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



**DBMS Advantages Mind Map:**



**Roles in a Database System**

1. 🧠 System Analyst

Role: Bridge between business needs and technical solutions.

Responsibilities:

Gather and analyze user requirements.

Define system functionality and technical specifications.

Create use-case scenarios and data flow diagrams.

Work with both users and developers to ensure the system meets business goals.

2. 🏗️ Database Designer

Role: Architect of the database structure.

Responsibilities:

Design the logical and physical structure of the database.

Create data models (ER diagrams, normalization).

Decide on tables, relationships, keys, and constraints.

Ensure data integrity and efficiency in design.

3. 👨‍💻 Database Developer

Role: Builds and implements the database systems.

Responsibilities:

Write SQL queries, stored procedures, triggers, and scripts.

Implement the designs created by the database designer.

Optimize query performance.

Develop logic for data manipulation, import/export, and automation.

4. 🛠️ DBA (Database Administrator)

Role: Oversees database operation and maintenance.

Responsibilities:

Install, configure, and upgrade database software.

Perform backups and recovery.

Manage user access, security, and permissions.

Monitor performance, troubleshoot issues, and ensure availability.

5. 📱 Application Developer

Role: Creates front-end or back-end apps that use the database.

Responsibilities:

Develop user interfaces and application logic.

Connect apps to databases via APIs or queries.

Ensure smooth interaction between the application and the database.

Implement error handling, transactions, and user interactions.

6. 📊 BI (Business Intelligence) Developer

Role: Transforms data into insights.

Responsibilities:

Create reports, dashboards, and visualizations.

Use tools like Power BI, Tableau, or SQL-based analytics.

Write complex queries for data analysis.

Help stakeholders make data-driven decisions.

**Types of Databases**

**Relational vs Non-Relational**

**Relational (RDBMS): Stores data in tables with rows and columns (e.g., MySQL, SQL Server).**

**Use Case: Structured data, complex queries (e.g., banking systems).**

**Non-Relational (NoSQL): Data is stored in flexible formats (e.g., MongoDB, Cassandra).**

**Use Case: Large-scale, unstructured data (e.g., social media, real- time analytics).**

**Centralized vs Distributed vs Cloud**

**Centralized: Single server for data storage. Simple but can have bottlenecks.**

**Distributed: Data spread across multiple servers. More scalable but complex (e.g., Cassandra).**

**Cloud Databases: Hosted on cloud platforms like AWS RDS, Google Cloud SQL. Scalable, flexible, and managed.**

**Cloud Storage and Databases**

**Cloud Storage: Remote storage accessible over the internet. Used by cloud databases for scalability and backup.**

**Cloud Database Advantages & Disadvantages**

**Advantages:**

**Scalability, cost-effective, high availability.**

**Managed services reduce administrative work.**

**Disadvantages:**

**Security concerns, latency, vendor lock-in.**

**Database Engines and Languages**

**What is a Database Engine**

**Software that manages database operations (e.g., SQL Server, MySQL).**

**Languages Used:**

**SQL Server: T-SQL**

**MySQL: ANSI SQL**

**Oracle: PL/SQL**

**PostgreSQL: ANSI SQL (with extensions)**

**Engine vs Language**

**Each database engine uses a specific language (e.g., T-SQL for SQL Server). While ANSI SQL is standard, each engine has its own extensions.**

**Database Migration**

**Can We Migrate Between Engines?**

**Yes, you can migrate databases (e.g., SQL Server to MySQL), but there are challenges:**

**Data types, stored procedures, triggers, and query syntax may need adjustments.**

**Consider compatibility, performance, and application integration before migration.**

**Logical vs Physical Schema**

**Logical Schema:Represents data structure and relationships (e.g., entities, attributes, and their connections).**

**Physical Schema:Defines how data is stored and accessed (e.g., tables, indexes, storage).**

**Difference:**

**Logical Schema: Focuses on data organization.**

**Physical Schema: Focuses on storage optimization.**

**Example:**

**Logical: Student entity with Student\_ID, Name, DOB.**

**Physical: Table students with student\_id INT, name VARCHAR, and indexed by student\_id.**