Module 1
Database Systems Concepts and Architecture

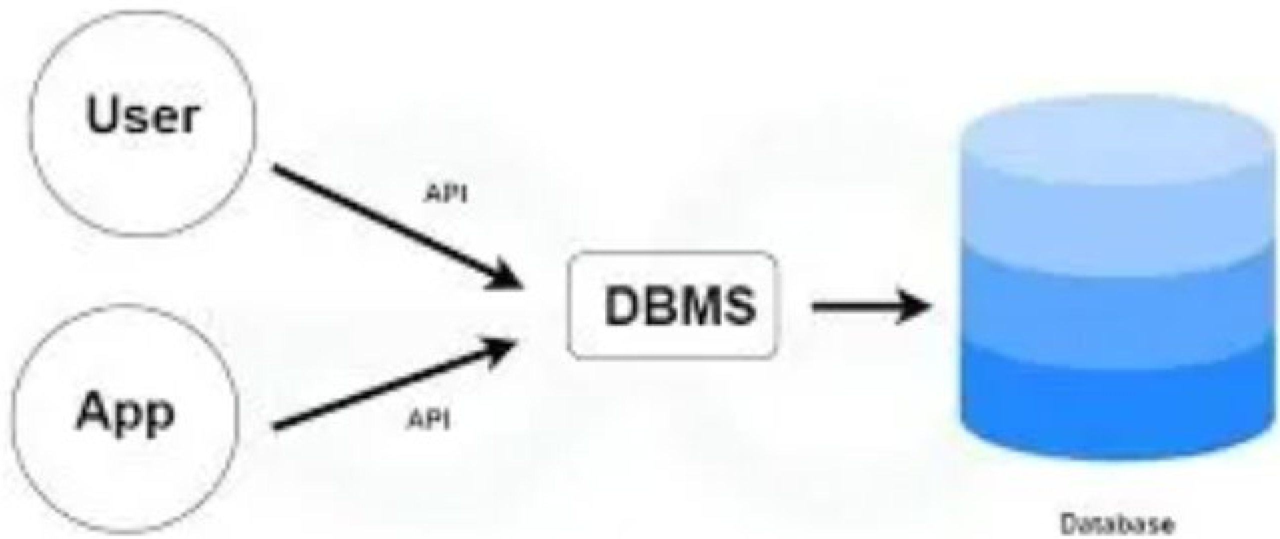
Need for database systems – Characteristics of Database Approach – Advantages of using DBMS approach - Actors on the Database Management Scene: Database Administrator - Classification of database management systems- Data Models - Schemasand Instances - Three-Schema Architecture - The Database System Environment -Centralized and Client/Server Architectures for DBMSs – Overall Architecture of Database Management Systems

Reference :

- ***R. Elmasri & S. B. Navathe, Fundamentals of Database Systems, Addison Wesley, 7th Edition, 2016***
- ***A. Silberschatz, H. F. Korth & S. Sudarshan, Database System Concepts, McGraw Hill,7th Edition 2019.***

Introduction of DBMS (Database Management System)

- A Database Management System (DBMS) is a software solution designed to efficiently manage organize and retrieve data in a structured manner.



Popular database management systems (DBMS)

- Oracle Database
- MySQL
- Microsoft SQL Server
- PostgreSQL
- MongoDB
- IBM Db2
- Amazon Aurora
- MariaDB
- SQLite
- Teradata
- SAP HANA
- Informix
- Redis
- Apache Cassandra

Front-End Technologies

- These are used to build the **client-side** (what users interact with in the browser).

Technology	Description
HTML	The standard markup language used to create the structure of web pages.
CSS	Stylesheet language used to describe the presentation of web pages, including layout, colors, and fonts.
JavaScript	A programming language that adds interactivity and dynamic behavior to web pages.

Frameworks & Libraries

Framework /Library	Description
React.js	A popular JavaScript library for building user interfaces, especially for single-page applications (SPA). Developed by Facebook, React allows for the creation of reusable UI components.
Angular	A TypeScript-based open-source framework developed by Google for building dynamic, single-page web applications. It uses a component-based architecture and provides tools for routing, state management, and more.
Vue.js	A progressive JavaScript framework for building user interfaces. Vue is easy to integrate with other projects and libraries, and it is designed to be flexible and incrementally adoptable.
Svelte	A compiler that generates efficient JavaScript code for building user interfaces. Unlike other frameworks, Svelte shifts much of the work to compile time, resulting in faster performance at runtime.
	A minimal JavaScript framework for composing

UI Libraries / CSS Frameworks

Framework /Library	Description
Bootstrap	Popular CSS framework with grid system and pre-designed UI components. Ideal for rapid development and responsive sites.
Tailwind CSS	Utility-first CSS framework allowing for custom designs with utility classes. Flexible and highly customizable.
Material-UI (MUI)	React-based UI framework following Material Design guidelines with customizable components.
Bulma	Flexbox-based CSS framework with simple, responsive design and easy-to-use components.
Foundation	Highly customizable, responsive framework for scalable and accessible web applications.

Back-end technologies

Technology	Short Description
Java (Spring Boot)	Java is widely used in enterprise apps, with Spring Boot simplifying microservice development.
Python (Django, Flask)	Python is versatile; Django is full-stack, Flask is lightweight for small apps, and FastAPI excels for APIs.
JavaScript (Node.js, Express.js)	Node.js allows server-side JS, while Express.js is a minimalist framework for APIs and web apps.
PHP (Laravel)	PHP is a server-side language, and Laravel provides a modern, elegant framework for web applications.
C# (.NET Core)	C# with .NET Core is used for scalable, cross-platform web apps and APIs, popular in enterprise environments.

Back-end technologies

Framework /Library	Description
Bootstrap	Popular CSS framework with grid system and pre-designed UI components. Ideal for rapid development and responsive sites.
Tailwind CSS	Utility-first CSS framework allowing for custom designs with utility classes. Flexible and highly customizable.
Material-UI (MUI)	React-based UI framework following Material Design guidelines with customizable components.
Bulma	Flexbox-based CSS framework with simple, responsive design and easy-to-use components.
Foundation	Highly customizable, responsive framework for scalable and accessible web applications.

Relational Databases (SQL)

Database	Short Description
MySQL	Open-source relational database, known for its speed and reliability. Widely used in web applications.
PostgreSQL	Advanced open-source database with strong support for SQL and ACID compliance, often used for complex applications.
SQLite	Lightweight, serverless database used for embedded systems or local storage, often used in mobile apps.
MariaDB	Open-source fork of MySQL, designed to be fully compatible while adding more features and optimizations.
Microsoft SQL Server	A relational database management system (RDBMS) from Microsoft, known for integration with other Microsoft products.
Oracle Database	Enterprise-level RDBMS known for handling large datasets, high availability, and complex enterprise applications.

NoSQL Databases

Database	Short Description
MongoDB	A document-oriented database, known for its flexibility and scalability, commonly used for large-scale web apps.
Redis	An in-memory data structure store, often used for caching, real-time analytics, and session management.
Cassandra	A highly scalable, distributed database designed for handling large amounts of data across multiple nodes with no downtime.
CouchDB	A document-based database that uses a schema-free model, designed for reliability and easy replication across nodes.
Firebase Realtime Database	A cloud-hosted NoSQL database from Firebase, designed for real-time syncing and building mobile and web apps.
DynamoDB (AWS)	A fully managed NoSQL database from AWS, designed for high availability, scalability, and low-latency performance

Web Servers

Web Server	Short Description
Apache	A highly configurable and widely-used open-source web server known for its stability and rich module ecosystem.
Nginx	A lightweight, high-performance web server and reverse proxy server, often used for load balancing and static content delivery.
Tomcat	An open-source application server for running Java-based web applications, especially Servlets and JSPs.
Jetty	A lightweight and fast Java web server and servlet container, often used in embedded systems and microservices.
Caddy	A modern web server with automatic HTTPS, easy configuration, and support for reverse proxying and load balancing.

DBMS

Feature	File System (Traditional System)	DBMS
Data organization	Data is organized as a hierarchical tree structure of files and directories.	Data is organized into tables and rows, with relationships established between tables.
Data redundancy	Data redundancy is high, as each file is stored in multiple directories.	Data redundancy is low, as data is stored in a centralized location and is not repeated.
Data Integrity	Data integrity is low, as there is no built-in mechanism for enforcing data constraints.	Data integrity is high, as constraints can be set on data to ensure accuracy and consistency.
Scalability	Scalability is limited, as the file system becomes slow and unwieldy as the number of files increases.	Scalability is good, as databases can handle large amounts of data and can be optimized for performance.

DBMS

Feature	File System	DBMS
Security	Security is basic, with limited control over who can access files.	Security is robust, with fine-grained control over who can access and modify data.
Query capabilities	Query capabilities are limited, with no support for complex queries or transactions.	Query capabilities are strong, with support for complex queries and transactions.
Data sharing	Data sharing is difficult, as multiple users must access the same physical file.	Data sharing is easy, as multiple users can access the same data simultaneously through the database management system.
Backup and recovery	Backup and recovery is time-consuming and difficult, as each file must be backed up individually.	Backup and recovery is straightforward, as the database management system handles backup and recovery operations.

Impact and Importance of Databases

- **Databases have greatly influenced** the widespread use of computers.
- **Critical role** in nearly all fields involving computer usage, including:
- Business
- E-commerce
- Social media
- Engineering
- Medicine
- Genetics
- Law
- Education
- Library science

Impact and Importance of Databases

Key Features of DBMS

- Data Modeling: Tools to create and modify data models, defining the structure and relationships within the database.
- Data Storage and Retrieval: Efficient mechanisms for storing data and executing queries to retrieve it quickly.
- Concurrency Control: Ensures multiple users can access the database simultaneously without conflicts.
- Data Integrity and Security: Enforces rules to maintain accurate and secure data, including access controls and encryption.
- Backup and Recovery: Protects data with regular backups and enables recovery in case of system failures.

What is a Database?

- A **database** is defined as a **collection of related data**.
- **Data** refers to known facts that:
- Can be **recorded**
- Have **implicit meaning**

Examples of Databases in Daily Life

•**Mobile phones** store data like names, phone numbers, and addresses.

•These are often managed by simple built-in **database software**.

•Similar data can be stored:

•In an **indexed address book**

•On a **computer hard drive** using software like **Microsoft Access** or **Excel**

Implicit Properties of a Database

1. **Represents a Real-World Aspect (Miniworld/UoD)**
   Changes in the real world are reflected in the database.
2. **Logically Coherent with Meaning**
   Random data is **not** a database; data must be **related** and **meaningful.**
3. **Purpose-Driven Design**
   Built for a **specific purpose**, with a clear audience and intended applications.

Why Accurate Representation Matters

- ✓ Real-world events (e.g., purchases, births) must be **quickly updated** in the database to ensure accuracy and reliability.
- ✓ A database should always reflect the **current state** of its miniworld.

Size and Complexity Can Vary

- ✓ **Small:** Address book with a few hundred entries.
- ✓ **Medium:** Library catalog with ~500,000 records.
- ✓ **Large:**
- **Facebook**: Tracks billions of users, relationships, posts, and privacy settings.
- **Amazon**: Stores data on 60+ million users and millions of items across 42+ terabytes of data

Types of Databases

- **Manual databases**: e.g., a library card catalog.
- **Computerized databases**: Managed by:

  Custom programs

  or

  A Database Management System (DBMS).

What is a DBMS (Database Management System)?

- Software that allows users to **define, build, manipulate, and share** a database.
- Handles:
- **Defining**: Data types, structures, and constraints.
- **Constructing**: Storing data on physical storage.
- **Manipulating**: Querying, updating, and reporting.
- **Sharing**: Multiple users/programs can access simultaneously.
- **Protecting**: Against hardware/software failures and unauthorized access.
- **Maintaining**: Supports long-term evolution of the database.

DBMS Components

- **Meta-data**: Descriptive data stored in a **catalog** or **dictionary**.
- **Queries:** Retrieve specific data.
- **Transactions:** Involve both reading and writing data.

General vs. Special-Purpose DBMS

- **General-purpose DBMS**: Can be used for many applications.
- **Custom-built DBMS**: Tailored for a specific task (e.g., airline reservations).

What is data, database, DBMS ?

- **Data:** Known facts that can be recorded and have an implicit meaning; raw
- **Database:** a highly organized, interrelated, and structured set of data about a particular enterprise
  - Controlled by a database management system (DBMS)
- **DBMS**
  - Set of programs to access the data
  - An environment that is both convenient and efficient to use
- Database systems are used to manage collections of data that are:
  - Highly valuable
  - Relatively large
  - Accessed by multiple users and applications, often at the same time.
- A **modern database system** is a complex software system whose task is to manage a large, complex collection of data.
- Databases touch all aspects of our lives

Database system environment

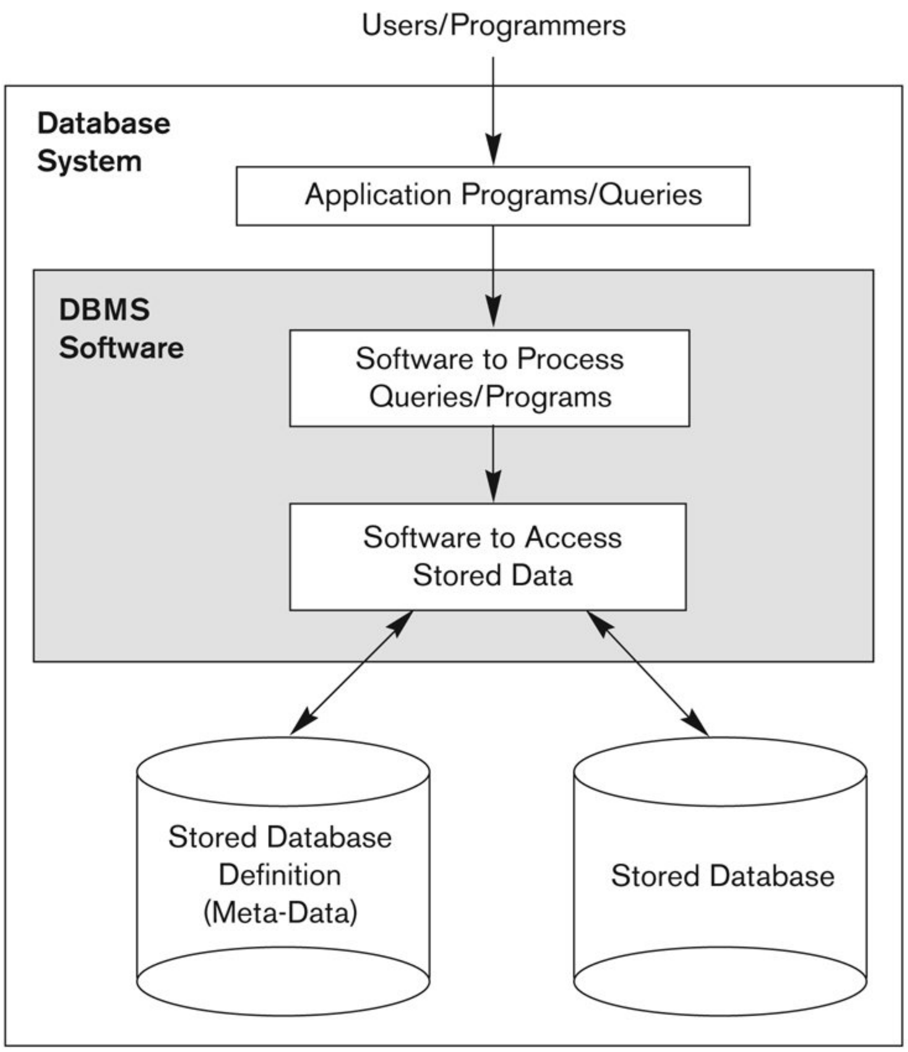


Figure 1.1
A simplified database system environment.

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

These informal queries and updates must be precisely specified in the query language of the DBMS (Database Management System) before the system can process them.

Examples of Queries (Retrieving Data)

These are requests to extract specific information from the database:

•**Retrieve the transcript** : a list of all **courses and grades** of **'Smith'**.
•**List the names of students** who took the **section** of the **'Database' course** offered in **fall 2008**, and their **grades** in that section.
•**List the prerequisites** of the **'Database' course**.

Examples of Updates (Modifying Data)

These are operations that change data stored in the database:

•**Change the class** of **'Smith'** to **sophomore**.
•**Create a new section** for the **'Database' course** for **this semester**.
•**Enter a grade of 'A'** for **'Smith'** in the **'Database' section** of **last semester**.

Database structure & few sample data records

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.

Steps in Database Design and Development

1. Requirements Specification and Analysis

- First phase in designing a **new application** or **new database**.
- Involves identifying and documenting all the data and functionality needs.

2. Conceptual Design

- The documented requirements are transformed into a **conceptual model**.
- This is typically done using tools like the **Entity-Relationship (ER) model**.
- Conceptual design is:
    - Easy to understand
    - Maintainable and modifiable

3. Logical Design (relational DBMS)

- The conceptual model is translated into a **logical data model**.
- This model is suitable for implementation in a **commercial DBMS**.

4. Physical Design

- Final stage where technical details are added for:
    - **Storage methods**
    - **Access paths**
- The database is **implemented**, **populated with actual data**, and **continuously maintained**.

Key Characteristics of the Database Approach

1. Self-Describing Nature of a Database System

- A **database system** stores not only the actual data but also a **complete description of the database structure** and constraints.
- This descriptive information is stored in the **DBMS catalog** as **meta-data**, which includes:
- ✓ File structures
- ✓ Data item types and formats
- ✓ Constraints
- The **DBMS software** and users refer to this catalog to understand how to access and interpret the data.
- This allows **general-purpose DBMS software** to work with **different databases** (e.g., university, banking, company databases) without being rewritten.
- In contrast, **traditional file systems** embed structure within application programs, requiring rewrites when data structures change.
- Some newer systems, like **NOSQL databases**, use **self-describing data formats** where data and structure are stored together.

Key Characteristics of the Database Approach

1. Self-Describing Nature of a Database System

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

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

Key Characteristics of the Database Approach

2. Insulation Between Programs and Data, and Data Abstraction

Program-Data Independence

•In traditional systems, changes to data structure (e.g., adding Birth_date to a STUDENT record) require **modifying all dependent programs**.

•In a **DBMS**, structure is stored in the **catalog**, so programs continue to work after changes—this is **program-data independence**.

Program-Operation Independence

•In **object-oriented** and **object-relational databases**, operations (methods) on data are defined with:

- ✓ An **interface** (name and parameter types)
- ✓ An **implementation** (actual logic)

•Applications invoke operations by interface, regardless of how they are implemented—this is **program-operation independence**.

Data Abstraction

•A **DBMS provides a conceptual view** of the data, hiding physical storage details.

•Users interact with logical entities like STUDENT.Name, without needing to know **byte positions or storage paths**.

•This abstraction is achieved using **data models** (e.g., relational, object-based), which define objects, properties, and relationships.

•In object-oriented models, even **behavior (operations)** like CALCULATE_GPA can be abstracted.

Key Characteristics of the Database Approach

2. Insulation Between Programs and Data, and Data Abstraction

Data Item Name	Starting Position in Record	Length in Characters (bytes)
Name	1	30
Student_number	31	4
Class	35	1
Major	36	4

Figure 1.4
Internal storage format for a STUDENT record, based on the database catalog in Figure 1.3.

Key Characteristics of the Database Approach

3. Support of Multiple Views of the Data

- A single database may have **many types of users**, each requiring a different **view** of the data.
- A **view** can be:
  - ✓ A **subset** of the database
  - ✓ A **virtual table** derived from base data (not stored physically)
- For example:
  - ✓ A user printing transcripts may only need student names and grades.
  - ✓ Another user may need to check if students met all **prerequisites** for a course.
- Views improve:
  - ✓ **Security** (limit access to sensitive data)
  - ✓ **User convenience** (simplified interfaces)
  - ✓ **Application customization**

Key Characteristics of the Database Approach

3. Support of Multiple Views of the Data

Figure 1.5(a). A second user, who is interested only in checking that students have taken all the prerequisites of each course for which the student registers, may require the view shown in Figure 1.5(b).

TRANSCRIPT

Student_name	Student_transcript				
	Course_number	Grade	Semester	Year	Section_id
Smith	CS1310	C	Fall	08	119
	MATH2410	B	Fall	08	112
Brown	MATH2410	A	Fall	07	85
	CS1310	A	Fall	07	92
	CS3320	B	Spring	08	102
	CS3380	A	Fall	08	135

(a)

COURSE_PREREQUISITES

Course_name	Course_number	Prerequisites
Database	CS3380	CS3320
		MATH2410
Data Structures	CS3320	CS1310

(b)

Figure 1.5
Two views derived from the database in Figure 1.2. (a) The TRANSCRIPT view. (b) The COURSE_PREREQUISITES view.

Key Characteristics of the Database Approach

4. Sharing of Data and Multiuser Transaction Processing

✓A **multiuser DBMS** supports **simultaneous data access** by many users/applications.

✓This avoids **data duplication** and ensures **data consistency** through controlled access.

Concurrency Control

R

✓equired when multiple users access or modify data at the same time.

✓Example: Multiple agents booking seats on the same flight—the D**BMS ensures** one seat is assigned to only one passenger.

✓

Tr*ansaction Management*

A

✓tr**ansaction is** a logical unit of work involving database operations (e.g., read, write).

✓Key transaction properties:

✓At

- ❖ **omicity:** All operations in a transaction are completed or none are.
- ❖ Is**olation:** Transactions appear to run independently, even when concurrent.
- ❖

OLTP (Online Transaction Processing)

Actors on the Scene

In large organizations, a wide range of people interact with a database system—each with different roles, responsibilities, and levels of technical knowledge. These people are known as **actors on the scene** because they are actively involved in the database's operation and use.

Actors on the Scene

1. Database Administrators (DBAs)

Manage the database and DBMS software.

Responsibilities:

- ✓ Authorize user access.
- ✓ Monitor and coordinate database usage.
- ✓ Acquire hardware and software resources.
- ✓ Ensure **security**, **data integrity**, and **performance**.

✓In large organizations, DBAs may lead a **technical team** handling these tasks.

Actors on the Scene

2. Database Designers

Design the overall database structure **before** implementation.

Key tasks:

- Identify what data needs to be stored.
- Choose appropriate **data structures**.
- Communicate with **end users** to gather requirements.
- Create and integrate **different user views** into a unified database design.

Often work under the DBA’s team.

Actors on the Scene

3. End Users

End users interact directly with the database. They fall into several categories:

- **Casual End Users**
  - Use the system **occasionally**.
  - Run **ad-hoc queries** using a **query interface**.
  - Typically mid- or upper-level managers.
- **Naive or Parametric End Users**
  - Use **predefined, repeated transactions** (called **canned transactions**).
  - Have minimal DBMS knowledge; interact through **apps or forms**.
  - **Examples:**
    - Bank tellers and customers checking balances or making deposits.
    - Reservation agents booking flights or hotel rooms.
    - Employees scanning packages in logistics.
    - Social media users posting and reading content.
-

Actors on the Scene

3. End Users

- **Sophisticated End Users**

✓Includes **engineers, scientists, business analysts**.
✓Use advanced DBMS features to write complex queries or develop custom applications

▪**Standalone Users**

✓Maintain **personal databases** using **ready-made software** (e.g., financial tools).
✓Typically use **menu-driven or graphical interfaces**.
✓Become proficient in a specific software package.

Actors on the Scene

4. System Analysts and Application Programmers (Software Engineers)

- **System Analysts**:
  - ✓ Identify **end user requirements**, especially for **naive users**.
  - ✓ Design **standardized canned transactions** for frequent use.
- **Application Programmers**:
  - ✓ Implement, test, debug, and maintain the programs developed by analysts.
  - ✓ Must understand the full capabilities of the **DBMS** to ensure optimal integration.

Workers Behind the Scene

A group of professionals who work **behind the scenes** to develop and support the database system environment. These individuals do **not focus on the data itself**, but on ensuring the DBMS runs efficiently and reliably.

Workers Behind the Scene

1. DBMS System Designers and Implementers

✓Design and build the **core DBMS software** as a complete package.

✓Develop internal modules such as:

- **Catalog management**
- **Query processing**
- **User interfaces**
- **Data access and buffering**
- **Concurrency control**
- **Recovery and security**

✓Ensure smooth integration with:

- **Operating systems**
- **Programming language compilers**

A DBMS is a **complex software system**, and these professionals are crucial in constructing its functionality.

Workers Behind the Scene

2. Tool Developers

✓Create **auxiliary software tools** that enhance DBMS design, usability, and performance

✓Tools may include:

- ❖ **Database design tools**
- ❖ **Performance monitoring utilities**
- ❖ **Natural language or graphical interfaces**
- ❖ **Simulation or prototyping tools**
- ❖ **Test data generators**

These tools are often **developed by third-party vendors** and may be purchased as **optional add-ons**.

Workers Behind the Scene

3. Operators and Maintenance Personnel (System Admins)

✓Ensure the **hardware and software infrastructure** supporting the DBMS is operational.

✓**Responsibilities:**

- ❖ Install, run, and maintain database-related systems.
- ❖ Monitor system health and performance.
- ❖ Apply software updates and backups.

They maintain the **technical environment** for smooth DBMS operation but do not interact with the data itself.

Advantages of Using the DBMS Approach

In addition to the four primary characteristics of database systems (Section 1.3), a well-designed DBMS provides numerous benefits that enhance usability, security, efficiency, and scalability.

1. Controlling Redundancy

Traditional systems store the same data in multiple files across departments, causing:

- ❖ **Duplication of effort**
- ❖ **Wasted storage**
- ❖ **Inconsistencies**

✓DBMS supports **data normalization** to reduce redundancy and ensure consistency.

✓In some cases, **controlled redundancy** (e.g., denormalization) is used to **improve performance**, and the DBMS enforces consistency across copies.

Advantages of Using the DBMS Approach

2. Restricting Unauthorized Access

✓Different users require **different levels of access**.
✓DBMS provides a **security and authorization subsystem** that:

- ❖ Creates **user accounts and passwords**
- ❖ Specifies **access rights** (e.g., read-only or read-write)

✓Access can also be restricted **at the software level** (e.g., only DBAs can run certain programs).

3. Providing Persistent Storage for Program Objects

✓DBMS stores **complex objects (e.g., C++ or Java classes)** persistently.
✓This eliminates the need for manual serialization and deserialization.
✓Solves the **impedance mismatch** problem between programming languages and data storage.

Advantages of Using the DBMS Approach

4. Efficient Query Processing

✓A DBMS must support **fast access** to data stored on disk.

✓It uses:

- ❖ **Indexes** (e.g., B-trees, hash structures)
- ❖ **Caching/buffering**
- ❖ **Query optimization modules**

✓Helps retrieve and process queries **quickly and efficiently**.

5. Backup and Recovery

❖The **backup and recovery subsystem** restores the database in case of:

- ❖ System failures
- ❖ Crashes during transactions
- ❖ Catastrophic hardware issues (e.g., disk failure)

✓Ensures **data integrity** and **business continuity.**

Advantages of Using the DBMS Approach

6. Providing Multiple User Interfaces

✓Different users have different needs:

- ❖ **Casual users**: query languages (e.g., SQL)
- ❖ **Programmers**: programming language APIs
- ❖ **Parametric users**: forms or canned transactions
- ❖ **Standalone/mobile users**: GUI apps or web interfaces

✓Modern DBMSs offer **GUIs and web-based interfaces**.

7. Representing Complex Relationships Among Data

✓Data can be related in many-to-many or hierarchical ways (e.g., students ↔ grades ↔ courses).

✓DBMS can:

- ❖ Represent these **relationships explicitly**
- ❖ Enable **easy querying and updating** of related data

Advantages of Using the DBMS Approach

8. Enforcing Integrity Constraints

✓Enforces **business rules** and **data validity**, such as:

- ❖ **Data types** (e.g., name = string, class = integer 1–5)
- ❖ **Key constraints** (e.g., unique course numbers)
- ❖ **Referential integrity** (e.g., foreign keys linking student to grade)

✓Helps prevent **invalid or inconsistent data**.

9. Supporting Inferencing and Triggers

✓Some systems support **deductive rules** to infer new data automatically.

✓**Triggers** can execute actions in response to certain changes in data.

✓**Stored procedures** and **active rules** automate tasks and maintain consistency.

10. Additional Organizational Benefits

✓**Enforcing Standards**: Consistent naming, formatting, and data usage across departments.

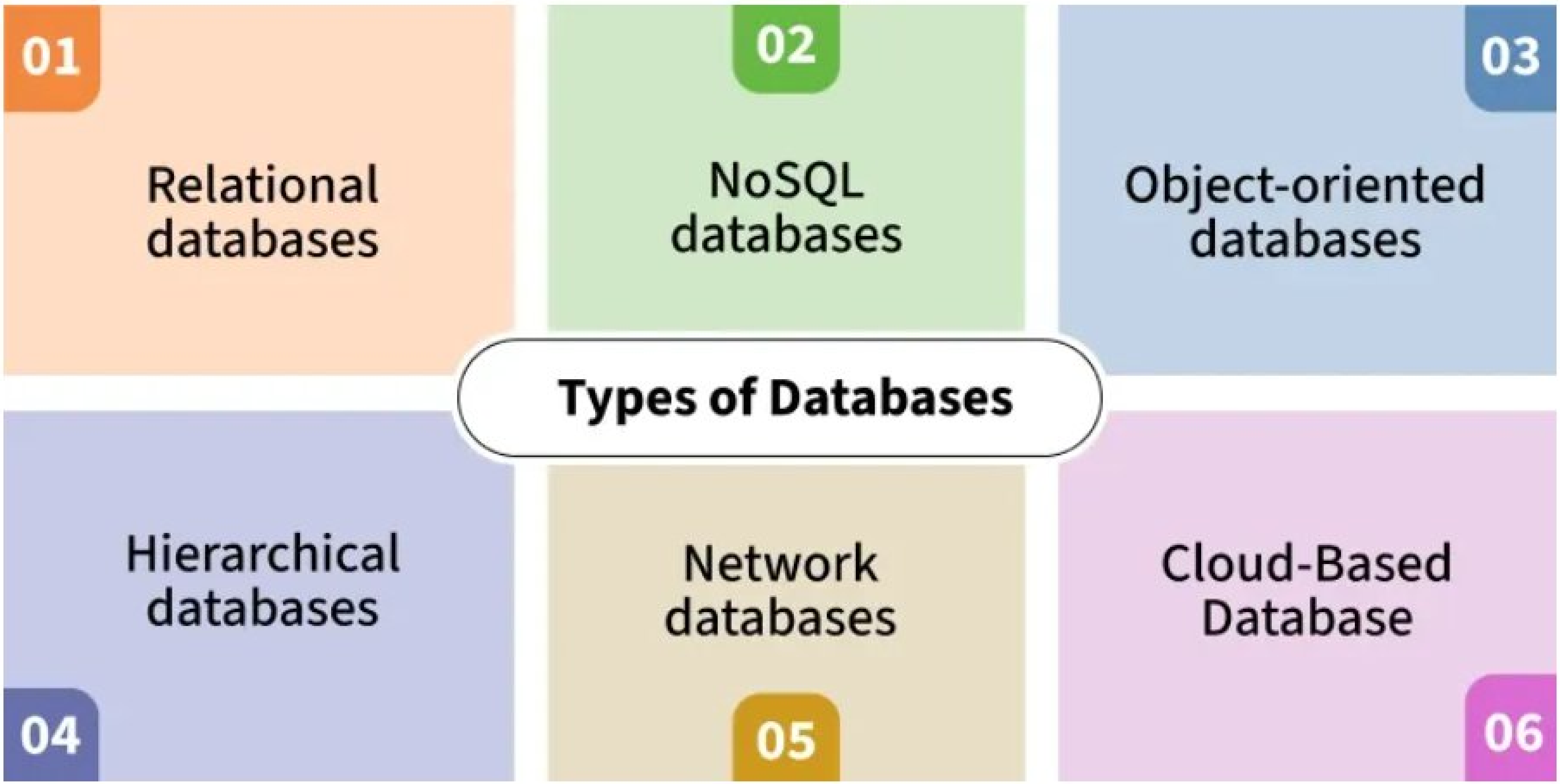
✓**Reduced Development Time**: Faster application development using reusable DBMS facilities.

✓**Flexibility**: Schema changes can be made with **minimal disruption**.

✓**Real-Time Updates**: Users access **up-to-date data instantly**.

✓**Economies of Scale:** Centralized resources reduce overall costs and prevent

Types of DBMS



Types of DBMS