CO{ }ELINE
by Rihal

Database

Done by: Fatma Aal Abdulsalam

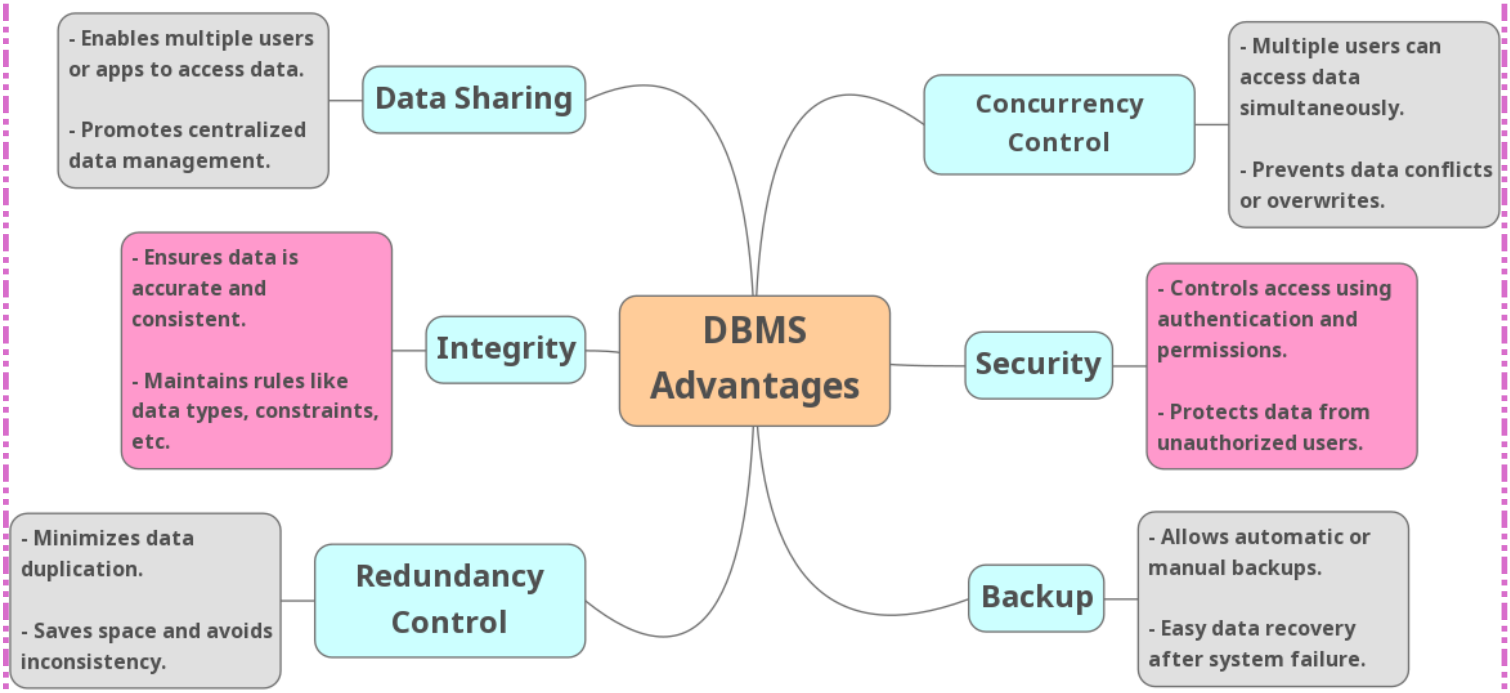
Team: Code Orbit

CODELINE
by Rihal

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

Feature	Flat File System	Relational Database (RDBMS)
Structure	Stores data in plain text files (e.g., CSV, TXT). Each file contains one table or list.	Stores data in structured tables with predefined schemas and relationships.
Data Redundancy	High – same data may be repeated in multiple files.	Low – normalized design reduces redundancy using relationships.
Relationships	No built-in relationships. Any linking must be done manually or via scripts.	Supports complex relationships using primary and foreign keys.
Example Usage	Simple address book, configuration files, log files, CSV exports.	Business applications, banking systems, inventory management.
Drawbacks	- No query language (must write code to filter/search) - Poor scalability - Prone to inconsistency	- Requires setup and a DBMS - May be overkill for very simple needs

2. DBMS Advantages – Mind Map



3. Roles in a Database System

1. System Analyst

- **Main Role**: Acts as a bridge between business needs and technical solutions.
- **Responsibilities**:
  - Gathers requirements from stakeholders.
  - Analyzes business processes and data needs.
  - Creates functional specifications for the database system.
  - Ensures the database aligns with business goals.

2. Database Designer

- **Main Role**: Designs the logical and physical structure of the database.
- **Responsibilities**:
  - Creates ER diagrams (Entity-Relationship models).
  - Defines tables, relationships, keys, and constraints.
  - Ensures normalization to reduce redundancy.
  - Plans for performance and scalability.

3. Database Developer

- **Main Role**: Builds and implements the database system.
- **Responsibilities**:
  - Writes SQL scripts to create tables, views, triggers, procedures, etc.
  - Develops complex queries and reports.
  - Works closely with the designer and application team.
  - Optimizes queries for performance.

4. Database Administrator (DBA)

- **Main Role**: Maintains and manages the database system.

- **Responsibilities**:
  - Installs and configures the DBMS (e.g., Oracle, MySQL, SQL Server).
  - Monitors performance and ensures uptime.
  - Performs backup and recovery.
  - Manages user access and security.
  - Applies patches and upgrades.

5. Application Developer

- **Main Role**: Builds the front-end or back-end applications that interact with the database.
- **Responsibilities**:
  - Writes code in programming languages (like Python, Java, PHP, etc.) to access and manipulate data.
  - Connects the UI with the database through APIs or SQL queries.
  - Ensures data flows correctly between the app and DB.

6. BI (Business Intelligence) Developer

- **Main Role**: Transforms raw data into meaningful insights and reports.
- **Responsibilities**:
  - Designs and builds dashboards, KPIs, and visual reports.
  - Uses tools like Power BI, Tableau, or SQL Server Reporting Services (SSRS).
  - Develops data models and performs ETL (Extract, Transform, Load) processes.
  - Supports decision-making through data analytics.

4. Types of Databases

1. Relational vs. Non-Relational Databases

Feature	Relational Databases (RDBMS)	Non-Relational Databases (NoSQL)
Structure	Data is stored in structured tables with rows and columns	Data is stored in flexible formats (documents, key-value, graphs, etc.)
Schema	Fixed schema with predefined data types	Dynamic schema – flexible and adaptable
Relationships	Strong use of foreign keys to define relationships	Not always relational – often denormalized
Examples	MySQL, PostgreSQL, Oracle	MongoDB, Cassandra, Couchbase, Redis
Best for	Structured, consistent data and complex queries	Large volumes of unstructured or semi-structured data

Use Case Examples:

- **Relational**: Banking systems, ERP, inventory management, student databases.
- **Non-Relational**:
  - **MongoDB**: Storing JSON-like documents in web applications.
  - **Cassandra**: Handling huge real-time data across distributed servers (e.g., Netflix, Facebook).

2. Centralized vs. Distributed vs. Cloud Databases

Type	Centralized	Distributed	Cloud
Definition	All data is stored in a single location/server	Data is spread across multiple servers/nodes	Hosted on cloud platforms over the internet
Access	Accessed from one location	Accessed from multiple locations	Accessed via the internet or cloud APIs

Management	Easy to manage but a single point of failure	Complex to manage but fault-tolerant	Managed by cloud provider (e.g., AWS, Azure)
Examples	Local MySQL server	Apache Cassandra, Google Spanner	Amazon RDS, Firebase, Google Cloud SQL

Use Case Examples:

- **Centralized**: Small office or school systems with all data on one server.
- **Distributed**: Global e-commerce platforms that need high availability and low latency.
- **Cloud**: SaaS applications, mobile apps, and scalable online platforms (e.g., ride-sharing, e-learning).

5. Cloud Storage and Databases

What is Cloud Storage?

Cloud Storage refers to the storage of data on remote servers that are accessible via the internet. These servers are managed by cloud service providers such as:

- **Amazon Web Services (AWS)**
- **Microsoft Azure**
- **Google Cloud Platform (GCP)**

Cloud storage eliminates the need for local storage hardware and allows users to store and access data from anywhere, at any time.

How Cloud Storage Supports Databases?

Cloud storage plays a critical role in supporting **cloud-based databases** in several ways:

- **Data Hosting**: Stores database files, backups, logs, and large datasets.
- **Scalability**: Easily expands storage capacity based on usage and needs.
- **Accessibility**: Enables global access to data through secure internet connections.
- **Reliability**: Offers built-in data replication, failover, and backup mechanisms.
- **Integration**: Seamlessly integrates with database services like Amazon RDS, Azure SQL Database, and Google Cloud Spanner.

Cloud storage provides the foundation that allows cloud-based databases to be fast, flexible, and scalable.

Advantages of Cloud-Based Databases

Cloud-based database services (e.g., **Amazon RDS**, **Azure SQL**, **Google Cloud Spanner**) provide numerous benefits:

1. **Scalability**
    - Instantly scale resources up or down to handle changing workloads.
2. **High Availability**
    - Automatic failover, multi-region replication, and minimal downtime.
3. **Cost-Efficiency**
    - Pay-as-you-go pricing eliminates upfront infrastructure costs.
4. **Managed Services**
    - Database maintenance (updates, backups, patches) is handled by the provider.
5. **Accessibility**
    - Accessible from anywhere with an internet connection and proper credentials.
6. **Security**
    - Data encryption, access control, and compliance certifications included.

Disadvantages / Challenges of Cloud-Based Databases

Despite the benefits, cloud-based databases come with some challenges:

1. **Internet Dependency**
   - Access and performance rely on a stable internet connection.
2. **Latency**
   - Data may take longer to retrieve if servers are geographically distant.
3. **Data Privacy and Compliance**
   - Storing sensitive data in the cloud may require strict regulatory compliance (e.g., GDPR, HIPAA).
4. **Vendor Lock-In**
   - Migrating databases between cloud providers can be complex and costly.
5. **Cost Management**
   - Unexpected usage spikes can lead to higher costs without proper monitoring.

Examples of Cloud Database Services

Cloud Provider	Database Service	Key Feature
Amazon AWS	Amazon RDS, DynamoDB	Relational and NoSQL support
Microsoft Azure	Azure SQL Database	Fully managed SQL database
Google Cloud	Cloud Spanner, Firestore	Globally distributed relational database