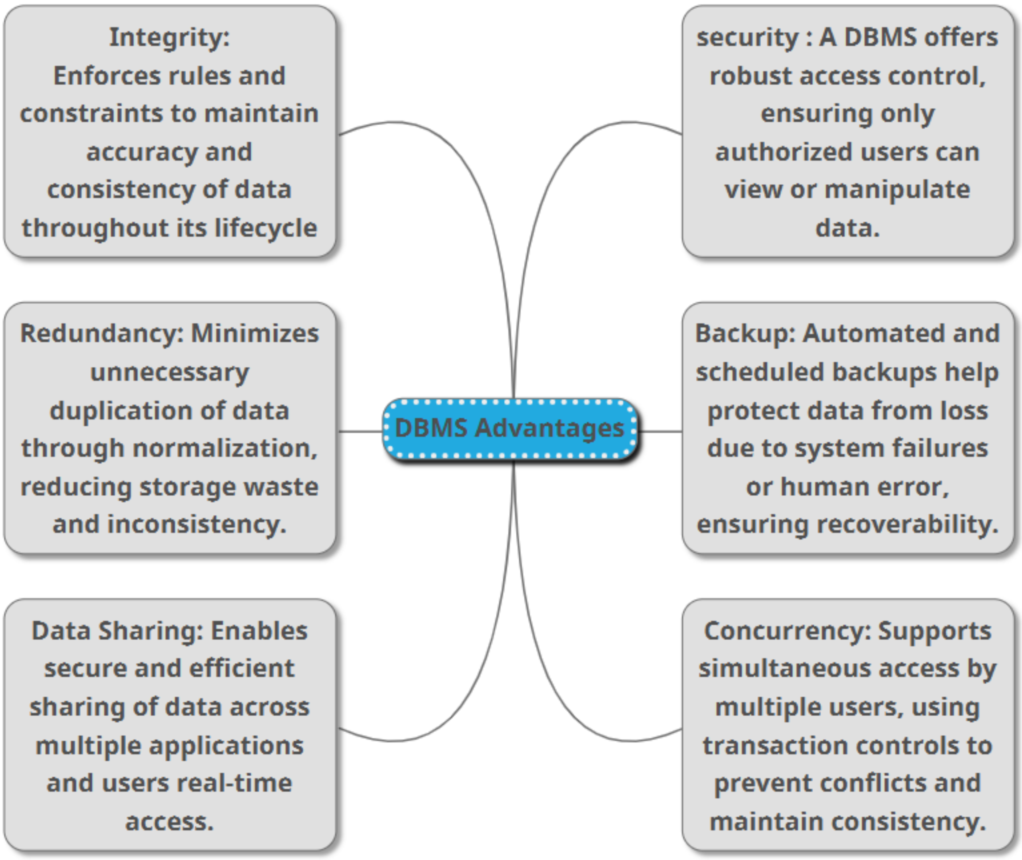
1. Comparison Assignment: Flat File Systems vs. Relational Databases

Category	Flat File Systems	Relational Databases
Data Organization	Stores information in plain files like .txt or .csv; data is typically unstructured and isolated.	Uses structured tables with predefined schemas, including columns and data types.
Duplication	Frequently contains repeated data due to lack of structure or relationship handling.	Minimizes data duplication through normalization and table references.
Handling Connections	Lacks built-in support for linking records across files; manual tracking is needed.	Supports complex relationships using primary and foreign keys.
Common Applications	Ideal for basic storage needs such as logs, simple lists, or flat data archives.	Used in systems requiring structured queries, like financial systems or online stores.
Limitations	Difficult to manage as data grows, lacks efficient search and update features, and risks inconsistency.	More complex to set up and manage but offers powerful querying and integrity enforcement.

2. **DBMS Advantages:**



DBMS Advantages

Integrity:
Enforces rules and constraints to maintain accuracy and consistency of data throughout its lifecycle

security : A DBMS offers robust access control, ensuring only authorized users can view or manipulate data.

Redundancy: Minimizes unnecessary duplication of data through normalization, reducing storage waste and inconsistency.

Backup: Automated and scheduled backups help protect data from loss due to system failures or human error, ensuring recoverability.

Data Sharing: Enables secure and efficient sharing of data across multiple applications and users real-time access.

Concurrency: Supports simultaneous access by multiple users, using transaction controls to prevent conflicts and maintain consistency.

Key Roles in a Database System:

🧠 System Analyst

Analyzes business requirements and translates them into technical specifications.
Acts as a link between users and the technical team.

🏗️ Database Designer

Creates the database structure, including tables, keys, and relationships.
Ensures the design supports scalability, efficiency, and data integrity.

🧑‍💻 Database Developer

Implements the database using SQL and other tools based on the design.
Develops stored procedures, triggers, and manages data logic.

🛠️ Database Administrator (DBA)

Manages database performance, security, and backup systems.
Ensures uptime, data integrity, and smooth operation of the database.

💻 Application Developer

Builds apps that interact with the database through APIs or queries.
Ensures smooth data flow between users and the backend.

📊 BI (Business Intelligence) Developer

Develops reports and dashboards for data analysis and decision-making.
Uses tools like Power BI or Tableau to deliver business insights.

4. Types of Databases:

Relational vs. Non-Relational Databases

- **Relational Databases**
  Store data in structured tables with predefined schemas (rows and columns).
  Use SQL for queries and support relationships via foreign keys.
  Example: MySQL, PostgreSQL
  Use Case: Banking systems, CRM tools
- **Non-Relational Databases**
  Store unstructured or semi-structured data in formats like key-value, document, graph, or column-based.
  Offer flexible schemas and are designed for scalability.
  Examples: MongoDB (document-based), Cassandra (column-based)
  Use Case: Social media feeds, IoT data, real-time analytics

---

Centralized vs. Distributed vs. Cloud Databases

- **Centralized Databases**
  All data is stored and maintained in a single location or server.
  Easier to manage but can become a bottleneck or single point of failure.
  Use Case: Small businesses, local inventory systems
- **Distributed Databases**
  Data is spread across multiple physical locations but appears as a single database.

Offers better performance, fault tolerance, and scalability.
Use Case: Global e-commerce platforms, telecom networks

- **Cloud Databases**
  Hosted on cloud platforms like AWS, Azure, or Google Cloud.
  Highly scalable, managed services with flexible pricing and global access.
  Use Case: SaaS applications, mobile backends, AI/ML data storage

5. Cloud Storage and Databases

What is Cloud Storage and How Does It Support Database Functionality?

Cloud storage is a service model where data is stored on remote servers accessed via the internet. It allows for scalable, on-demand data storage without managing physical hardware.
Databases in the cloud use this storage to persist data, enabling remote access, automatic scaling, and integration with cloud-based applications.

Advantages of Using Cloud-Based Databases

- **Scalability –** Automatically adjust resources based on demand, without manual intervention.
- **High Availability** – Built-in redundancy and failover options ensure minimal downtime.
- **Managed Services –** Providers handle backups, updates, and maintenance.
- **Global Access –** Accessible from anywhere with internet connectivity.
- **Cost Efficiency –** Pay-as-you-go pricing avoids large upfront infrastructure costs.

Examples: Azure SQL Database, Amazon RDS, Google Cloud Spanner

---

Disadvantages or Challenges with Cloud-Based Databases

- **Internet Dependency –** Requires a stable internet connection; downtime may impact access.
- **Security Concerns –** Although providers offer strong security, storing sensitive data off-premises raises compliance and privacy concerns.
- **Limited Control –** Some configuration and tuning options are restricted in fully managed services.

- **Cost Over Time –** While affordable initially, long-term costs can grow with usage or data egress fees.