Chapter 1

Introduction: Databases and Database Users
Outline

- Types of Databases and Database Applications
- Basic Definitions
- Typical DBMS Functionality
- Example of a Database (UNIVERSITY)
- Main Characteristics of the Database Approach
- Database Users
- Advantages of Using the Database Approach
- 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)
  - Data Warehouses
  - Real-time and Active Databases
  - Many other applications

- First part of book focuses on traditional applications

- A number of recent applications are described later in the book (for example, Chapters 24, 26, 28, 29, 30)
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 |

<table>
<thead>
<tr>
<th>SECTION</th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Section_identifier</td>
<td>Course_number</td>
<td>Semester</td>
<td>Year</td>
<td>Instructor</td>
</tr>
<tr>
<td>85</td>
<td>MATH2410</td>
<td>Fall</td>
<td>04</td>
<td>King</td>
</tr>
<tr>
<td>92</td>
<td>CS1310</td>
<td>Fall</td>
<td>04</td>
<td>Anderson</td>
</tr>
<tr>
<td>102</td>
<td>CS3320</td>
<td>Spring</td>
<td>05</td>
<td>Knuth</td>
</tr>
<tr>
<td>112</td>
<td>MATH2410</td>
<td>Fall</td>
<td>05</td>
<td>Chang</td>
</tr>
<tr>
<td>119</td>
<td>CS1310</td>
<td>Fall</td>
<td>05</td>
<td>Anderson</td>
</tr>
<tr>
<td>135</td>
<td>CS3380</td>
<td>Fall</td>
<td>05</td>
<td>Stone</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>GRADE REPORT</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Student_number</td>
<td>Section_identifier</td>
<td>Grade</td>
</tr>
<tr>
<td>17</td>
<td>112</td>
<td>B</td>
</tr>
<tr>
<td>17</td>
<td>119</td>
<td>C</td>
</tr>
<tr>
<td>8</td>
<td>85</td>
<td>A</td>
</tr>
<tr>
<td>8</td>
<td>92</td>
<td>A</td>
</tr>
<tr>
<td>8</td>
<td>102</td>
<td>B</td>
</tr>
<tr>
<td>8</td>
<td>135</td>
<td>A</td>
</tr>
</tbody>
</table>

**Figure 1.2**
A database that stores student and course information.

<table>
<thead>
<tr>
<th>PREREQUISITE</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Course_number</td>
<td>Prerequisite_number</td>
<td></td>
</tr>
<tr>
<td>CS3380</td>
<td>CS3320</td>
<td></td>
</tr>
<tr>
<td>CS3380</td>
<td>MATH2410</td>
<td></td>
</tr>
<tr>
<td>CS3320</td>
<td>CS1310</td>
<td></td>
</tr>
</tbody>
</table>
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

<table>
<thead>
<tr>
<th>Relation_name</th>
<th>No_of_columns</th>
</tr>
</thead>
<tbody>
<tr>
<td>STUDENT</td>
<td>4</td>
</tr>
<tr>
<td>COURSE</td>
<td>4</td>
</tr>
<tr>
<td>SECTION</td>
<td>5</td>
</tr>
<tr>
<td>GRADE_REPORT</td>
<td>3</td>
</tr>
<tr>
<td>PREREQUISITE</td>
<td>2</td>
</tr>
</tbody>
</table>

### COLUMNS

<table>
<thead>
<tr>
<th>Column_name</th>
<th>Data_type</th>
<th>Belongs_to_relation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Character (30)</td>
<td>STUDENT</td>
</tr>
<tr>
<td>Student_number</td>
<td>Character (4)</td>
<td>STUDENT</td>
</tr>
<tr>
<td>Class</td>
<td>Integer (1)</td>
<td>STUDENT</td>
</tr>
<tr>
<td>Major</td>
<td>Major_type</td>
<td>STUDENT</td>
</tr>
<tr>
<td>Course_name</td>
<td>Character (10)</td>
<td>COURSE</td>
</tr>
<tr>
<td>Course_number</td>
<td>XXXXNNNN</td>
<td>COURSE</td>
</tr>
<tr>
<td></td>
<td>....</td>
<td>.....</td>
</tr>
<tr>
<td></td>
<td>....</td>
<td>.....</td>
</tr>
<tr>
<td></td>
<td>....</td>
<td>.....</td>
</tr>
<tr>
<td>Prerequisite_number</td>
<td>XXXXNNNN</td>
<td>PREREQUISITE</td>
</tr>
</tbody>
</table>

*Note: Major_type is defined as an enumerated type with all known majors. XXXXNNNNN 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.
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.

- **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.
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.
  - Their use has not taken off much.
- Many relational DBMSs have incorporated object database concepts, leading to a new category called object-relational DBMSs (ORDBMSs)
- *Extended relational* systems add further capabilities (e.g. for multimedia data, XML, and other data types)
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). (see Ch. 27).
- Script programming languages such as PHP and JavaScript allow generation of dynamic Web pages that are partially generated from a database (see Ch. 26).
  - Also allow database updates through Web pages
Extending Database Capabilities

- New functionality is being added to DBMSs in the following areas:
  - Scientific Applications
  - XML (eXtensible Markup Language)
  - Image Storage and Management
  - Audio and Video Data Management
  - Data Warehousing and 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.
Summary

- Types of Databases and Database Applications
- Basic Definitions
- Typical DBMS Functionality
- Example of a Database (UNIVERSITY)
- Main Characteristics of the Database Approach
- Database Users
- Advantages of Using the Database Approach
- When Not to Use Databases