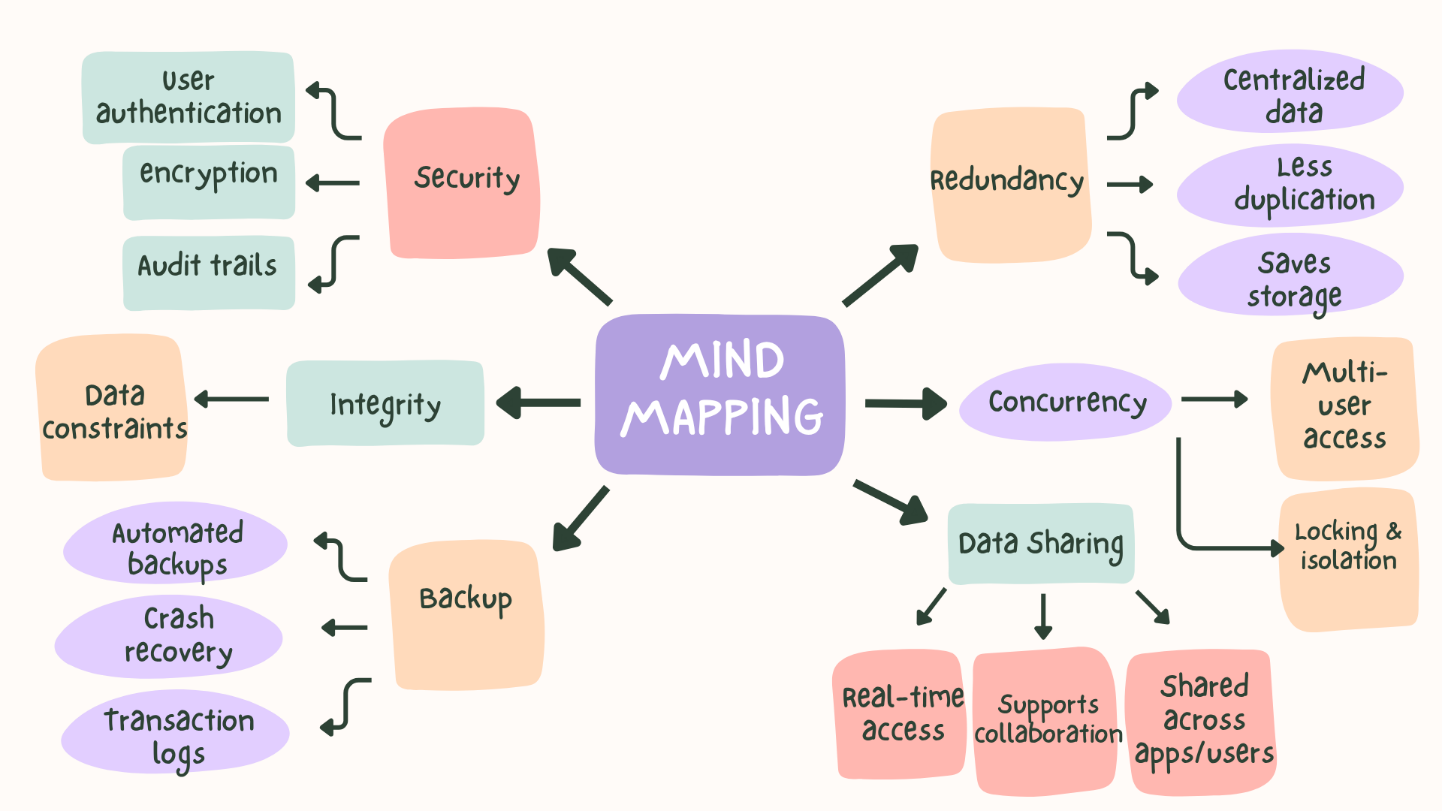
1. **Comparison: Flat File Systems vs. Relational Databases:**

|  |  |  |
| --- | --- | --- |
| Feature | Flat File Systems | Relational Databases |
| Structure | Data stored in a signal table or text file | Data stored in multiple related tables |
| Data Redundancy | duplicate data appears in multiple files | normalization minimizes redundancy |
| Relationships | all data kept separately | Supports relationships using primary and foreign keys |
| Example Usage | Small applications (configuration files) | Business applications (banking systems) |
| Drawbacks | Hard to update, lacks security and poor scalability | More complex setup, requires database software and management |

1. **DBMS Advantages – Mind Map:**
2. **Roles in a Database System:**

|  |  |  |
| --- | --- | --- |
|  | Role | Key Responsibilities: |
| System Analyst | Bridge between business needs and technical solutions | * Gathers and analyses business requirements from stakeholders * Identifies data needs and system workflows * Creates functional specifications and use cases |
| Database Designer | Architect of the database structure | * Designs the logical and physical data model * Applies normalization to reduce redundancy * Defines constraints, data types, and referential integrity rules |
| Database Developer | Builder of database logic and objects | * Writes and optimizes SQL scripts * Implements the database schema designed |
| Database Administrator (DBA) | Guardian of the database environment | * Installs, configures, and maintains DBMS software * Manages user accounts, roles, and security permissions * Performs backups, recovery, and disaster planning |
| Application Developer | Creator of software that uses the database | * Develops front-end or back-end applications * Writes code to query, insert, update, or delete data * Collaborates with Database Developers |
| BI (Business Intelligence) Developer | Data storyteller and analytics enabler | * Creates dashboards, reports, and visualizations * Transforms raw operational data into actionable business insights * Works with aggregated, historical, and dimensional data models |

1. **Types of Databases:**
2. **Relational vs. Non-Relational Databases:**

* **Relational Databases:**
* **Structure:** Data stored in tables (rows and columns) with predefined schemas.
* **Relationships:** Tables linked via keys (primary/foreign keys).
* **Query Language:** SQL (Structured Query Language).
* **Examples:** MySQL, PostgreSQL, Oracle, Microsoft SQL Server.
* **Non- Relational Databases:**
  + **Structure:** Flexible schemas; data stored as documents, key-value pairs, columns, or graphs.
  + **Scalability:** Designed for horizontal scaling (adding more servers).
  + **Examples:**
    - **MongoDB** → Document-based (stores JSON-like documents)
    - **Apache Cassandra** → Wide-column store (optimized for high write throughput & availability)

|  |  |  |
| --- | --- | --- |
|  | Relational Databases | Non-Relational Databases |
| **Data Model** | Tables | Document |
| **Schema** | Fixed | Dynamic |
| **Relationships** | Native support via foreign keys | Not natively supported |
| **Query Language** | SQL | Varies by type (e.g., MongoDB Query Language, CQL, etc.) |
| **Performance** | Optimized for complex queries & joins | Optimized for high-speed reads/writes on large datasets |
| **Examples** | MySQL, PostgreSQL, Oracle, SQL Server | MongoDB (document), Cassandra (column), Redis (key-value), Neo4j (graph) |
| **Best Use Cases** | Banking, ERP, payroll, inventory systems | Real-time analytics, IoT, content management, social media, gaming |

1. **Centralized vs. Distributed vs. Cloud Databases:**

* **Centralized Databases:** 
  + **Definition:** Stored and maintained on a single server or location.
  + **Control:** One point of management and access.
  + **Pros:** Simpler security, easier backup, consistent state.
  + **Cons:** Single point of failure; limited scalability.
* **Distributed Databases:**
  + **Definition:** Data spread across multiple physical locations (servers, regions, or countries), but logically interconnected.
  + **Types:** Homogeneous (same DBMS everywhere) or heterogeneous (different systems).
  + **Pros:** High availability, fault tolerance, local data access.
  + **Cons:** Complex synchronization, network dependency.
* **Cloud Databases:** 
  + **Definition:** Hosted on cloud platforms (e.g., AWS, Azure, Google Cloud); can be relational or NoSQL.
  + **Deployment Models:**
    - **DBaaS (Database as a Service):** Fully managed (e.g., Amazon RDS, Azure Cosmos DB)
    - **Self-managed on cloud VMs**
  + **Pros:** Elastic scaling, pay-as-you-go pricing, automatic backups, high availability
  + **Cons:** Ongoing costs, potential vendor lock-in, security considerations

|  |  |  |  |
| --- | --- | --- | --- |
|  | Centralized Databases | Distributed Databases | Cloud Databases |
| **Definition** | Data stored and managed on a  single server | Data stored across multiple physical locations | Database hosted, managed, and accessed over the internet via cloud platforms |
| **Architecture** | Single-node system | Multi-node, often geographically dispersed | Can be centralized or distributed—but hosted in the cloud |
| **Data Access** | All users connect to one central system | Users access local or remote nodes; system appears unified | Accessed via APIs or internet; location abstracted from user |
| **Scalability** | Limited | High | Very high |
| **Performance** | Fast for local users; slows with high load or remote access | Optimized for local access; latency depends on network | Depends on cloud provider; global CDNs and edge locations reduce latency |
| **Security** | Easier to secure (one location) | Harder (data in transit, multiple entry points) | Shared responsibility (provider secures infrastructure; user secures data/access) |
| **Examples** | Legacy ERP on a company server, local school database | Google Spanner, Apache Cassandra (multi-region), airline reservation systems | Amazon RDS, Google Cloud Fire store, Azure Cosmos DB, MongoDB Atlas |
| **Best Use Cases** | Small businesses, departmental systems, local applications | Global enterprises needing local data access and high availability (e.g., banking, telecom) | Startups, SaaS apps, scalable web services, remote teams |

1. **Cloud Storage and Databases:**

|  |  |
| --- | --- |
| Concept | Explanation |
| **What is Cloud Storage** | A service that stores data online, accessible via the internet (e.g., Google Drive, Amazon S3). |
| **How it supports databases?** | Cloud storage provides the infrastructure for hosting cloud databases, ensuring scalability, availability, and remote access. |
| **Advantages** | * **Rapid Deployment:**  Launch a production-ready database in minutes—no hardware setup or software installation. * **Elastic Scalability:**  Easily scale compute and storage up or down based on demand (e.g., handle Black Friday traffic spikes). * **Automated Management:** patching, backups, monitoring, and updates**.** * **Integrated Security:**  Built-in features like encryption at rest/in transit. |
| **Disadvantages** | * **Ongoing Costs:**  Can become expensive over time—especially with egress fees, premium features, or idle resources. * **Limited Control:**  Less access to OS, file system, or low-level tuning (in fully managed services like RDS). * **Network Dependency:** Performance and availability depend on internet connectivity and cloud provider uptime. * **Latency for Global Apps:**  Even with replication, cross-region queries may introduce latency unless using globally distributed databases (e.g., Spanner). |