**Data,Database and File System**

Data is referred to as the collection of raw facts or figures that can be processed to derive meaning or knowledge. It is a collection of information gathered by observations, measurements, research, or analysis. In simpler terms, data is any fact that can be stored, e.g., "EXY", "12", etc.

**Data when processed becomes Information**. Information is the knowledge obtained from investigating, studying, or instructing data. For example, "Raj" is just data that can be written in a register, but when assigned to a person, it becomes information, i.e., the name of that man.

Consider the string “orange”. Here, it could mean a color or a fruit. This term “orange” is data. However, if we say “color orange” or “fruit orange”, it becomes information.

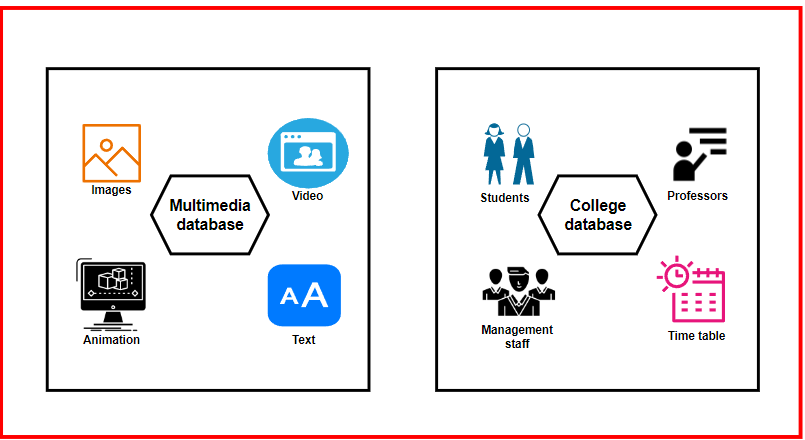
Similarly, “12” could mean age, pocket money, or roll number. Hence, this is data. When we say “Roll number 12,” it becomes information.

**Database**

A database is a structured collection of interrelated data organized in a way that enables efficient storage, retrieval, and manipulation of information. Key characteristics of databases include:

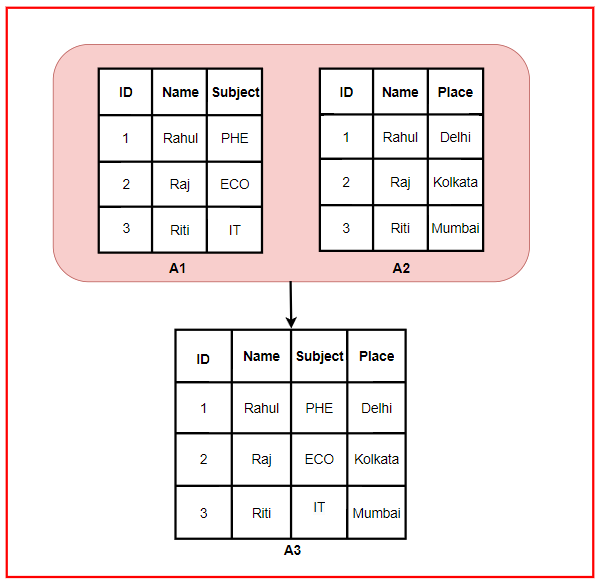
* It is a collection of interrelated data.
* It can be stored in the form of tables.
* It can be of any size.

Examples of databases:



For example, in a multimedia database, the image table would contain information about images like pixels, length, and width. Similarly, the video table would have features like pixels and video length. The image and video tables would be part of the multimedia database.

Similarly, a college database would have tables like professor and student tables that are related to the timetable for class schedules.



Here, we can see that "ID" and "Name" are common columns in both A1 and A2 tables. Hence, it is a collection of related data. When merging A1 and A2, we get an A3 table with columns like ID, Name, Subject, and Place.

**File System**

A file system is a structure that an operating system uses to manage and organize files on a storage device, such as a hard drive or USB flash drive. It defines how data is organized, accessed, and stored on the storage device. The file system acts as an interface between the user and the data.

**Disadvantages of the File System**

* **Data Redundancy**: Imagine a company using separate spreadsheets for sales, customer contacts, and inventory. If a customer buys a product, their information might be entered in all three spreadsheets, causing duplication.
* **Poor Memory Utilization**: Due to storing the same information like customer names and phone numbers in multiple spreadsheets, memory resources are poorly utilized.
* **Data Inconsistency**: If a customer's address is updated in the customer contacts spreadsheet but remains unchanged in the sales spreadsheet, it causes data inconsistency.
* **Data Security**: File systems do not ensure controlled access to sensitive data. In DBMS, it is possible to limit access to specific data, which protects against unauthorized access.

**Database Management System (DBMS)**

A Database Management System (DBMS) is software designed to manage, manipulate, and organize large volumes of data efficiently. It acts as an interface between the database and the users or applications, providing tools for storing, retrieving, updating, and managing data securely.

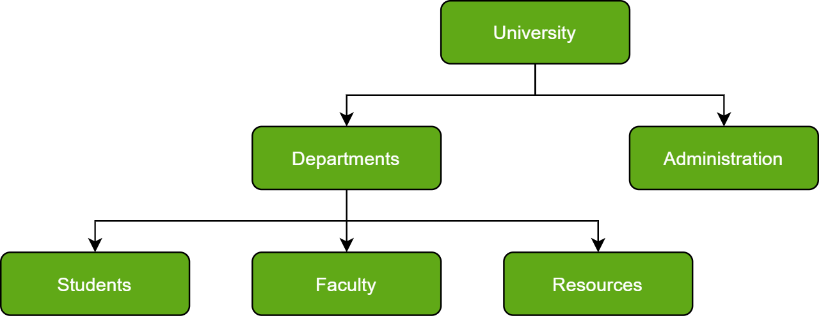
**Real-life Applications of DBMS**

* **Banking Systems**: DBMS maintains a centralized and secure database of customer information like personal details and transaction history. For example, banks rely on DBMS to ensure data consistency when handling millions of transactions.
* **Airline Reservation Systems**: Airlines use DBMS to manage flight schedules, seat availability, and reservations. DBMS helps maintain data integrity across thousands of flights and millions of passengers.
* **Education Management Systems**: Schools and universities use DBMS to store and manage student information such as academic records and attendance.

**Types of Database**

**1. Hierarchical Databases**

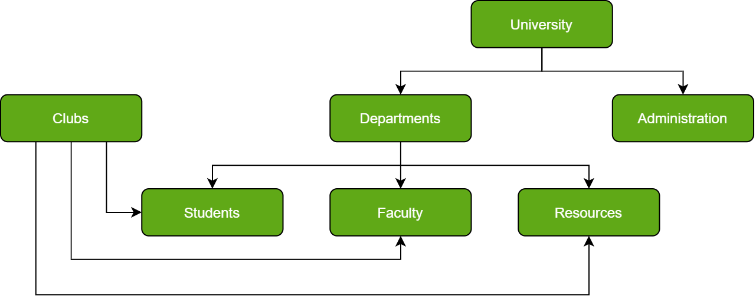
**Hierarchical databases** organize data in a**tree-like structure,**where each parent record can have multiple child records**.**This model works well for scenarios where data follows a predefined hierarchical relationship, where data is arranged in **levels**or **ranks**.



This structure can also be viewed as a **parent-child relationship**, where each parent record can have multiple child records, but a child record can only have one parent. As more data is added, the structure expands like a tree.

**2. Network Databases**

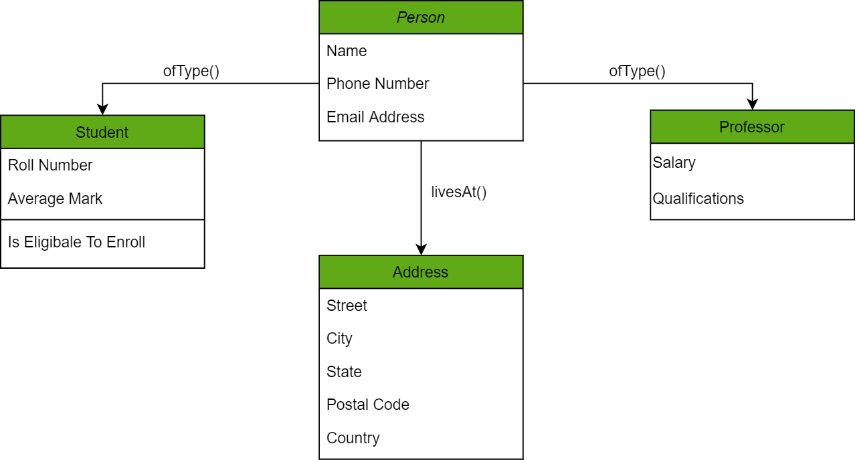
A **network databases** builds on the hierarchical model but allows child records to link to **multiple parent records**, creating a **web-like structure** of interconnected data. This results in a more flexible structure, often referred to as a **graph model**, where entities can be connected in many different ways.



This model is ideal for **complex frameworks**as it effectively represents **many-to-many relationships**. Additionally, its structure simplifies the use of certain database management languages.

**3. Object-Oriented Databases**

Object-oriented databases are based on the principles of **(OOP)**, where data is stored as objects. These objects include attributes (data) and methods (functions), making them easily **referenced**and **manipulated**. These databases are designed to handle complex data structures such as multimedia, graphics, and large files.



**4. Relational Databases**

Relational databases are the most widely used type of database today. They store data in **tables**, with rows representing records and columns representing attributes of the records. In this database, every piece of information has a relationship with every other piece of information. This is on account of every data value in the database having a **unique identity** in the form of a record.   
Note that all data is tabulated in this model. Therefore, every row of data in the database is linked with another row using a  **primary key**. Similarly, every table is linked with another table using a **foreign key**. Refer to the diagram below and notice how the concept of '**Keys**' is used to link two tables.

**5. Cloud Databases**

A **cloud database** operates in a **virtual environment** hosted on cloud computing platforms. It is designed for storing, managing, and executing data over the internet, providing **flexibility**and **scalability**. Cloud databases are widely used for applications requiring **dynamic workloads**, as they eliminate the need for on-premises infrastructure.

Common **cloud services** for accessing and managing databases include [SaaS](https://www.geeksforgeeks.org/software-as-a-service-saas/)**(Software as a Service)** and [PaaS](https://www.geeksforgeeks.org/platform-as-a-service-paas-and-its-types/)**(Platform as a Service)**, which simplify database operations for businesses. Popular cloud platforms offering database services include:

* Amazon Web Services (AWS)
* Google Cloud Platform (GCP)
* Microsoft Azure
* ScienceSoft, etc.

**6. Centralized Databases**

A **centralized database** is a database stored and managed at a **single location**, such as a **central server** or data center. It ensures **higher security** and consistency as all data is maintained in one place, making it easier to control and manage.

Users access the database remotely to fetch or update information. Centralized databases are commonly used in **enterprise systems** where data consistency and security are critical. However, **scalability**and **performance limitations** should be carefully considered.

**7. Personal Databases**

A **personal database** is a **small-scale database** designed for a single user, typically used on personal computers or mobile devices. These databases are ideal for managing individual data like **contacts**, **budgets**, **notes**, or **schedules**. They are lightweight, easy to use, and require **minimal database**administration, making them accessible for non-technical users. Examples are:

* **Microsoft Access**: A simple database solution for personal or small business needs.
* **SQLite**: A lightweight, self-contained database commonly used in mobile and desktop applications.

**8. Operational Databases**

An **operational data**base is designed to **manage**and **process real-time data** for daily operations within organizations and businesses. It allows users to create, update, and delete data efficiently, ensuring that the database reflects current activities and transactions.

These databases handle live transactions and provide quick access to up-to-date data. **Example**: **SAP HANA** is an example of an operational database used for high-speed transactions and analytics.

**9. NoSQL Databases**

A **NoSQL database** (short for "non-SQL" or "non-relational") provides a mechanism for **storing**and **retrieving**data that does not rely on traditional table-based relational models. Instead, it uses flexible data models like **key-value pairs**, **documents**, **column families**, or **graphs**, making it ideal for handling unstructured, semi-structured, and structured data.

NoSQL databases are known for their **simplicity of design**, **horizontal scalability** (adding more servers for scaling), and **high availability**. Unlike relational databases, their data structures allow faster operations in certain use cases. [MongoDB](https://www.geeksforgeeks.org/mongodb-an-introduction/), for instance, is a widely used document-based NoSQL database.

**What is DBMS and its applications**

The acronym **DBMS** stands for “Database Management System”. A DBMS is a software application that acts as an interface between the data and the end user, allowing users to efficiently store, manage, retrieve, and manipulate large volumes of structured data. DBMS provides these functionalities by allowing database managers to implement certain rules and regulations in the system for different operations over the data.

Imagine your college’s library. Your library has thousands of books, and each book contains valuable information. Now, think of your library as a database, and each book as a record in that database.

* **Organisation**:

Data is arranged into tables in DBMSs, each having a unique collection of rows (like individual books) and columns (like book characteristics).

* **Search and Retrieval**:

In a DBMS, users can perform queries to search for specific data using structured query language (SQL) or other interfaces.

* **Access Control**:

In the library, not everyone can access every book. There are rules in place. Similarly, in a DBMS, access control mechanisms ensure that only authorized users can access certain data, protecting sensitive information.

* **Concurrency Control**:

In a DBMS, concurrency control mechanisms manage multiple users accessing the database simultaneously, preventing data corruption or loss.

* **Data Integrity**:

DBMS ensures data integrity by enforcing rules such as constraints and validations to maintain the accuracy and consistency of the data.

**Applications of DBMS**

* **Banks**:

DBMS allows banks and financial institutions to create a centralised and secure database of customer accounts, personal details, transaction history, and more. They maintain data integrity, security, and access control, which is crucial in financial applications.

* **Schools and Colleges**:

Educational institutions use DBMS to manage student records like admissions, grades, enrolled courses, faculty directories, and administrative functions.

* **E-Commerce**:

Online retail platforms rely on DBMS to store product catalogues, customer orders, payment information, and inventory data. A robust DBMS supports high transaction volumes and enables real-time inventory management and order processing.

* **Enterprise Resource Planning (ERP)**:

For any large enterprise, DBMS offers a consolidated repository and interface to store, manage, and report over large volumes of data generated by multiple concurrent operations like supply chain management, customer relationship management, finance, and human resources

**Need, advantages and disadvantages of DBMS**

**Need of DBMS**

* **Managing Data**:DBMS assists in managing operations like inserting, deleting, and manipulating data effectively.
* **Ensuring Data Accuracy and Security**:DBMS provides access control, ensuring that data is not accessible to unauthorized parties, which guarantees security and data accuracy.
* **Supporting Decision-Making**:DBMS provides control over how data is handled, which supports essential decision-making for businesses and organizations.

DBMS serves as the core of contemporary information systems, facilitating efficient data management and serving as a foundation for a wide range of applications and services.

**Advantages of DBMS**

* **Data Security**:DBMS implements security mechanisms that regulate access to sensitive information, safeguarding it from unauthorized access and potential data breaches.
* **Data Redundancy and Inconsistency**:DBMS eliminates data redundancy, minimizing storage needs and ensuring consistency by maintaining a unified version of the data. *Example:* If multiple registers store the same information, they would unnecessarily consume memory and increase the time required for updates.
* **Data Integrity**:DBMS ensures data integrity by enforcing rules and constraints that prevent incorrect or inconsistent data from entering the database. *Example:* If an age is to be entered in the database, the system can enforce that the input must be an integer.
* **Data Scalability**:DBMS can handle large datasets and scale to accommodate increasing amounts of data as an organization grows. *Example:* Adding 1200 new employees' details to a database originally containing 40 employees can be done seamlessly using DBMS.
* **Data Abstraction**:DBMS offers data abstraction, allowing users and applications to interact with the database without needing to understand its underlying complexities.

**Disadvantages of DBMS**

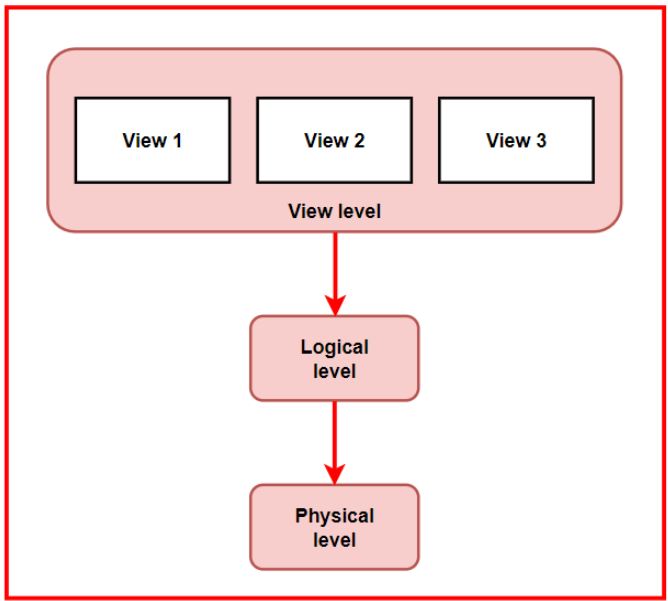
* **Cost**:Acquiring, deploying, and sustaining DBMS software can incur significant costs. Additionally, the hardware required for its operation may also be expensive.
* **Scale Projects**:For small-scale applications with minimal data storage needs, using a comprehensive DBMS may introduce unnecessary complexities and burdens. *Example:* If an organization only wants to store data for 20 people and does not plan to add more, a DBMS may be overkill and more streamlined alternatives would be better suited.
* **Vendor Lock-In**:Switching to a different DBMS can be challenging due to differences in data formats and query languages, leading to vendor lock-in. *Example:* A company using SQL to store data may find it difficult to switch to NoSQL due to their structural differences (SQL stores data in rows and columns, while NoSQL uses key-value pairs).

**Data Abstraction**

Database systems are built with complex ways of organizing data. To make it easier for people to use the database, the creators hide the complicated details that users don't need to worry about. This hiding of unnecessary complexities from users is called data abstraction.

*For example:* A user will not be concerned about indexing, memory storage locations, or what data structures are used behind the scenes. The user can directly interact with the database without needing to understand these underlying complexities.

**Levels of Abstraction**



* **Physical Level**:

This is the lowest level of data abstraction. It describes how data is stored in a database, providing details about complex data structures.

*For example:* Think of a library storing books. The physical level involves how the books are physically stored on shelves, the materials used to make the shelves, and how the shelves are arranged within the library space.

* **Logical Level**:

This is the middle level of the 3-level data abstraction architecture. It describes what data is stored in the database and how it is structured.

*For example:* Consider a library's catalogue system. At the logical level, it’s like looking at how books are categorised and indexed for easy retrieval. This includes details like the indexing methods used and how books are classified by subject, author, and title. It helps visitors search and find books based on their interests.

* **View Level**:

This is the highest level of data abstraction. It describes the interaction between users and the database system, focusing on what users can see and access.

*For example:* In a library, different visitors may have different levels of access. At the view level, it's like determining who can borrow specific types of books or access restricted sections based on their role or membership level.

**DBMS Architecture**

A schema is a logical container or structure that organises and defines the structure of a database.

It defines how data is organised, what data types are used, what constraints are applied, and the relationships between different pieces of data. A schema acts as a blueprint for the database, ensuring data integrity, consistency, and efficient data retrieval.

**Types of Schema**

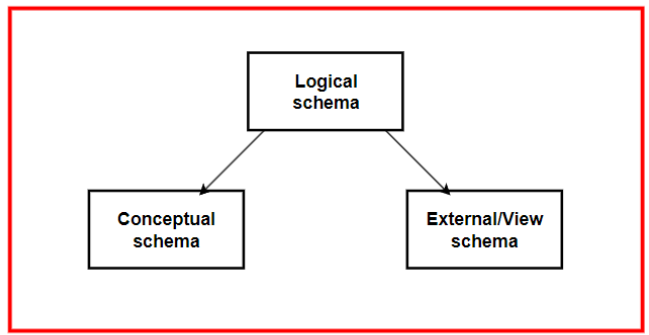
* 1. **Physical Schema**:

A physical schema defines how data is stored on the underlying hardware, including details such as storage format, file organisation, indexing methods, and data placement.

**Characteristics of Physical Schema:**

* Its primary focus lies in enhancing the storage and retrieval of data to boost performance.
* Modifications made to the physical schema demand meticulous planning and can potentially affect the overall performance of the database.
* **Example:** Deciding to use clustered indexes on specific columns for faster retrieval.
  1. **Logical Schema**:

A logical schema defines the database’s structure from a logical or conceptual perspective, without considering how the data is physically stored.



**Types of Logical Schema**

* **Conceptual Schema:**

The conceptual schema represents the overall view of the entire database. It defines the high-level structure and relationships between all data elements.

*For example:* Consider a university database with entities such as Student (StudentID, Name, Address, DateOfBirth), Course (CourseID, CourseName, Credits), and Department (DepartmentID, DepartmentName, OfficePhone). Relationships among them are:

* Students can enrol in multiple Courses.
* Each Course can be taken by multiple students.
* Each Course is offered by one Department, but a Department can offer multiple Courses.

The focus is on what the data represents (students, courses, departments) and how these entities are interrelated through relationships like enrollment and offerings.

* **External/View Schema:**

An external schema defines the user-specific views of the database. It focuses on the portions of the database that are relevant to specific user roles or applications.

*For example:* In a university database, for a Student's Portal, the view will have StudentProfile (StudentID, Name, Address, CoursesEnrolled). This view provides a student with access to their data and their course enrollments, but not to other students' information or course details like credits or department.

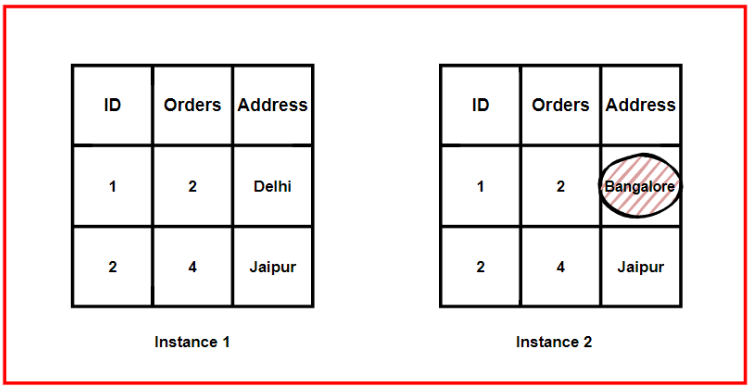
**Characteristics of Logical Schema**

* It delineates how data is structured into tables, the interconnections between these tables, and the restrictions placed on the data.
* Logical schemas prioritise data modelling and database design over considerations related to hardware or storage specifics.
* **Example:** Defining tables, specifying primary and foreign keys, and creating views for data access.

**Instance**

The information residing within a database at a specific point in time is referred to as the database’s "instance."

Within a given database schema, the declarations of variables within its tables pertain to that specific database. The term "instance" in this context denotes the current values of these variables at a particular moment in time for that database.



The figure depicts the customer database in different instances. In instance 1, the customer with ID 1’s address is Delhi; however, it gets updated to Bangalore. Hence, in instance 2, we get a different value. Instances help in accommodating the updates taking place in the database

**3-tier Architecture**

This architecture separates the application into three logically distinct layers:

* **Presentation Layer** - Handles the user interface
* **Application Layer** - Manages business logic
* **Data Layer** - Manages data storage and processing

In this architecture, the end user has no idea about the existence of the database beyond the application server. Similarly, the database has no knowledge about any other user beyond the application.

Let us imagine an online shopping scenario. Whenever you go onto a platform like Amazon and perform an operation, such as searching for a product or adding an item to your cart, you interact with the frontend or User Interface (UI) of the application, which is represented by the **Presentation Layer**.

Any operation performed at the presentation layer gets sent to the next layer, the **Application Layer**, as a request. Upon receiving the request, the application layer interprets the request, formulates a corresponding database query, and sends it to the **Data Layer**. The data layer, which stores all the data, receives the query and executes it against the database.

Relevant information, such as product details and user account information, is retrieved, and the results are sent back to the application layer. The application layer processes the data further if necessary (e.g., formatting) and sends the response back to the presentation layer. The presentation layer then presents the results to the user, completing the process.

**Advantages of 3-Tier Architecture**

* **Scalability:** Enhanced scalability due to the distributed deployment of application servers. Each layer can be adjusted without altering other layers.
* **Security:** The client does not directly interact with the server, providing an added layer of security.
* **Modularity and Maintainability:** Maintenance is simplified due to the separation of responsibilities.
* **Performance:** Individual optimisation of presentation or application tiers is possible, leading to better performance.

**Disadvantages of 3-Tier Architecture**

* **Increased Complexity:** The introduction of an extra middle layer increases the complexity of the system. Communication points are also doubled.
* **Potential Latency Issues and Bottlenecks:** The added step of processing increases the possibility of latency and bottlenecks, as problems can arise in any layer at any time.
* **Longer Development Time:** Implementing three tiers with different logics and distributed responsibilities results in longer development time.
* **Resource Overhead:** Implementing an extra tier causes resource overhead for development, processing, and maintenance of the architecture.