of Databases and Database Applications
- Basic Definitions
- Typical DBMS Functionality
- Main Characteristics of the Database Approach
- Types of Database Users
- Advantages of Using the Database Approach
- Historical Development of Database Technology
- Extending Database Capabilities
- When Not to Use Databases

Types of Databases and Database Applications

- Traditional Applications:
  - Numeric and Textual Databases
- More Recent Applications:
  - Multimedia Databases
  - Geographic Information Systems (GIS)
  - Biological and Genome Databases
  - Data Warehouses
  - Mobile databases
  - Real-time and Active Databases

Basic Definitions

- **Database:**
  - A collection of related data.
- **Data:**
  - Known facts that can be recorded and have an implicit meaning.
- **Context/Problem (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.

Impact of Databases and Database Technology

- Businesses: Banking, Insurance, Retail, Transportation, Healthcare, Manufacturing
- Service Industries: Financial, Real-estate, Legal, Electronic Commerce, Small businesses
- More recently: Social Networks, Environmental and Scientific Applications, Medicine and Genetics, Smart mobile devices

Simplified database system environment

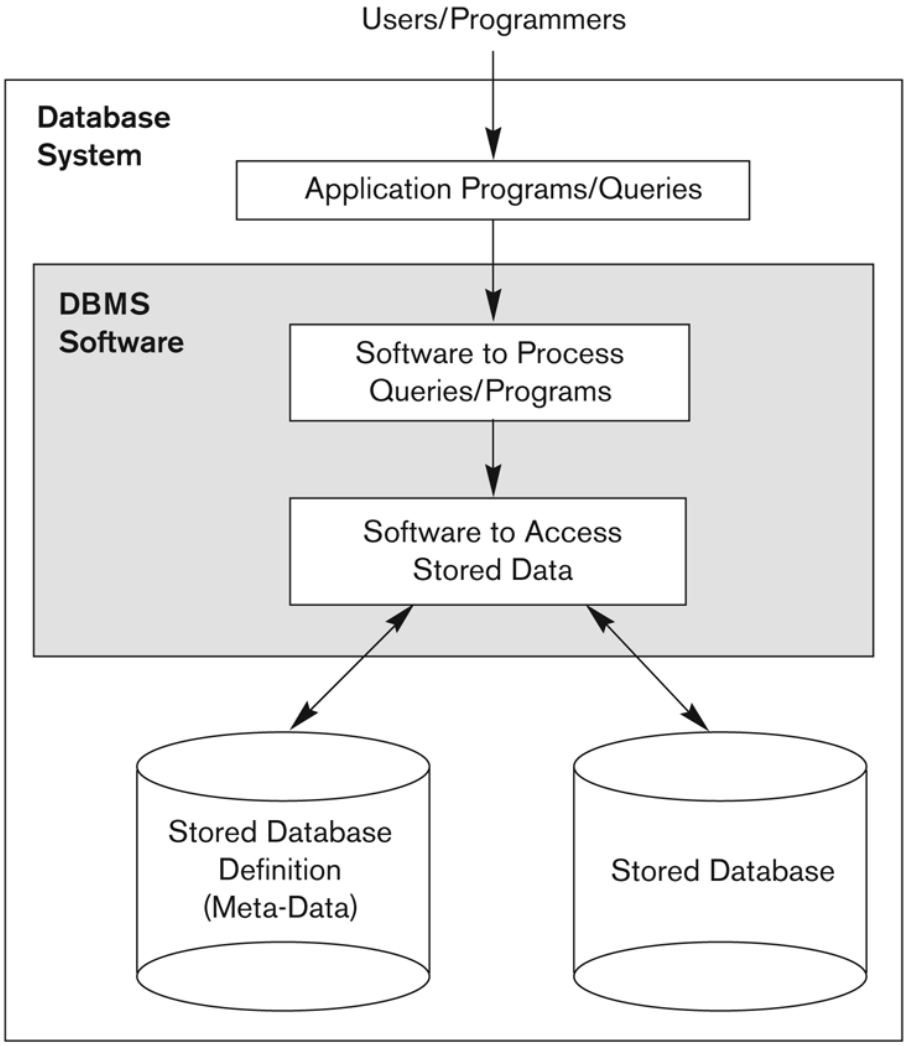

Figure 1.1
A simplified database system environment.

Typical DBMS Functionality

- ***Define*** a particular database in terms of its data types, structures, and constraints
- ***Construct*** or Load the initial database contents on a secondary storage medium
- ***Manipulating*** the database:
  - Retrieval: Querying, generating reports
  - Modification: Insertions, deletions and updates to its content
  - Accessing the database through Web applications
- ***Processing*** **and** ***sharing*** by a set of concurrent users and application programs – yet, keeping all data valid and consistent

Application Activities Against a Database

- Applications interact with a database by generating
  - **Queries**: that access different parts of data and formulate the result of a request
  - **Transactions**: that may read some data and "update" certain values or generate new data and store that in the database
- Applications must not allow **unauthorized** users to access data
- Applications must keep up with **changing user requirements** against the database

Additional DBMS Functionality

- DBMS may additionally provide:
  - Protection or Security measures to prevent unauthorized access
  - “Active” processing to take internal actions on data
  - Presentation and Visualization of data
  - Maintenance of the database and associated programs over the lifetime of the database application

Example of a Database

- Mini-world for the example:
  - Part of a UNIVERSITY environment.

- Some mini-world *entities*:
  - STUDENTs
  - COURSEs
  - SECTIONs (of COURSEs)
  - (academic) DEPARTMENTs
  - INSTRUCTORs

Example of a Database

- 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

- Entities and relationships are typically expressed in a conceptual data model.

Example of a simple database

COURSE

Course_name	Course_number	Credit_hours	Department
Intro to Computer Science	CS1310	4	CS
Data Structures	CS3320	4	CS
Discrete Mathematics	MATH2410	3	MATH
Database	CS3380	3	CS

SECTION

Section_identifier	Course_number	Semester	Year	Instructor
85	MATH2410	Fall	04	King
92	CS1310	Fall	04	Anderson
102	CS3320	Spring	05	Knuth
112	MATH2410	Fall	05	Chang
119	CS1310	Fall	05	Anderson
135	CS3380	Fall	05	Stone

GRADE_REPORT

Student_number	Section_identifier	Grade
17	112	B
17	119	C
8	85	A
8	92	A
8	102	B
8	135	A

PREREQUISITE

Course_number	Prerequisite_number
CS3380	CS3320
CS3380	MATH2410
CS3320	CS1310

Figure 1.2
A database that stores student and course information.

Main Characteristics of the Database Approach

- Self-describing nature of a database system:
  - A DBMS **catalog** stores the description of a particular database (e.g. data structures, types, and constraints)
  - The description is called **meta-data**.
- Insulation between programs and data:
  - Called **program-data independence**.
  - Allows changing data structures and storage organization without having to change the DBMS access programs.

Example of a simplified database catalog

RELATIONS

Relation_name	No_of_columns
STUDENT	4
COURSE	4
SECTION	5
GRADE_REPORT	3
PREREQUISITE	2

COLUMNS

Column_name	Data_type	Belongs_to_relation
Name	Character (30)	STUDENT
Student_number	Character (4)	STUDENT
Class	Integer (1)	STUDENT
Major	Major_type	STUDENT
Course_name	Character (10)	COURSE
Course_number	XXXXNNNN	COURSE
….	….	…..
….	….	…..
….	….	…..
Prerequisite_number	XXXXNNNN	PREREQUISITE

Note: Major_type is defined as an enumerared type with all known majors. XXXXNNNN is used to define a type with four alpha characters followed by four digits

Figure 1.3
An example of a database catalog for the database in Figure 1.2.

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**.
  - Programs refer to the data model constructs rather than data storage details

- 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 multi-user transaction processing:
  - Allowing a set of **concurrent users** to retrieve from and to update the database.
  - *Concurrency control* within the DBMS guarantees that each **transaction** is correctly executed or aborted
  - *Recovery* subsystem ensures each completed transaction has its effect permanently recorded in the database

Database Users

- Users may be divided into
  - Those who use and control the database content, and those who design, develop and maintain database applications (Actors).
  - Those who design and develop the DBMS software and related tools, and the computer systems operators (Developers).

Database Users

- **Database Administrators (DBA):**
  - Responsible for authorizing access to the database, for coordinating and monitoring its use, acquiring software and hardware resources, controlling its use and monitoring efficiency of operations.

- **Database Designers (DBD):**
  - 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.

Database End Users

- **End-users:** They use the data for queries, reports and some of them update the database content.
  - **Casual**:
    - They access database occasionally when needed
  - **Naïve** or Parametric:
    - They use previously well-defined functions in the form of “canned transactions” against the database.
    - Bank-tellers or reservation clerks are parametric users who do this activity for an entire shift of operations.
    - Social Media Users post and read information from websites

Database 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.
  - Another example is a user that maintains a database of personal photos and videos.

Database Users

- **System Analysts and Application Developers**
  - **System Analysts:**
    - They understand the user requirements and design applications to meet those requirements.
  - **Application Programmers:**
    - Implement the specifications developed by analysts and test and debug them before deployment.
  - **Business Analysts:**
    - There is an increasing need for such people who can analyze vast amounts of business data and real-time data for better decision making related to planning, advertising, marketing etc.

Database Users

- **System Designers and Implementers:**
  - Design and implement DBMS packages in the form of modules and interfaces and test and debug them.
  - The DBMS must interface with applications, language compilers, operating system components, etc.

- **Operators and Maintenance Personnel:**
  - They manage the actual running and maintenance of the database system hardware and software environment.

Advantages of Using the Database Approach

- Controlling **redundancy** in data storage and in development and maintenance efforts.
- **Restricting unauthorized access** to data. Only the DBA staff uses privileged commands and facilities.
- Providing **storage structures** (e.g. indexes) for efficient query processing
- Providing optimization of queries for efficient 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** from the stored data using deductive and active rules and triggers.

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), Web page layouts, 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 current information:
  - Extremely important for on-line transaction systems such as shopping, airline, hotel, car reservations.
- Economies of scale:
  - **Wasteful overlap of resources and personnel** can be avoided by consolidating data and applications across departments.

Historical Development of Database Technology

- Early Database Applications:
  - The Hierarchical and Network Models were introduced in mid 1960s and dominated during the seventies.
  - A bulk of the worldwide database processing still occurs using these models, particularly, the hierarchical model using IBM's IMS system.

- Relational Model based Systems:
  - Relational model was originally introduced in 1970, was heavily researched and experimented within IBM Research and several universities.
  - Relational DBMS Products emerged in the early 1980s.

Historical Development of Database Technology

- Object-oriented and emerging applications:
  - Object-Oriented Database Management Systems (OODBMSs) were introduced in late 1980s and early 1990s to cater to the need of complex data processing in CAD and other applications.
  - Many relational DBMSs have incorporated object database concepts, leading to a new category called *object-relationa*l DBMSs (ORDBMSs)

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 access to data by multiple users is not required.
- When a DBMS may be infeasible:
  - In embedded systems where a general purpose DBMS may not fit in available storage

When not to use a DBMS

- When no DBMS may suffice:
  - If there are stringent real-time requirements that may not be met because of DBMS
  - If the database system is not able to handle the complexity of data because of modeling limitations (e.g., in complex genome and protein databases)
  - If the database users need special operations not supported by the DBMS (e.g., GIS and location based services).