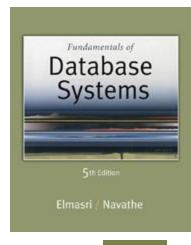


5th Edition

Elmasri / Navathe

Chapter 1

Introduction: Databases and Database Users





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.

Simplified database system environment

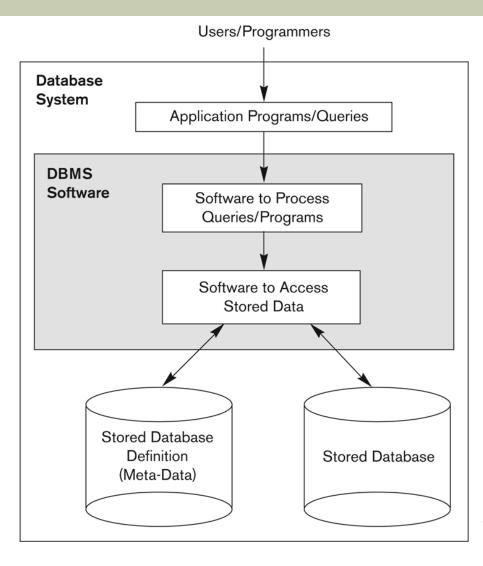


Figure 1.1
A simplified database system environment.

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

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
- Maintaining the database and associated programs over the lifetime of the database application
 - Called database, software, and system maintenance

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

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 entities and relationships are typically expressed in a conceptual data model, such as the ENTITY-RELATIONSHIP data model (see Chapters 3, 4)

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	В
17	119	С
8	85	Α
8	92	Α
8	102	В
8	135	Α

PREREQUISITE

Figure 1.2A database that stores student and course information.

Course_number	Prerequisite_number
CS3380	CS3320
CS3380	MATH2410
CS3320	CS1310

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.
- This allows the DBMS software to work with different database applications.

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

Figure 1.3

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

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

Main Characteristics of the Database Approach (continued)

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 (continued)

- 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
 - OLTP (Online Transaction Processing) is a major part of database applications. This allows hundreds of concurrent transactions to execute per second.

Database Users

- Users may be divided into
 - Those who actually use and control the database content, and those who design, develop and maintain database applications (called "Actors on the Scene"), and
 - Those who design and develop the DBMS software and related tools, and the computer systems operators (called "Workers Behind the Scene").

Database Users

Actors on the scene

Database administrators:

 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:

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.

Categories of End-users

- Actors on the scene (continued)
 - End-users: They use the data for queries, reports and some of them update the database content. End-users can be categorized into:
 - 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 (continued)

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 its own internal database.
- Another example is a user that maintains an address book

Advantages of Using the Database Approach

- Controlling redundancy in data storage and in development and maintenance efforts.
 - Sharing of data among multiple users.
- Restricting unauthorized access to data.
- Providing persistent storage for program Objects
 - In Object-oriented DBMSs see Chapters 20-22
- Providing Storage Structures (e.g. indexes) for efficient Query Processing

Advantages of Using the Database Approach (continued)

- 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

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 (continued)

- 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 airline, hotel, car reservations.
- Economies of scale:
 - Wasteful overlap of resources and personnel can be avoided by consolidating data and applications across departments.

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.