Unit # 1
Introduction to DBMS
(Week-1)

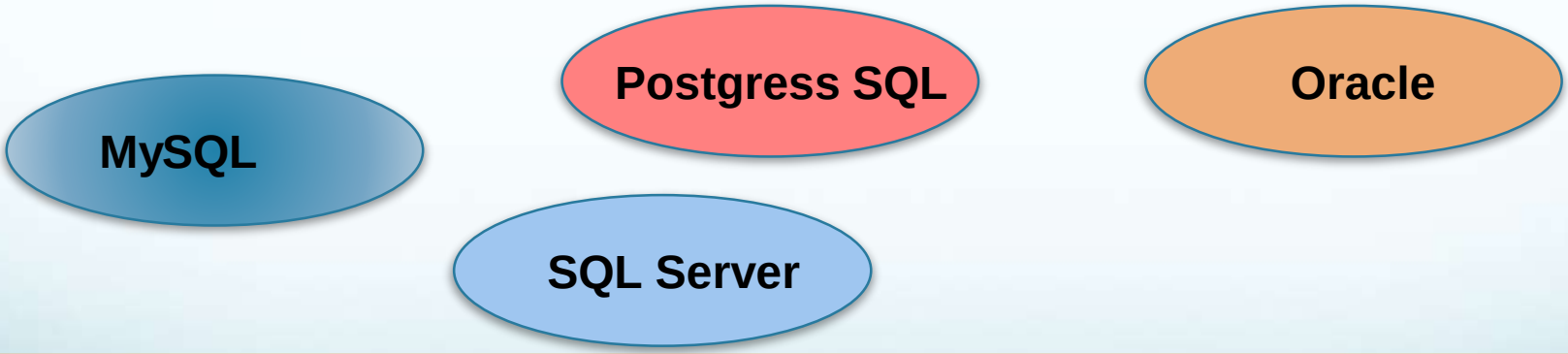
Introduction to DBMS

Data :

Data refers to any collection of facts, figures, or statistics that can be used for analysis, decision-making, or other purposes. It can be in various forms such as numbers, text, images, or videos.

Database : " A database is a structured collection of **related** data stored and accessed electronically." Few examples ??

- *Banking systems* (e.g., keeping track of account balances).
- *Social media platforms* (e.g., storing user information, posts, and comments).
- *University systems* (e.g., managing student information, courses, grades).



A database management system (DBMS) is a collection of programs that enables users to create and maintain a database.

Database + SW Programs = DBMS

Introduction to DBMS (contd)

- A fundamental characteristic of the database approach is that the database system contains not only the database itself but also a complete definition or description of the database structure and constraints.
- This definition is stored in the system **catalog,** which contains information such as the structure of each file, the type and storage format of each data item, and various constraints on the data.
- The information stored in the catalog is called **meta-data** and it describes the structure of the primary database

- UNIVERSITY database
  - Information concerning students, courses, and grades in a university environment

- **Data records**
  - STUDENT
  - COURSE
  - SECTION
  - GRADE_REPORT
  - PREREQUISITE

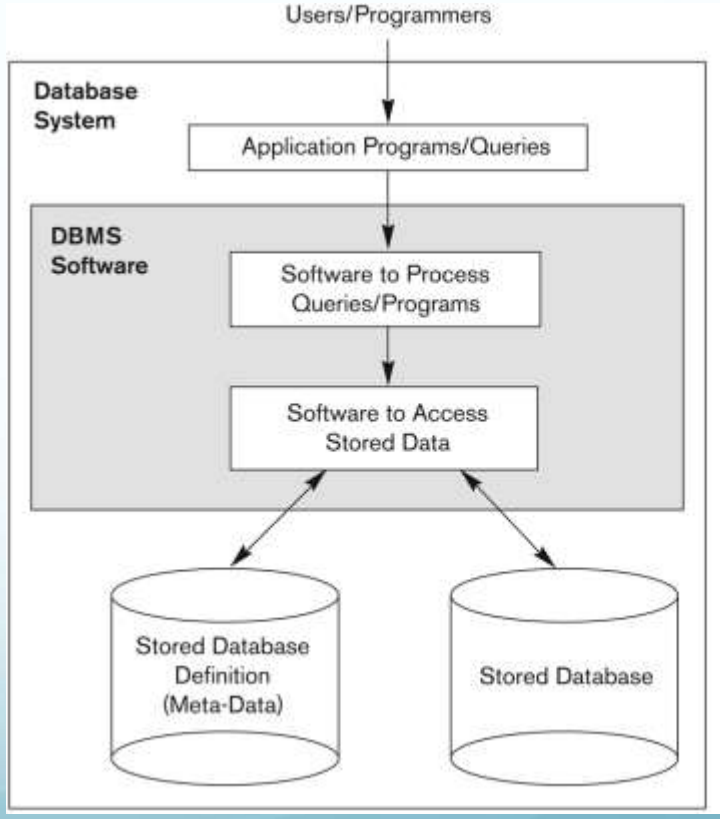


Figure 1
A simplified database system environment

Example (contd)

Structure

Specify structure of records by specifying data type for each data element:

- String of alphabetic characters
- Integer
- Alphanumeric
- Etc.

Construct UNIVERSITY database

- Store data to represent each student, course, section, grade report, and prerequisite as a record in appropriate file
- Relationships among the records
- Manipulation involves querying and updating.
  - Examples of queries:
    - Retrieve the transcript
    - 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:
    - 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

Example (contd)

STUDENT

Name	Student_number	Class	Major
Smith	17	1	CS
Brown	8	2	CS

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	07	King
92	CS1310	Fall	07	Anderson
102	CS3320	Spring	08	Knuth
112	MATH2410	Fall	08	Chang
119	CS1310	Fall	08	Anderson
135	CS3380	Fall	08	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.

Example (contd)

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

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.

Characteristics of DBMS system ?

Database Management System (DBMS) provides a range of features to manage and control data effectively. These features make DBMS an essential tool for modern applications, helping ensure data integrity, security, performance, and accessibility. The key features of a DBMS include:

1. Data Manipulation

DBMS enables users to perform operations such as

- Insertion of new data.
- Updating existing data.
- Deleting data.
- Retrieving data using queries (typically SQL queries).

2. **Data Security**

A DBMS provides security mechanisms to ensure that only authorized users can access or modify the data. This includes:

- *Authentication:* Ensuring that users are who they claim to be (e.g., via passwords).
- *Authorization:* Granting specific permissions to users or roles to access or modify certain parts of the database (e.g., restricting certain users from deleting records).
- *Encryption:* Protecting sensitive data by storing it in encrypted formats.

Characteristics of DBMS system (contd) ?

3. Data Integrity

DBMS ensures the accuracy and consistency of data through rules.

- *Primary Key Constraint:* Ensures each record in a table is unique.
- *Foreign Key Constraint:* Enforces referential integrity between tables.
- *Unique Constraints:* Ensure that certain fields or sets of fields contain unique values.
- *Not Null Constraints:* Ensure that specific fields cannot be left empty.

4. Concurrency Control

DBMS supports simultaneous access to data by multiple users without conflicts. Concurrency control mechanisms ensure that transactions executed concurrently produce the same result as if they were executed serially.

- *Locking mechanisms* (e.g., shared and exclusive locks) are used to ensure that two users cannot update the same data simultaneously.
- *Transaction* isolation levels (like read committed, repeatable read, and serializable) ensure data consistency while allowing multiple transactions to proceed concurrently.

Characteristics of DBMS system (contd) ?

5. Transaction Management

A DBMS ensures that all database operations are executed in a transactional way, following the ACID properties:

- *Atomicity:* A transaction is all-or-nothing; either all its operations are performed, or none of them are.
- *Consistency:* A transaction brings the database from one valid state to another, preserving the integrity of the data.
- *Isolation:* Transactions are executed independently, and intermediate steps are hidden from other transactions.
- *Durability:* Once a transaction is completed, its effects persist even in case of system failures.

6. Back Up & Recovery

DBMS provides tools to create backups of the database and ensure that data can be recovered after a failure (e.g., hardware or software failure, crash)

- Automatic backups and incremental backups help in restoring data to a previous state.
- Recovery mechanisms ensure that the database can recover to a consistent state after unexpected failures, using techniques such as logs, journaling, and check pointing.

Characteristics of DBMS system (contd) ?

7. Data Independence

Data Independence refers to the separation of data from the applications that use it. DBMS provides two types:

- *Logical data independence:* The ability to change the logical schema (e.g., adding new fields to a table) without altering application programs.
- *Physical data independence:* The ability to change the physical storage (e.g., moving data from one server to another) without affecting the logical structure of the database or the applications.

8. Views of Data

DBMS allows the creation of views, which are virtual tables that provide specific users or applications with a customized subset of the database. These views can:
Automatic backups and incremental backups help in restoring data to a previous state.

- Processed outputs such as , aggregated data, totals, averages, or counts, without exposing the detailed records.
- Provide security by restricting access to sensitive data. (Read only access)
- Show only the relevant portions of the data based on user roles.

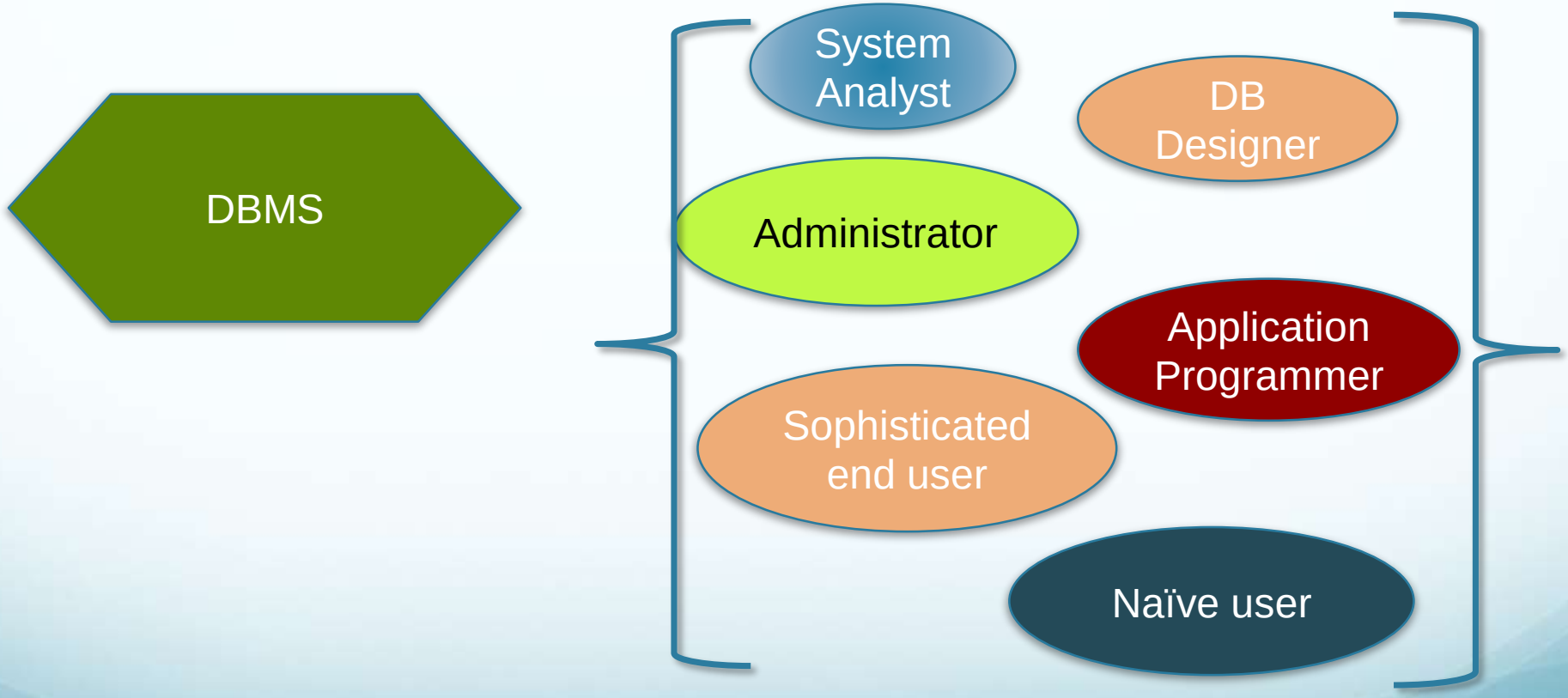
9. Indexing

To speed up the retrieval of data…

10. Data Definition and Data Schema ?? (DDL / DML !!!)

DBMS Players , Users , Actors

In a Database Management System (DBMS), various actors (or roles) are involved in designing, maintaining, managing, and using the database system. Each of these actors plays a crucial role in ensuring the smooth operation and efficient use of the database. The key actors in a DBMS environment include:



DBMS Players , Users , Actors (contd)

1. System Analyst

System analysts bridge the gap between business users and technical teams. They analyze business requirements and ensure that the database and application development meet these requirements:

- Understanding and documenting business processes.
- Translating business needs into technical requirements that the database designers and developers can use.
- Designing user interfaces that interact with the DBMS.
- Working closely with developers and end users to ensure that the system meets both functional and performance goals.

2. Database Designers/Architect

Database designers or architects are responsible for creating the database schema and determining how the data should be structured. Their work occurs during the design phase of the database.

- Designing the conceptual, logical, and physical structure of the database.
- Defining schemas, tables, relationships, and constraints.
- Working closely with developers, DBAs, and end users to understand data and application requirements.
- Developing ER diagrams and normalization strategies to organize data efficiently.

DBMS Players , Users , Actors (contd)

3. Database Administrator

The DBA is responsible for the overall management, control, and maintenance of the DBMS. This is one of the most critical roles as the DBA ensures that the database is functioning correctly, efficiently, and securely.

- Database design and implementation.
- User access control, including creating user roles and granting/restricting permissions.
- Backup and recovery planning to prevent data loss.
- Monitoring performance tuning to ensure the system is running optimally.
- Enforcing data integrity and security policies.
- Ensuring compliance with organizational and legal data requirements.

4. Application Programmer

Application programmers are responsible for developing the software applications that interact with the DBMS to carry out various tasks such as data entry, updating, querying, and reporting.

- Writing SQL queries or using an API to interact with the database.
- Developing the front-end applications that interface with the DBMS (e.g., web applications, mobile apps).
- Writing stored procedures, triggers, and functions in the DBMS to automate tasks.
- Testing and debugging applications to ensure they work correctly with the database.

DBMS Players , Users , Actors (contd)

5. Casual/ Naïve users

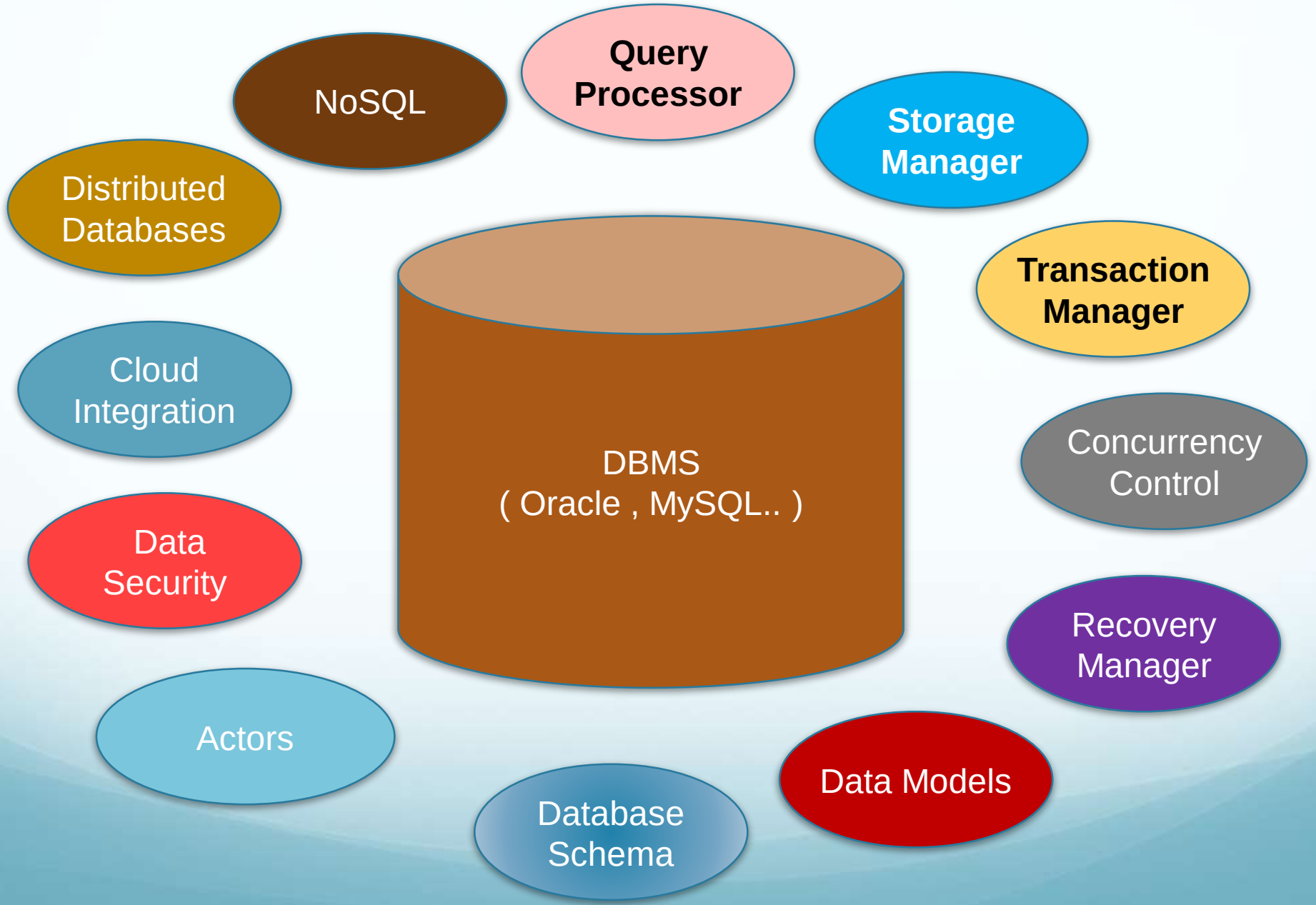
There are different types of end users based on how frequently and in what way they use the system. They can be Casual users or Naïve users.

- Casual users are the individuals who directly interact with the DBMS to perform operations such as querying, updating, and reporting on data. They access the database occasionally , often use ad-hoc queries to retrieve information and do not typically perform updates. Examples: Managers, analysts.
- Naïve users use predefined queries and queries that have been programmed by developers. They are users who typically perform **repetitive tasks**, such as data entry, processing orders, or managing transactions. Examples: Bank tellers, airline reservation clerks.

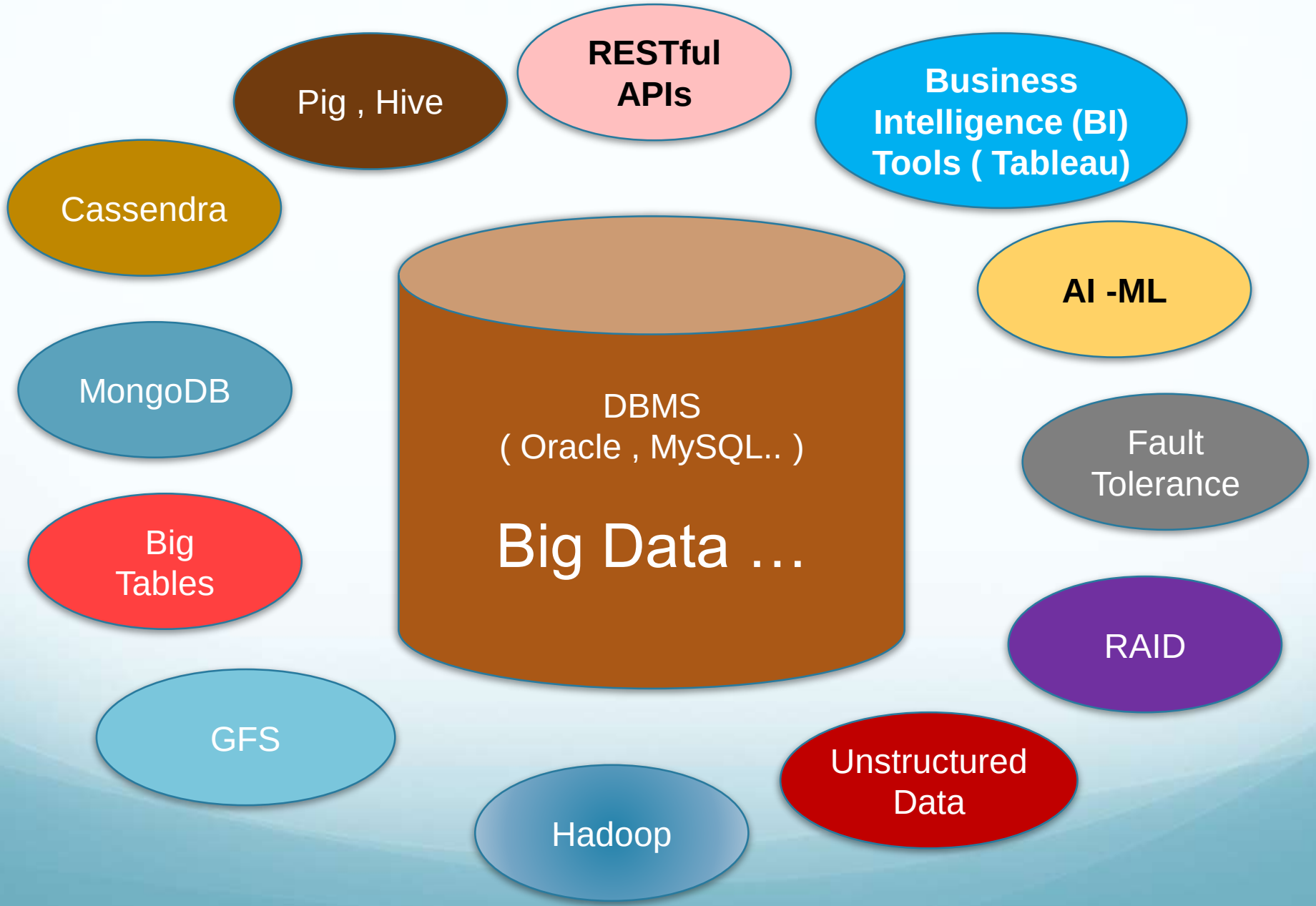
6. Sophisticated users

Users who write complex queries and may interact with the DBMS using advanced tools (e.g., SQL or reporting tools).Typically, these users have a good understanding of the database and may create complex reports or analytics. Examples: Data scientists, business analysts.

DBMS Ecosystem



DBMS Ecosystem … Modern Day picture



DBMS Architecture

The architecture of a Database Management System (DBMS) defines how data is managed, stored, and retrieved in an efficient and organized manner. A DBMS typically follows a Centralized Architecture or Client-Server architecture.

1. Centralized Architecture for DBMS

In a centralized architecture, the entire database system (the DBMS software, storage, and processing) is located on a single server or a set of tightly coupled servers. All data processing and storage are done centrally, and users access the database remotely through terminals or thin clients. This architecture was commonly used in earlier mainframe-based systems and is still relevant in certain environments today.

- This architectures uses mainframe computers to provide the main processing for all system functions, including user application programs and user interface programs, as well as all the DBMS functionality.
- The Terminals did not have processing power and only provided display capabilities A and connected to the central computer via various types of communications networks.
- As prices of hardware declined, most users replaced their terminals with PCs and workstations.

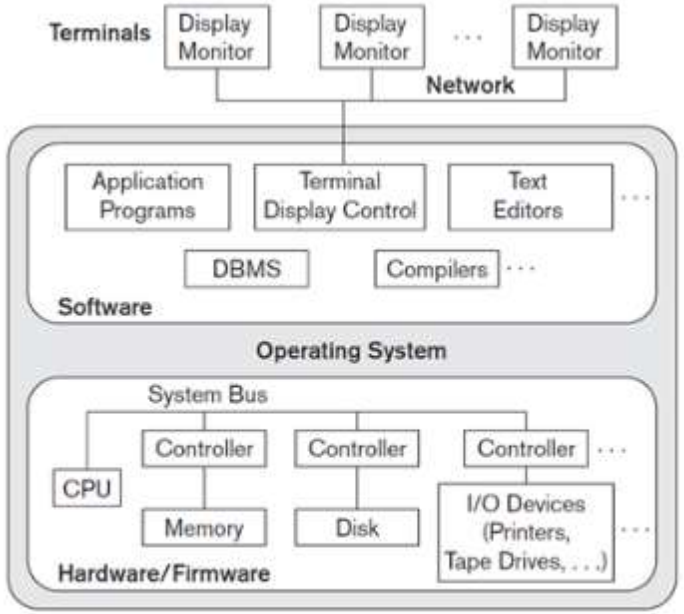


Figure 2.4
A physical centralized architecture.

DBMS Architecture (contd)

2. Basic Client/Server Architectures

The client/server architecture was developed to deal with computing environments in which a large number of PCs, workstations, file servers, printers, data base servers, Web servers, e-mail servers, and other software and equipment are connected via a network. The idea is to define specialized servers with specific functionalities.

File server : Connect a number of PCs or small workstations as clients to a file server that maintains the files of the client machines.

Print server : Another machine can be designated as a **printer server** by being connected to various printers; all print requests by the clients are forwarded to this machine.

Similarly , **Web servers** or **e-mail servers** also fall into the specialized server category. The resources provided by specialized servers can be accessed by many client machines. The **client machines** provide the user with the appropriate interfaces to utilize these servers, as well as with local processing power to run local applications. This concept can be carried over to other software packages, with specialized programs.

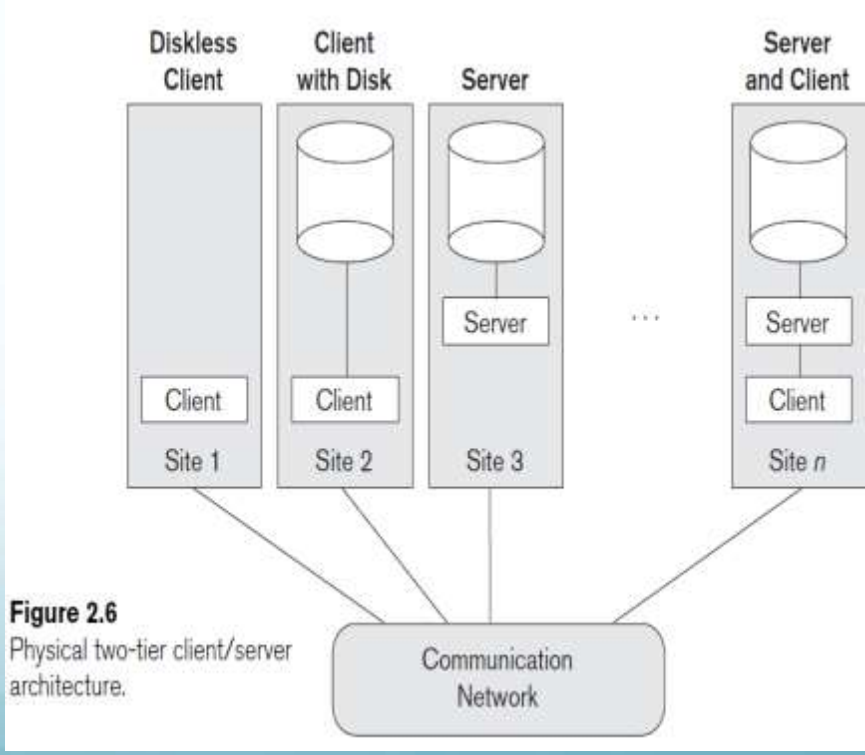


Figure 2.6
Physical two-tier client/server architecture.

DBMS Architecture (contd)

3. Three-Tier and n-Tier Architectures

Three-tier and N-tier architectures are used in modern distributed systems, particularly for enterprise applications, including Database Management Systems (DBMS). These architectures provide a way to separate different concerns (such as user interface, application logic, and data storage) into distinct layers or tiers. This separation allows for scalability, flexibility, and better management of the application. The three-tier architecture separates an application into three distinct layers or tiers:

- ✓ Presentation Tier (Client Layer)
- ✓ Application/Logic Tier (Middle Layer)
- ✓ Data Tier (Database Layer)

1. Presentation Tier (Client Layer)

This tier is responsible for the **user interface** (UI). It interacts with the end-user, displaying the data retrieved from the database and sending user requests to the application tier. This includes web browsers, mobile apps, or desktop applications that present data to the users and allow them to interact with the system.

2. Application Tier (Middle Layer / Business Logic Layer)

This is where the **business logic** of the application resides. It processes user requests, performs business rules, and interacts with the data tier to retrieve or modify data. This includes , Application servers, API servers, web servers, or microservices.

DBMS Architecture (contd)

The Application plays an intermediary role by running application programs and storing business rules (procedures or constraints) that are used to access data from the database server. It is also responsible to check client Authenticity before forwarding request of client to Database. It receives back the response from DB and re routes back to the client.

1. *Data Tier (Database Layer)*

This tier is responsible for data storage and management. It handles all the operations related to querying, updating, and storing data in the database. This includes , Relational databases (e.g., MySQL, PostgreSQL, Oracle), NoSQL databases, or any persistent data storage system. It is responsible for creating schemas, tables, indexes, and relationships. Promises ACID property compliance and ensures queries are executed in the most efficient way.

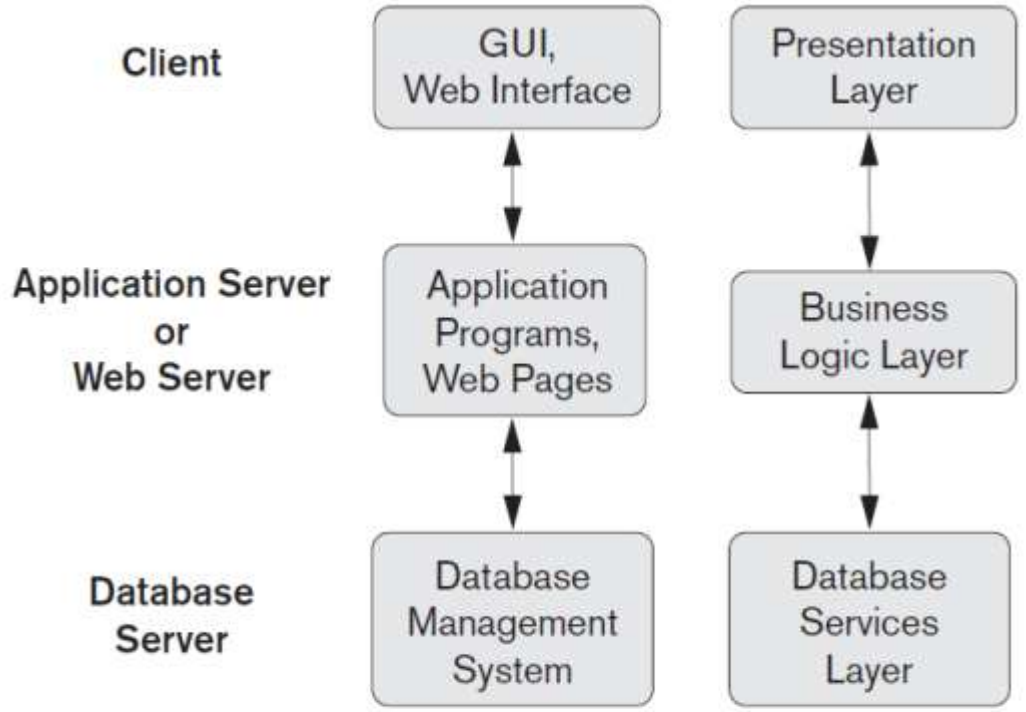


Figure 2.7
Logical three-tier client/server architecture, with a couple of commonly used nomenclatures.

Data Independence

In traditional file processing, the structure of data files is embedded in the application programs, so any changes to the structure of a file may require changing all programs that access that file.

> For example, a file access program may be written in such a way that it can access only STUDENT records of the structure. If we want to add another piece of data to each STUDENT record, say the Birth_date, such a program will no longer work and must be changed.

By contrast, in a DBMS environment, we only need to change the description of STUDENT records in the catalog to reflect the inclusion of the new data item Birth_date; no programs are changed.

In DBMS , thus the structure of data files is stored in the DBMS catalog separately from the access programs. This is called **program-data independence**. It is the ability to modify a schema definition at one level without affecting a schema definition in the next higher level.

Physical data independence

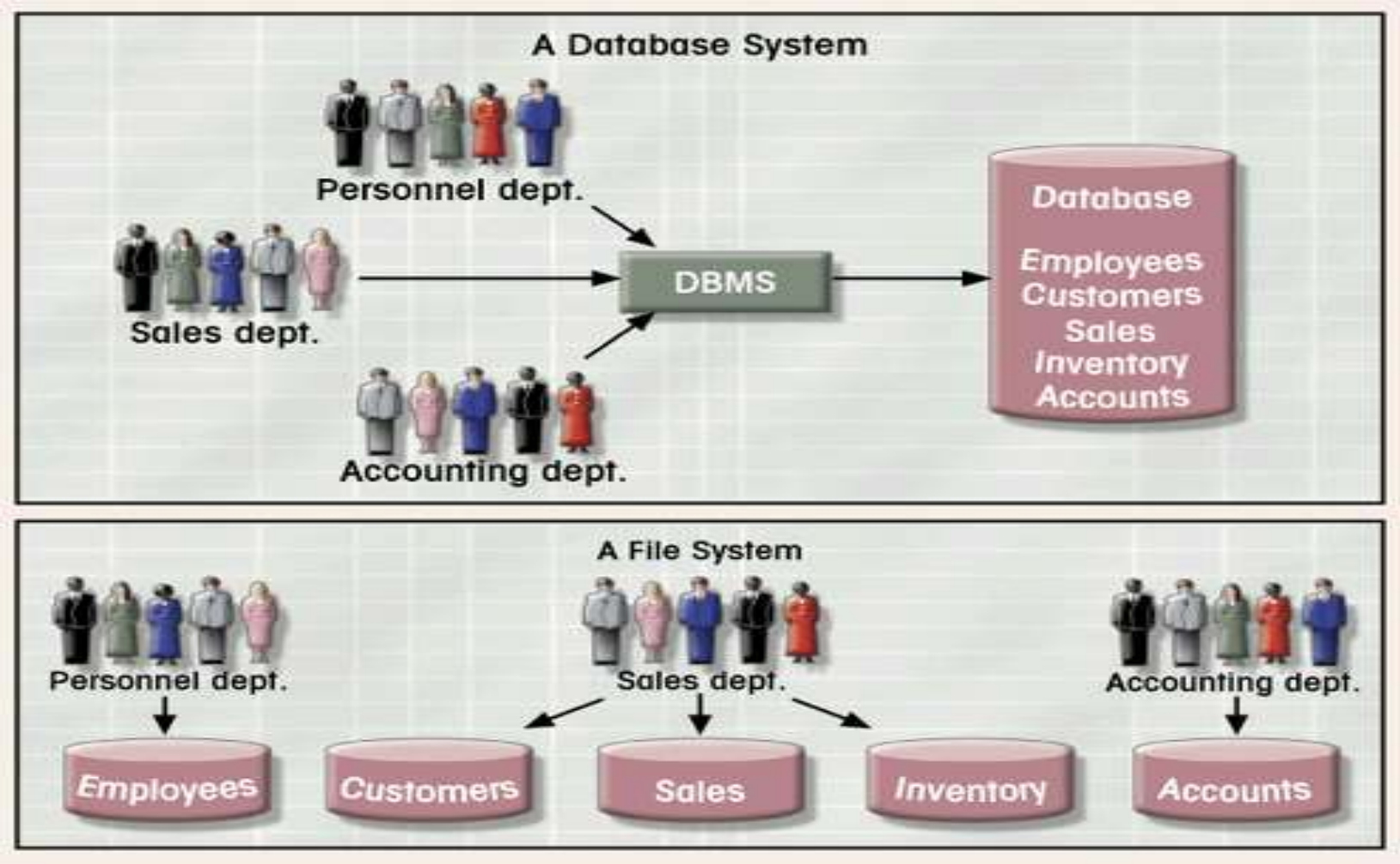
- Ability to modify physical schema without causing the conceptual (or external) schema or application programs to be re-written.

Logical data independence

- Ability to modify the conceptual schema without having to change the external schemas or application programs.

Database vs. File Systems

The **Database Management System (DBMS)** and **File System** are two different approaches to data storage and management. While a file system manages and organizes files on storage devices, a DBMS provides a higher level of functionality for data management, enabling more advanced operations, security, and consistency.



Advantages of DBMS

1. Controlling Redundancy

Problem with File Systems: In traditional file systems, data is often duplicated across multiple files, leading to data redundancy (duplicate data) and inconsistency (mismatched or outdated data across files).

Advantage of DBMS: A DBMS minimizes redundancy by storing data centrally, allowing multiple applications and users to access the same data. This leads to data consistency since all users refer to the same source of information.

> In a university database, instead of each department maintaining its own copy of student records, a centralized DBMS stores all student data. If a student changes their address, it needs to be updated in only one place, ensuring consistency.

2. Transaction Management

Problem with File Systems: Ensuring that multiple operations are either fully completed or not executed at all (atomicity) is difficult to implement in file-based systems.

Advantage of DBMS: A DBMS supports transaction management, ensuring that all operations within a transaction are completed successfully before the data is saved. If any operation fails, the system rolls back to maintain consistency. This guarantees the ACID properties (Atomicity, Consistency, Isolation, Durability) of transactions.

Advantages of DBMS

3. Concurrent Access

Problem with File Systems: When multiple users try to access or update the same file, it can lead to issues like data corruption or loss.

Advantage of DBMS: DBMS systems allow multiple users to access and modify the data concurrently while maintaining data consistency and isolation. This is done through concurrency control mechanisms like locking and transaction isolation levels.

4. Backup and Recovery

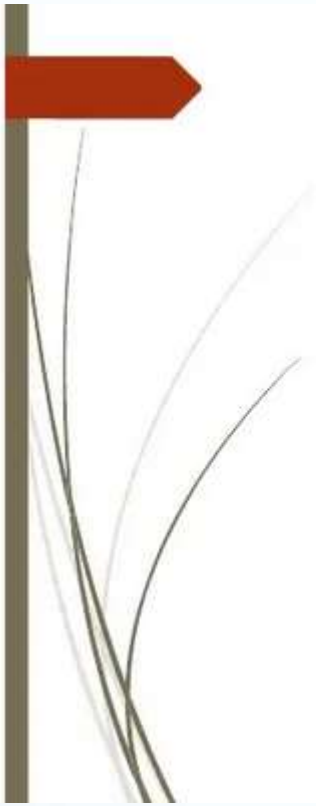
Problem with File Systems: Regular backups and reliable recovery options are difficult to implement.

Advantage of DBMS: A DBMS provides automated backup and recovery mechanisms. In case of system failure, the DBMS ensures that data can be recovered to a consistent state using techniques such as logging and check pointing.

5. Improved Data Integration

Problem with File Systems: Managing and linking data from multiple sources can be inefficient and difficult.

Advantage of DBMS: A DBMS allows integration of data across multiple sources by defining relationships between tables and databases, providing a holistic view of the organization's data.



Thanks