DATABASE SYSTEMS

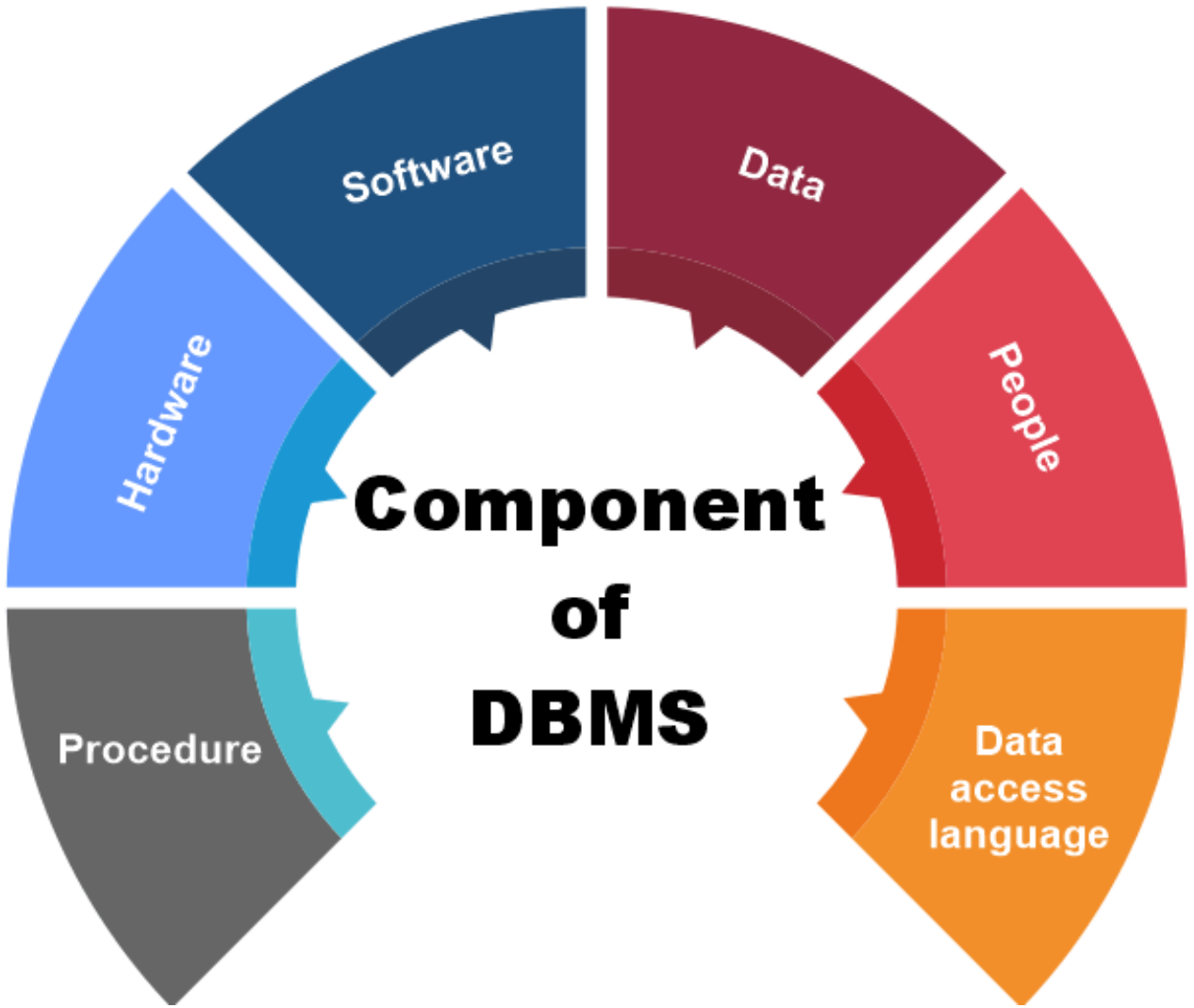
Database Architecture

Faculty of AI & MMG

What is Database Architecture?

- Database architecture refers to the design and structure of a database system.
- Determines how data is stored, accessed, and managed.

- **Importance:**
  - Ensures efficient data handling.
  - Supports scalability, security, and multi-user environments.



MAJOR COMPONENTS OF DATABASE SYSTEMS

Major Components of Database Systems

- **Procedures:** Instructions and rules for database operation, maintenance, and use.
  - The procedure is a type of general instruction or guidelines for the use of DBMS.
  - This instruction includes how to set up the database, how to install the database, how to log in and log out of the database, how to manage the database, how to take a backup of the database, and how to generate the report of the database.
  - In DBMS, with the help of procedure, we can validate the data, control the access and reduce the traffic between the server and the clients.
- **Hardware:** The physical devices where the database resides and operates, such as servers, storage devices, and network infrastructure.

Major Components of Database Systems

- **Software:** This includes:
  - Database Management System (DBMS): Software responsible for managing, storing, retrieving, and manipulating data.
  - Application Software: Programs that interact with the database to provide user interfaces or business logic.
  - Operating System: Supports the DBMS and related applications.
- **Data:** The database contains the actual data and metadata.
  - Here metadata means data about data.
  - For example, when the user stores the data in a database, some data, such as the size of the data, the name of the data, and some data related to the user, are stored within the database. These data are called metadata.

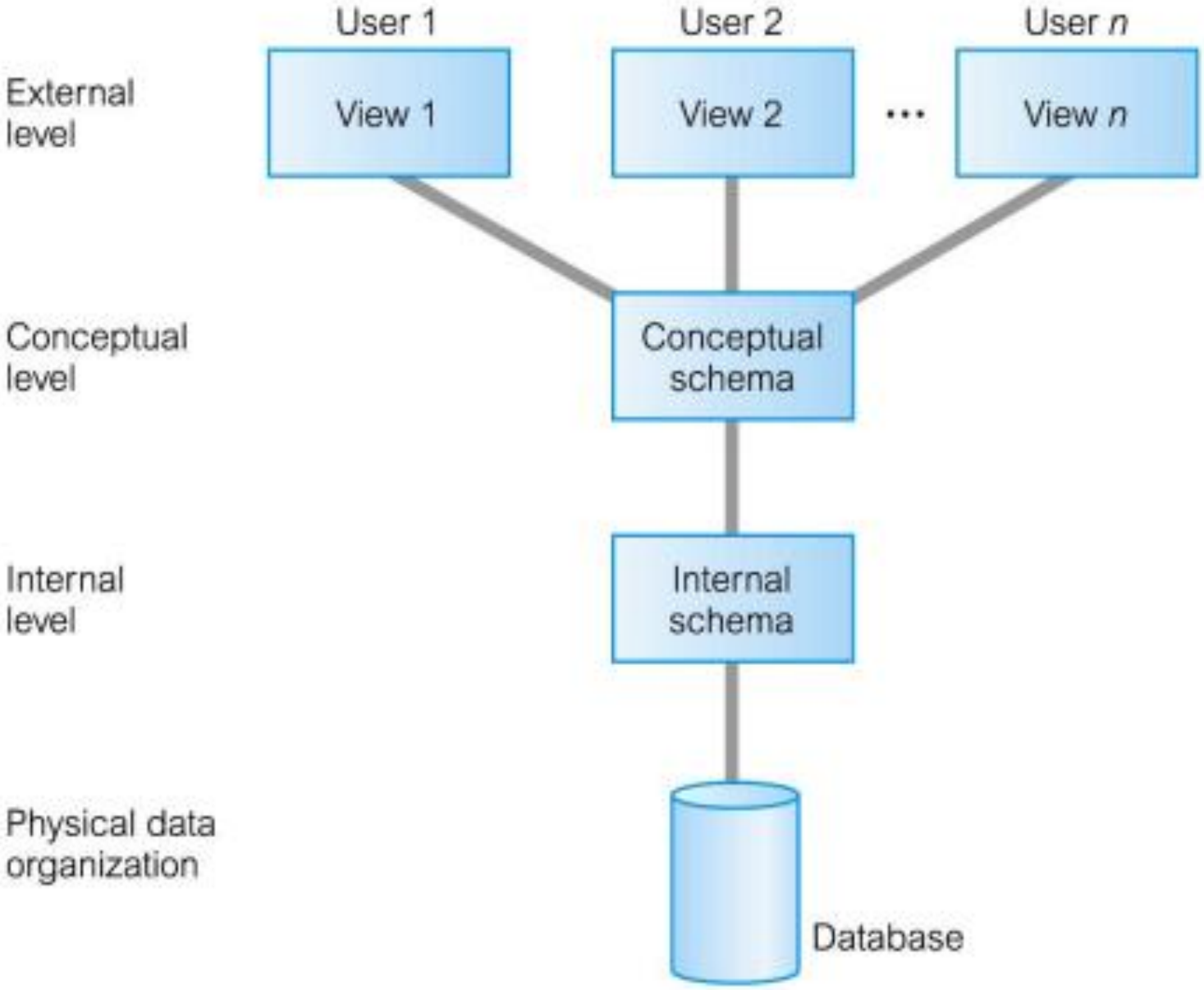
Major Components of Database Systems

- **People / Users:** The people or applications interacting with the database. Users can be categorized as:
  - End Users: Access data directly for queries, reports, or updates.
  - Application Developers: Build programs to interact with the database.
  - Database Administrators (DBAs): Manage the database, ensuring efficiency, security, and backup.
- **Database Access Language:** It is a language that allows users to write commands to perform the desired operations on the data that is stored in the database.
  - e.g. SQL

Three-Level Architecture

The ANSI-SPARC Three-Level Architecture is a framework proposed by the **American National Standards Institute (ANSI)** and the **Standards Planning and Requirements Committee (SPARC).**

It is a logical database architecture designed to separate the user's view of the database from its physical storage.



THREE-LEVEL ARCHITECTURE

Purpose of Three-Level Architecture

Data Independence:

- Allows changes at one level without affecting the others.

Flexibility:

- Supports multiple views of the data for different users.

Abstraction:

- Hides complex details of physical data storage from users and applications.

Consistency and Security:

- Enables centralized control of data access and modifications.

Analogy

External Level:

Think of a restaurant menu. Different customers see the menu with dishes relevant to their preferences (vegetarian, non-vegetarian), without needing to know the recipes or kitchen operations.

Conceptual Level:

The chef's recipe book represents the conceptual schema. It lists all dishes and their preparation steps, which is consistent for all customers.

Internal Level:

The kitchen setup represents the internal level. It includes storage, ovens, and appliances required for cooking but hidden from the customers

Levels of the Architecture

1. **External Level (User View):**
   - This is the **highest level** of abstraction.
   - This level describes that part of database that is relevant to a particular user.
   - Defines how users or applications interact with the database.
   - Each user can have a **customized view** of the database without being aware of the full schema.
   - **Example:**
     - An HR manager sees only employee records, while a finance team sees payroll data.

Levels of the Architecture

1. **External Level (User View):**
   - The external schema describes the structure and organization of data as perceived by individual users or applications interacting with the database.
   - It defines user-specific views or subsets of the overall database, tailored to meet specific user or application requirements.
   - **Example in University Management System:** In a university management system, the external schema might include views for different user roles such as students, professors, and administrative staff. For example:
     - **Student View:** Includes information such as studentID, name, enrolled courses, grades, and personal details.
     - **Professor View:** Includes information such as professorID, courses taught, class schedules, and research projects.
     - **Administrator View:** Includes information such as departmental data, faculty information, course offerings, and enrollment statistics.

Levels of the Architecture

2. Conceptual Level (Logical View):

- Represents the **logical structure** of the entire database.
- Defines the data organization (e.g., entities, relationships) and constraints, but **abstracts away physical details** like storage and indexing.
- Ensures a unified logical representation that connects all user views.
- Community view of the database. This level describes what data is stored in database and relationships among the data.
- **Example:**
  - A relational schema with tables, columns, keys, and relationships.

Levels of the Architecture

2. Conceptual Level (Logical View):

- The conceptual schema represents the entire database from a unified, abstract perspective, independent of specific user or application requirements.
- It defines the logical structure of the database, including all entities, attributes, relationships, and integrity constraints.
- **Example in University Management System:** In the conceptual schema of a university management system:
- Entities include Student, Professor, Course, Department, Class, Assignment, and Grade.
- Attributes are defined for each entity (e.g., Student ID, Name, Email, Course Name, Department Name).
- Relationships are established between entities (e.g., Enrolls In, Teaches, Belongs To).

Levels of the Architecture

3. Internal Level (Physical View):

- The **lowest level** of abstraction.
- Focuses on the **physical storage** of data (e.g., how data is stored on disk, indexing, file systems).
- Ensures efficient storage, retrieval, and maintenance.
- Describes how the data is stored in the database.
- **Example:**
    - Data stored using B-trees or hash indexing.

Levels of the Architecture

3. Internal Level (Physical View):

- The physical schema describes the internal organization of data within the database system, including storage structures, access methods, and optimization techniques.
- It specifies how data is stored on storage devices and includes details such as data types, constraints, indexes, and storage allocation specifics.
  - **Example in University Management System**: In the physical schema of a university management system:
  - Data types are specified for attributes (e.g., Student_ID as INT, Name as VARCHAR).
  - Constraints are defined (e.g., Primary Key constraints on Student_ID, Foreign Key constraints on Course_ID).
  - ndexes are created for optimizing data retrieval (e.g., Index on Course_ID for efficient course lookups).

Diagram: ANSI-SPARC Architecture

External Level: User-Specific Views

↕

Conceptual Level: Logical Schema (Unified View of Data)

↕

Internal Level: Physical Storage (File Systems, Indexing)

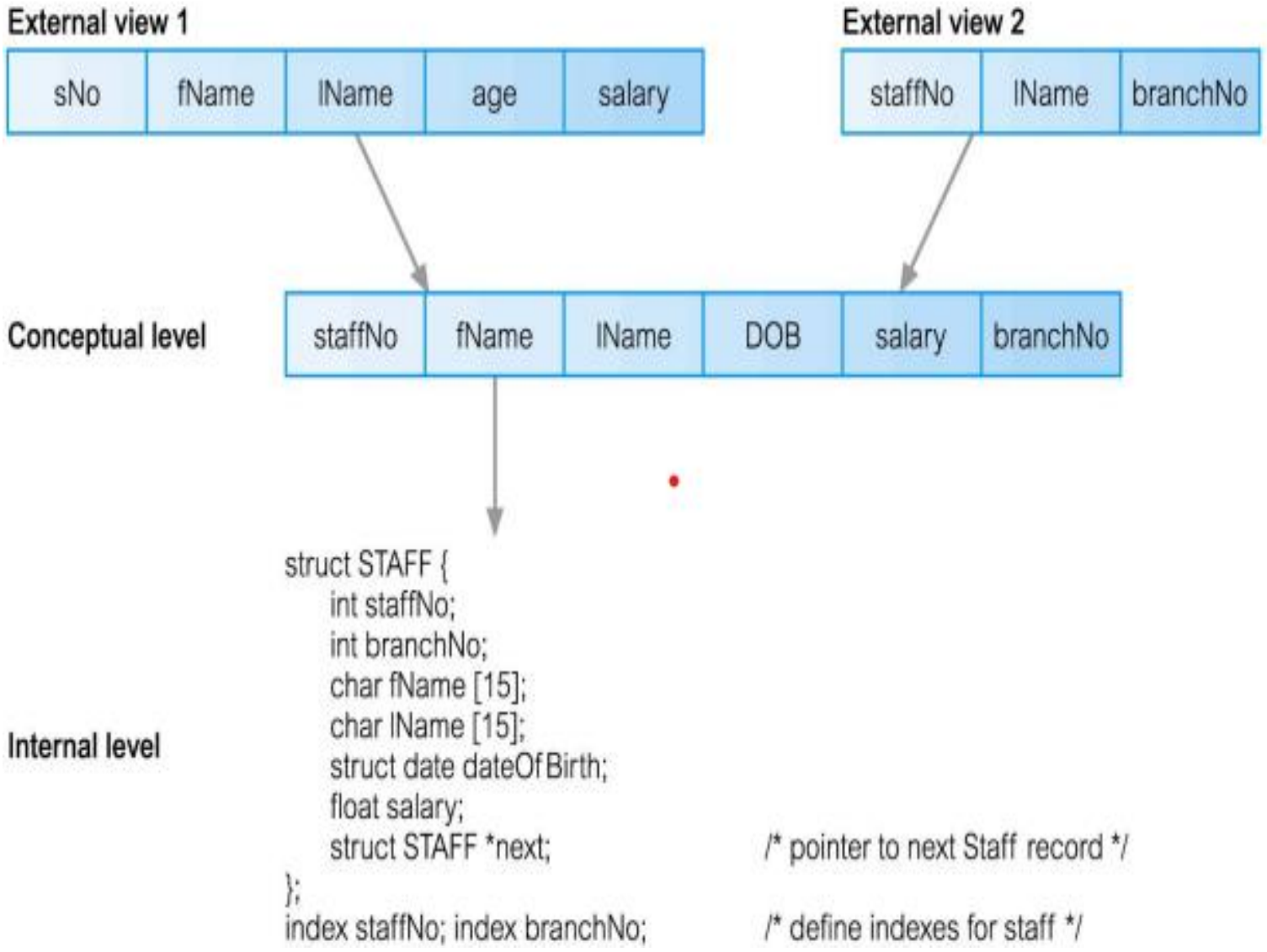
Example in Real Systems

Banking System:

- **External Level:** A customer sees their account balance, while an auditor views transaction logs.
- **Conceptual Level:** Represents the relationships between accounts, transactions, and customers.
- **Internal Level:** Data is stored using indexed tables on disk.

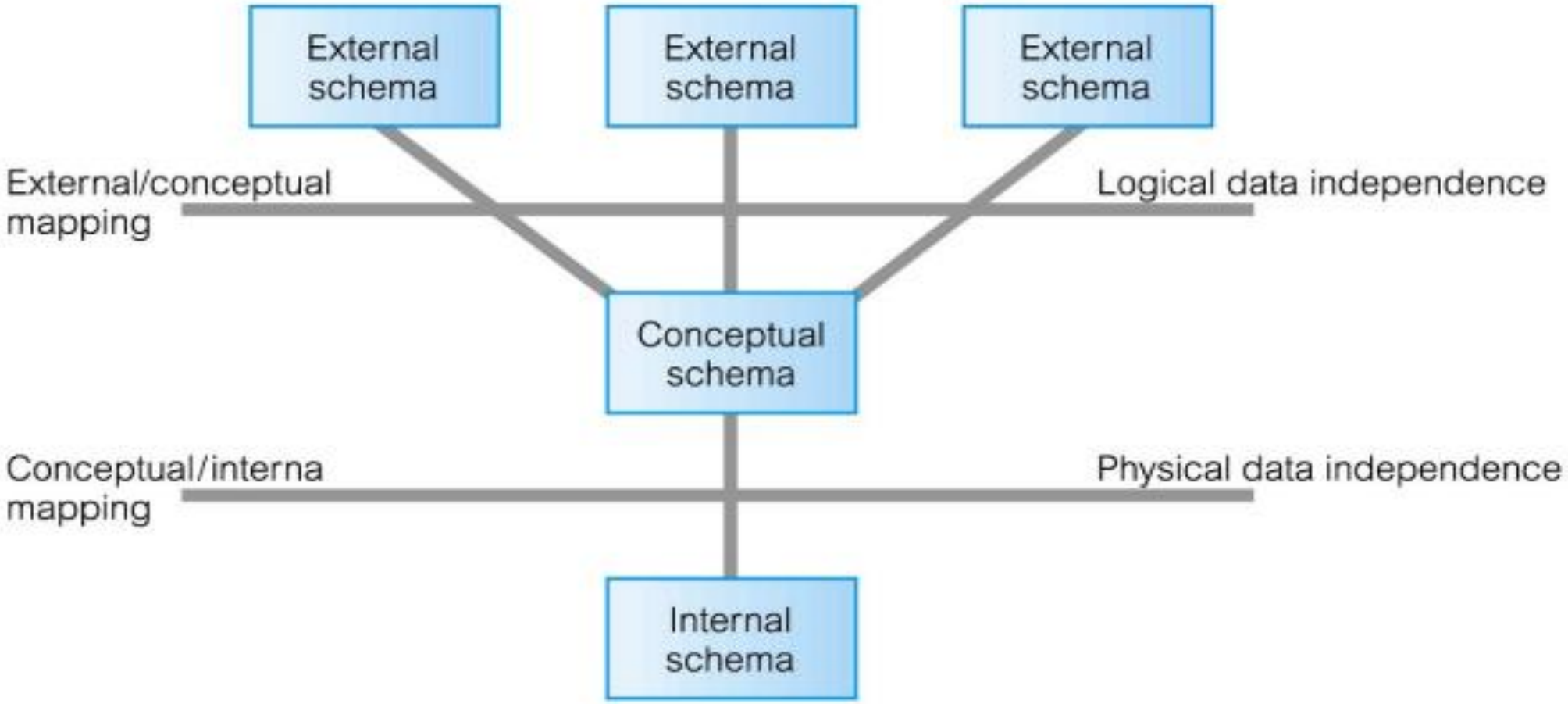
E-Commerce:

- **External Level:** Customers see product categories, while inventory managers see stock levels.
- **Conceptual Level:** Defines relationships between products, orders, and customers.
- **Internal Level:** Products and orders are stored physically with indexing for quick access.



DIFFERENCES BETWEEN THREE LEVELS OF ANSI-SPARC ARCHITECTURE

Data Independence and the ANSI-SPARC Three-Level Architecture



- **Data independence** ensures that changes at one level of a database system do not affect higher levels.
- **Logical Data Independence**
  - Logical data independence refers to the **ability to change the conceptual schema without requiring changes to the external schemas** or application programs.
  - This means modifications such as **adding or removing entities, attributes, or relationships do not impact how users interact with the database**.
  - For example, if a university database adds a new attribute for student email addresses, logical data independence ensures that existing applications retrieving student information remain unaffected.

Data Independence and the ANSI-SPARC Three-Level Architecture

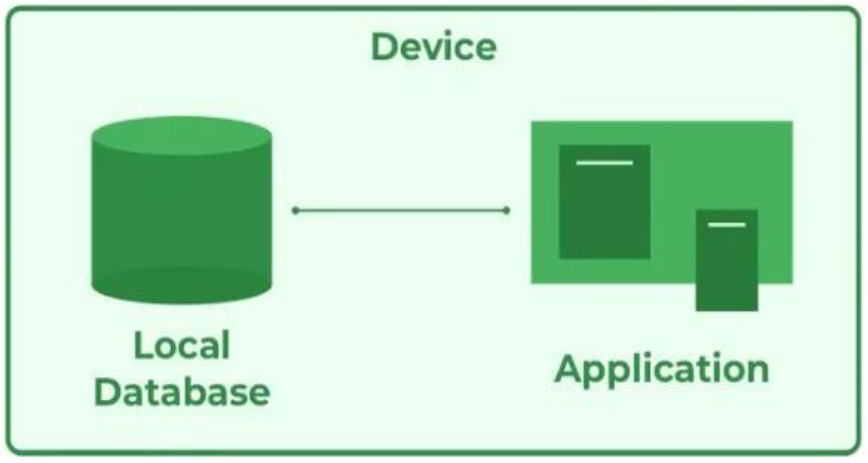
- **Physical Data Independence**
  - Physical data independence **ensures that changes to the internal schema (how data is physically stored) do not affect the conceptual schema**.
  - Changes such as using different storage structures, file organizations, indexing techniques, or **new storage devices should not impact the way data is logically structured**.
  - From a user's perspective, the only **noticeable change might be improved performance.**

Data Independence and the ANSI-SPARC Three-Level Architecture

Single-Tier, Two-Tier, and Three-Tier Architectures of Database Systems

- **Real-Life Analogy:** Think of a **restaurant**:
- **Single-Tier**: A food cart where the chef prepares and serves food directly to the customer.
- **Two-Tier**: A restaurant where a waiter takes orders and directly communicates with the kitchen.
- **Three-Tier**: A restaurant with a digital ordering system where waiters input orders into a system, which processes the request before sending it to the kitchen.

3-Tier / Database & Application Architecture

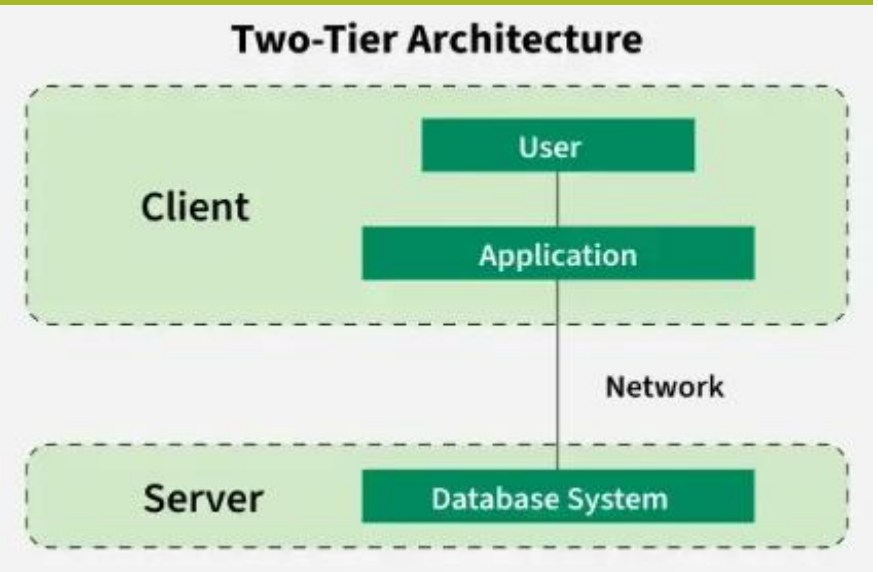


1. Single-Tier Architecture

- In a single-tier architecture, the database and application reside on the same machine.
- This structure is commonly used in **standalone** applications, where there is no network communication between different components.
- **Example:** A personal finance management software installed on a user's laptop that manages all data locally.

- **Advantages:**
  - Simple setup with no network dependency.
  - Faster data access as everything is on a single system.
  - Less complex management.
- **Disadvantages:**
  - Not suitable for multi-user environments.
  - Limited scalability.
  - Poor security as everything is stored in one place.

3-Tier / Database & Application Architecture



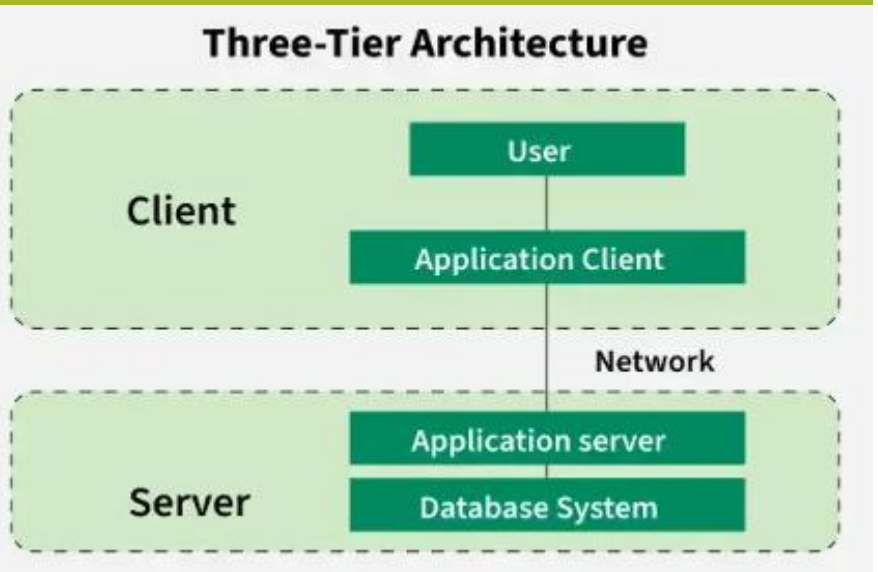
2. Two-Tier Architecture (Client-Server Model)

- In a two-tier architecture, the system is divided into:
  - **Client Tier (User Interface & Application Logic)**
  - **Database Server (DBMS & Data Storage)**
- The client directly communicates with the database server via SQL queries.

- **Example:** A banking system where tellers use client applications to send SQL queries to a central database.

- **Advantages:**
  - Improved performance by separating the application and database.
  - Centralized database management.
  - Supports multiple users.
- **Disadvantages:**
  - **Fat Client Problem**: The client application requires significant resources (processing and memory).
  - Limited scalability as adding more clients increases server load.
  - Network congestion if too many queries are sent from multiple clients simultaneously.

3-Tier / Database & Application Architecture



3. Three-Tier Architecture

- The **three-tier model** improves upon the two-tier architecture by adding a **middle tier (Application Server)** between the client and database.
- The system is divided into:
  - **Presentation Layer (Client Interface)** – Runs on user devices.
  - **Application Layer (Business Logic & Processing)** – Runs on an intermediate server, handling business logic and communicating with the database.
  - **Database Layer (DBMS & Data Storage)** – Stores and retrieves data.
- **Example:** A web-based e-commerce platform where users access a web application, which in turn interacts with a database to retrieve product details.
- **Advantages:**
  - **Better Performance**: The application server handles processing, reducing the load on the database.
  - **Scalability**: Additional application servers can be added to handle more users.
  - **Improved Security**: Users do not directly access the database; the application layer serves as a buffer.
- **Disadvantages:**
  - Increased system complexity.
  - Higher infrastructure and maintenance costs.
  - Requires proper network and server management.


Q&A