Database Report

Done By: Mariya Nabhan Allamki

Date: 30-7-2025

Flat File Systems

1. Structure

- A simple way to store data, where information is kept in a single, plain text file, often with records separated by delimiters like commas or tabs.
- Each record is typically a single line, and fields are separated by delimiters.

2. Data Redundancy

- Flat file systems often lead to duplicate data entries because there's no mechanism to enforce data integrity.(High Redundancy)
- Redundant data can consume more storage space and complicate data management.

3. Relationships

- Flat file systems do not support relationships between data entries. All data is treated independently.
- Without relationships, maintaining data consistency is challenging, as updates must be manually handled across duplicate entries.

4. Example Usage

- Often used for simple data storage needs, such as:
  - Comma-separated CSV file.
  - Name-and-address lists.
  - A sheet of paper with a name, address, and phone number.

5. Drawbacks

- Flat file systems become increasingly difficult to manage due to data volume grows.
- Limited capabilities for data querying and analysis compared to relational databases.
- Higher chances for errors and inconsistencies due to the absence of relationships and constraints.
- Flat files don't make it easy to avoid data duplication because they only contain one relational table.

Relational Databases

1. Structure

- Relational databases organize data into tables that consist of rows and columns.
- Each table represents a specific entity, with columns defining attributes and rows representing records.

2. Data Redundancy

- Relational databases minimize data redundancy through normalization, which ensures that each piece of data is stored only once. (**Low Redundancy**)
- This structure reduces the amount of storage needed and improves data consistency.

3. Relationships

- Relational databases allow for complex relationships between tables, such as:
    - **One-to-One**: Each record in one table corresponds to one record in another.
    - **One-to-Many**: A record in one table can relate to multiple records in another.
    - **Many-to-Many**: Multiple records in one table can relate to multiple records in another.
- Relationships help maintain data integrity through foreign keys and constraints.

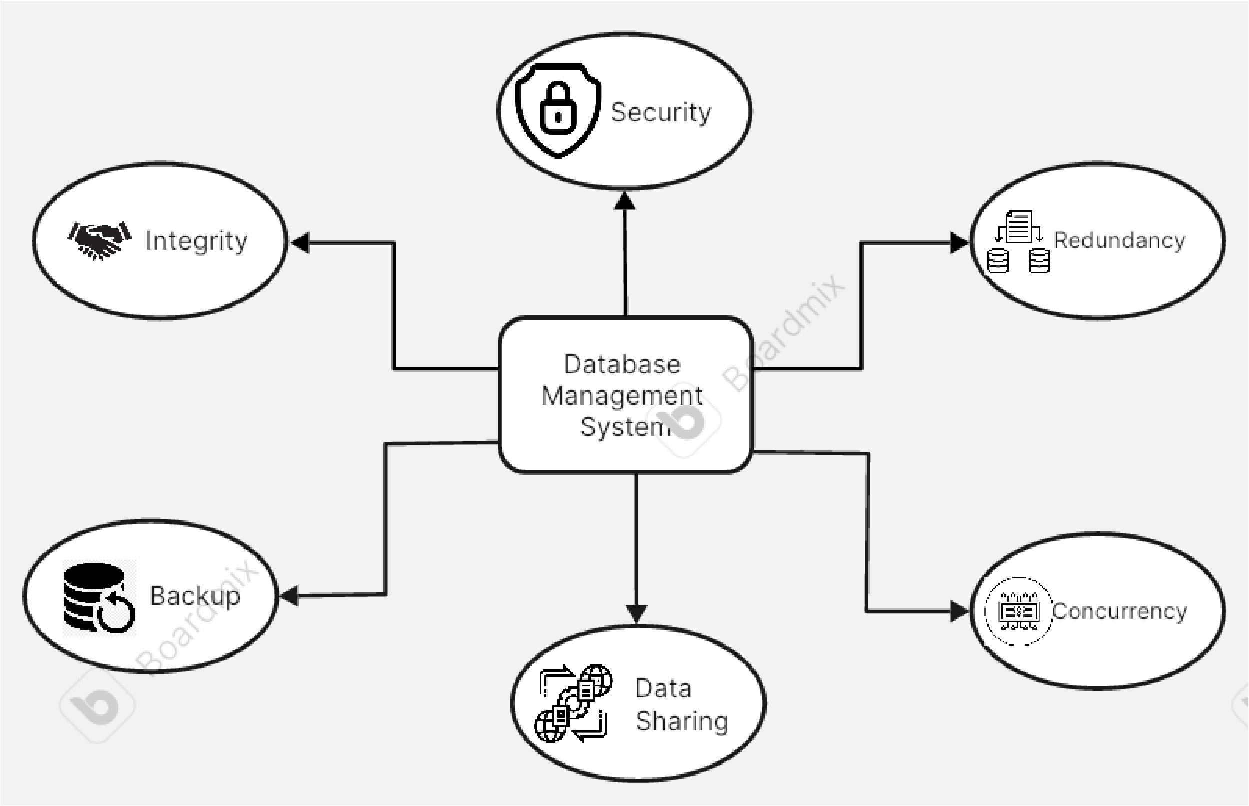
4. Example Usage

- Commonly used in various applications, such as:
    - Enterprise resource planning (ERP) systems.
    - Customer relationship management (CRM) systems.
    - E-commerce platforms for managing products, orders, and customers.

5. Drawbacks

- Setting up and maintaining a relational database can be complex and require specialized knowledge.
- For very large datasets or highly complex queries, performance can suffer compared to simpler data storage solutions.
- Licensing and resource requirements for relational database management systems (RDBMS) can be higher than for flat file systems.

DBMS Advantages



Roles in a Database System

1. System Analyst

- Analyzes business requirements and processes to determine data needs.
- Works with stakeholders to gather and document requirements.
- Evaluates existing systems and recommends improvements or new solutions.
- Acts as a liaison between technical teams and business users.

2. Database Designer

- Designs the structure of the database, including tables, fields, and relationships.
- Ensures the design supports data integrity, normalization, and efficient queries.
- Collaborates with system analysts to align design with user requirements.
- Creates data models and diagrams to visually represent the database structure.

3. Database Developer

- Implements the database design by writing SQL scripts and creating database objects (tables, views, stored procedures).
- Develops data import/export processes and manages data migration.
- Works on optimizing database performance and query efficiency.
- Collaborates with application developers to ensure seamless data integration.

4. Database Administrator (DBA)

- Manages and maintains the database systems, ensuring availability and performance.
- Performs regular backups, recovery, and disaster recovery planning.
- Monitors database security and implements access controls.
- Troubleshoots issues and performs maintenance tasks, such as updates and patches.

5. Application Developer

- Develops software applications that interact with the database.
- Writes code to perform CRUD (Create, Read, Update, Delete) operations on database records.

- Ensures applications are user-friendly and meet business requirements.
- Collaborates with database developers to optimize data access and performance.

6. BI (Business Intelligence) Developer

- Designs and implements data analysis solutions to support decision-making.
- Develops data visualization tools and dashboards to present insights.
- Works with data from various sources, including databases and external APIs.
- Analyzes trends and patterns in data to inform business strategies.

Types of Databases

Relational Databases

A type of database management system that stores data is a structured format using columns and rows. Its characteristic is strong consistency and data integrity and Relationships between tables are defined using foreign keys.

Examples: MySQL, PostgreSQL, Oracle Database.

Non-Relational Databases

Non-relational databases, also known as NoSQL databases, store data in various formats such as documents, key-value pairs, or graphs. They feature a schema-less design, allowing for flexibility in data storage, and are specifically designed for scalability to handle large volumes of unstructured data.

Examples: MongoDB, Cassandra.

Centralized vs. Distributed vs. Cloud Databases

Centralized Databases: Centralized databases store all data in a single location, typically on a central server. This setup makes them easier to manage and secure, but it also presents a risk as they can become a single point of failure.

Distributed Databases: Distributed databases spread data across multiple locations or servers, either geographically or within a network. This configuration enhances availability and fault tolerance, but it can also lead to challenges with data consistency and management.

Cloud Databases: Cloud databases are hosted on cloud infrastructure, allowing for scalable and flexible data storage. They are accessible from anywhere with internet connectivity and typically offer a pay-as-you-go pricing model, making them a cost-effective solution.

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

Use Case Examples

- **Relational Databases**: Accounting systems, where data integrity and relationships are crucial.
- **Non-Relational Databases**: MongoDB and Cassandra.
- **Centralized Databases**: Frequently used in small businesses where data management and security are simple, such as customer relationship management (CRM) systems.

- **Distributed Databases**: large organizations with multiple locations, such as global e-commerce platforms.
- **Cloud Databases**: Perfect for startups and businesses looking for scalable solutions without investing in physical infrastructure, such as data warehousing for analytics and reporting.

Relationship Between Cloud Storage and Databases

Cloud Storage is a model of computer data storage in which data, said to be on "**the cloud**", is stored remotely in logical pools and is accessible to users over a network, typically the Internet.

Advantages of Using Cloud-Based Databases

- Companies pay only for the storage they actually use, leading to operating expenses rather than capital expenses.
- Businesses can reduce energy consumption by up to 70%, contributing to sustainability.
- Organizations can choose between off-premises, on-premises, or hybrid cloud storage based on their needs.
- Object storage architecture provides built-in storage availability and data protection, reducing additional costs.
- Storage maintenance tasks, such as capacity expansion, are managed by the service provider.

Disadvantages with Cloud-Based Databases

- Data is stored offsite, limiting customization and control and Larger businesses with complex needs may find this restrictive.
- Vendor lock-in makes it challenging to switch providers and Medium-to-large businesses face complications due to large data volumes.
- Access to data relies on a stable internet connection; outages lead to downtime and Slow connections increase wait times for data access.
- Handing over data to third parties raises security concerns and Past incidents highlight potential data loss risks.
- Long-term contracts can be problematic if storage needs decrease and Businesses may end up paying for unused storage.

References:

- Bieri, C. (2021). An overview into the InterPlanetary File System (IPFS): use cases, advantages, and drawbacks. *Communication Systems XIV*, *28*.
- Folk, M. J., & Zoellick, B. (1992). *File structures* (Vol. 2). Reading: Addison-Wesley.
- Wu, J., Ping, L., Ge, X., Wang, Y., & Fu, J. (2010, June). Cloud storage as the infrastructure of cloud computing. In *2010 International conference on intelligent computing and cognitive informatics* (pp. 380-383). IEEE.