Database Search and Reporting
Task 1 Report

Name: Malik Khodari

Date: 16/6/2025

1- Comparison between Flat File Systems vs. Relational Databases

1. Structure

- *Flat File Systems*: Data stored in plain text files with records separated by delimiters (comma, tab).
- *Relational Databases*: Organized into tables with rows and columns, with defined schemas for data types and constraints.

2. Data Redundancy

- *Flat File Systems*: Commonly involves duplicated data, increasing storage and risk of inconsistency.
- *Relational Databases*: Minimized redundancy due to normalization (3NF, BCNF), improving consistency.

3. Relationships

- *Flat File Systems*: No inherent relationship; manual cross-referencing required.
- *Relational Databases*: Built-in relationships via foreign keys, enabling automated and consistent data linkage.

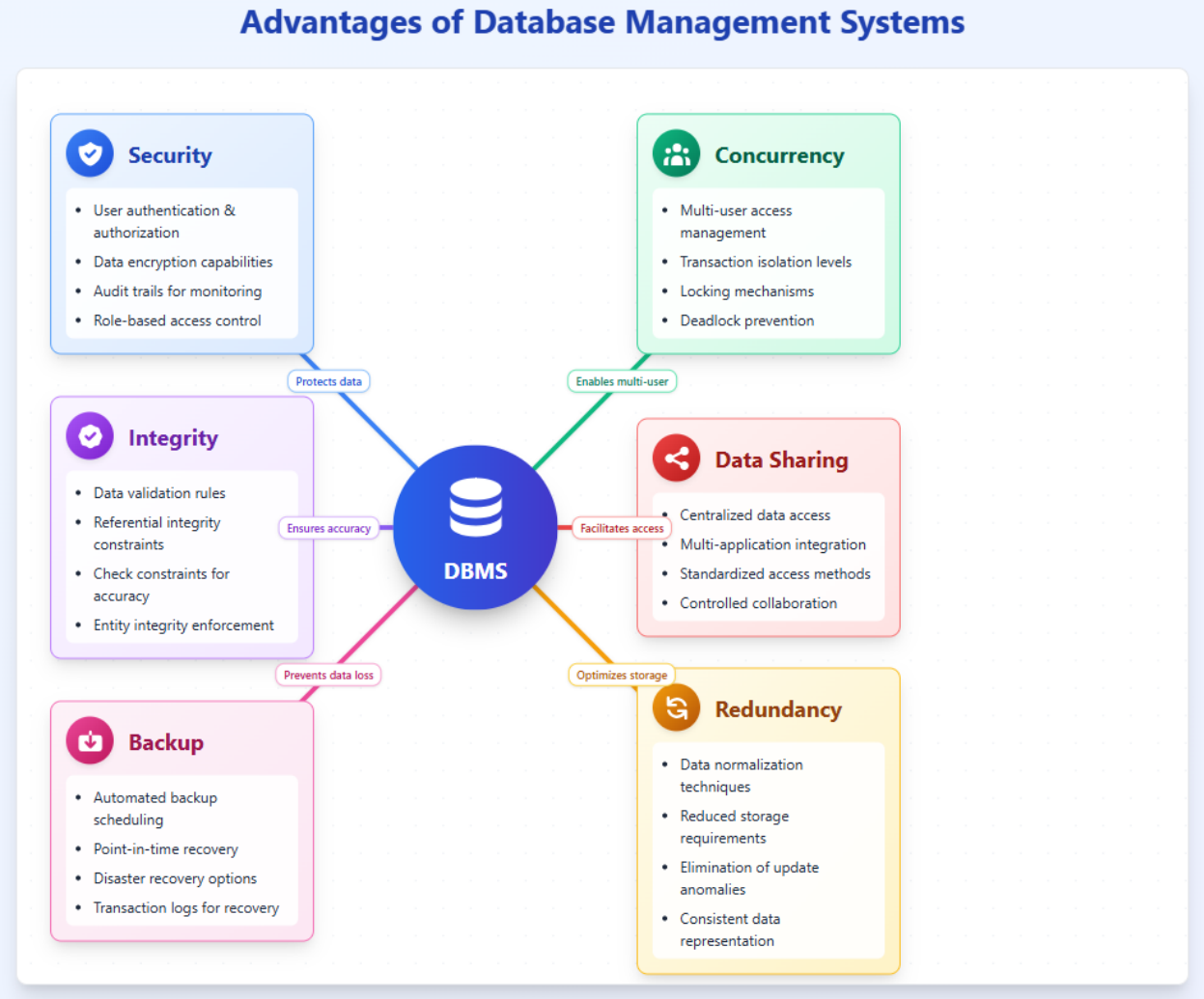
4. Example Usage

- *Flat File Systems*: Suitable for simple logs, configuration files, and lightweight data exchanges.
- *Relational Databases*: Ideal for complex, multi-user applications such as CRM systems, financial records, and inventory management.

5. Drawbacks

- *Flat File Systems*: Limited scalability; challenging to maintain integrity; not suited for concurrent access.
- *Relational Databases*: Higher complexity, initial setup cost, and resource-intensive for minimal or trivial data storage tasks.

2- DBMS Advantages – Mind Map



3- Roles in a Database System

1. System Analyst

Role: Bridge between business needs and technical solutions.
Responsibilities:

- Gather and analyze user requirements.
- Create system specifications and use cases.
- Define functional and non-functional requirements.
- Ensure the database system aligns with business objectives.
- Collaborate with stakeholders and developers during the planning phase.

---

2. Database Designer

Role: Architect of the database structure.
Responsibilities:

- Design conceptual, logical, and physical data models.
- Define relationships between entities.
- Apply normalization to reduce redundancy.
- Ensure data integrity, scalability, and performance.
- Produce ER diagrams and schema definitions.

---

3. Database Developer

Role: Builds and programs the database system.
Responsibilities:

- Write SQL scripts for data manipulation (DML) and definition (DDL).
- Implement stored procedures, triggers, and functions.
- Collaborate with designers and application developers.
- Optimise query performance.

- Ensure data logic follows the business rules.

---

4. Database Administrator (DBA)

Role: Overseer of database operation, security, and maintenance.
Responsibilities:

- Install, configure, and upgrade DBMS software.
- Manage user access and security.
- Monitor performance and storage.
- Set up and test backup and recovery strategies.
- Troubleshoot database issues and optimise uptime.

---

5. Application Developer

Role: Builds front-end or back-end software that interacts with the database.
Responsibilities:

- Design and code user interfaces or APIs.
- Integrate the database with application logic.
- Use SQL or ORM tools to retrieve and store data.
- Ensure seamless data exchange between the user and the DBMS.
- Test and debug application-database interactions.

---

6. BI (Business Intelligence) Developer

Role: Transforms raw data into insights for decision-making.
Responsibilities:

- Design and maintain data warehouses and data marts.
- Develop dashboards, reports, and KPIs using BI tools (e.g., Power BI, Tableau).
- Extract, transform, and load (ETL) processes.
- Perform data analysis and create visualisations.

- Support management with data-driven insights.

4- Types of Databases

Relational vs. Non-Relational Databases

Feature	Relational Databases (RDBMS)	Non-Relational Databases (NoSQL)
Data Structure	Tables (rows & columns), fixed schema	Flexible formats: key-value, document, column, or graph
Schema	Rigid, predefined	Dynamic, schema-less or flexible
Query Language	Structured Query Language (SQL)	Varies (e.g., MongoDB uses BSON/JSON-based queries)
Best For	Structured data, complex queries, ACID compliance	Semi-structured/unstructured data, scalability, big data
Examples	MySQL, PostgreSQL, Oracle, SQL Server	MongoDB, Cassandra, CouchDB, Redis

Use Cases:

- **Relational**: Banking systems, ERP, CRM, university records.
- **Non-Relational**: Real-time analytics, social media content, IoT data, document storage.

Centralized vs. Distributed vs. Cloud Databases

Type	Description	Use Cases	Examples
Centralized	All data is stored on a single server or location.	Small businesses, desktop apps	MS Access, single-instance MySQL
Distributed	Data is stored across multiple physical locations but appears as one system.	Global applications, real-time processing	Google Spanner, Apache Cassandra
Cloud	Hosted on cloud infrastructure; scalable and accessible via the internet.	SaaS, scalable web apps, remote access	Amazon RDS, Azure SQL, Firebase

Use Case Highlights:

- **Centralized DB**: A local library system storing books and users on one server.
- **Distributed DB**: E-commerce platforms like Amazon needing low-latency global access.
- **Cloud DB**: Startups deploying apps using managed services like Google Firebase for scalability.

5- Cloud Storage and Databases

What is Cloud Storage?

Cloud storage is a model of data storage in which digital data is stored on remote servers accessed via the internet. These servers are maintained by a cloud service provider (e.g., Amazon Web Services, Microsoft Azure, Google Cloud Platform). Cloud storage provides scalable, on-demand storage infrastructure without the need for physical hardware on-site.

How Does It Support Database Functionality?

Cloud storage supports database systems by:

- Hosting the **underlying data** used by cloud-based databases.
- Offering **scalable capacity**, allowing databases to grow as needed.
- Enabling **backup and disaster recovery**, ensuring data safety and availability.
- Supporting **high availability zones** for databases to run with minimal downtime.
- Facilitating **real-time access and global distribution** of data, which is essential for distributed and replicated databases.

Cloud-based databases like **Amazon RDS**, **Azure SQL**, and **Google Cloud Spanner** are built on top of cloud storage to offer full database functionalities (queries, transactions, indexing, etc.) with added flexibility and scalability.

Advantages of Cloud-Based Databases

Advantage	**Description**
Scalability	Auto-scale compute and storage resources to match demand (horizontal and vertical scaling).
Cost Efficiency	Pay-as-you-go models reduce upfront hardware/software costs.
Availability & Reliability	High availability via data replication across multiple zones/regions.
Managed Services	Providers handle patching, backups, security, and performance monitoring.
Remote Accessibility	Accessible globally from any device with internet connection.
Integration	Easily integrates with cloud analytics, AI/ML tools, and app development platforms.

Disadvantages or Challenges of Cloud-Based Databases

Challenge	Description
Data Security and Privacy	Sensitive data stored offsite may raise regulatory and privacy concerns (e.g., GDPR, HIPAA).
Downtime Risks	Dependence on internet connectivity and cloud service uptime.
Vendor Lock-In	Difficult migration between cloud providers due to proprietary systems and tools.
Latency Issues	Remote access may introduce latency, especially for high-speed transaction systems.
Ongoing Costs	Misconfigured services or unexpected usage can lead to high operational costs over time.

Reference:

- Connolly, T., & Begg, C. (2015). *Database systems: A practical approach to design, implementation, and management* (6th ed.). Pearson Education.
- Elmasri, R., & Navathe, S. B. (2016). *Fundamentals of database systems* (7th ed.). Pearson Education.
- Rittinghouse, J. W., & Ransome, J. F. (2016). *Cloud computing: Implementation, management, and security* (2nd ed.). CRC Press.
- Coronel, C., & Morris, S. (2019). *Database systems: Design, implementation, & management* (13th ed.). Cengage Learning.