Task 1

1. Comparison Assignments

What is a Relational Database?

A flat file, also known as a text database, stores data in plain text format and is organized as a single table with no relationships between tables.

What is a Relational Database?

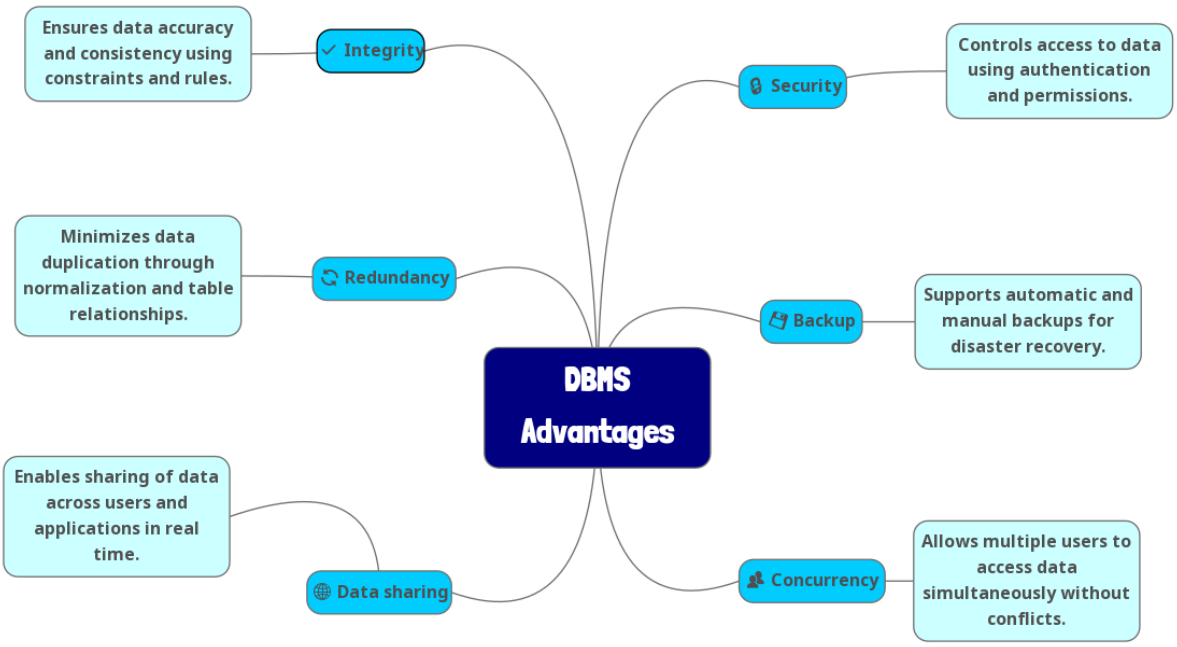
A relational database is a type of database that stores and organizes data in a collection of tables. These tables are related to each other through the use of a common field known as a primary key. Relational databases are used to store, organize and retrieve data quickly and efficiently.

DIFFERENCE BETWEEN RELATIONAL DATABASE VS FLAT FILE

	Relational Database	Flat File
STRUCTURE	Data is organized into multiple related tables with rows and columns.	Data is stored in a single table or text file, usually line by line.
DATA REDUNDANCY	Low redundancy due to normalization and relationships.	High redundancy as data is repeated across records.
RELATIONSHIPS	Supports complex relationships using primary and foreign keys.	No inherent relationships between data records.
EXAMPLE USAGE	Commonly used in applications where data integrity and complex relationships are important, such as enterprise applications, e-commerce systems, and data analytics.	Ideal for small applications, simple data storage needs, or when data relationships are not complex.

DRAWBACKS	Requires a DBMS, more complex setup, and higher cost.	Not scalable, difficult to query and maintain with large datasets.

2. DBMS Advantages Mind Map



3. Roles in a Database System

Database Administrator
Database administrators (DBAs) are responsible for the management and maintenance of a company database. They design, write and maintain computer database systems to ensure that the right person retrieves the right information at the right time. A professional DMA will always keep the database up and running smoothly throughout the day.

Database Designer
A database designer creates and manages detailed data models, including tables, views, and procedures, to support efficient and scalable databases. They design schemas, manage user access, monitor performance, and work with large datasets. The role requires technical skills in data modeling and programming, along with strong communication, problem-solving, and time management abilities.

Database Developer

A database developer is responsible for the performance, integrity, and security of databases, often handling planning, development, and troubleshooting tasks. They work closely with business clients to align database functions with business goals and are involved in tasks like ensuring data security, optimizing systems, creating scripts and procedures, and writing technical documentation. Essential skills include SQL, Microsoft Office, communication, analytical thinking, and relationship-building. A bachelor's degree in computer science or a related field is typically required, with experience in systems like Oracle, SQL Server, or MySQL adding to their professional value.

System Analyst

A system analyst acts as a bridge between business needs and technical solutions. They study current systems, gather and analyze requirements, and propose improvements or new systems to meet organizational goals. In the context of databases, they help define data requirements and workflows to guide the design and development of database systems.

Application Developer

Application developers build software applications that interact with the database. They write code in languages like C#, Java, or Python to create user interfaces and logic that read from or write to databases. Their focus is on integrating databases with business applications to deliver user-friendly, functional tools.

BI (Business Intelligence) Developer

A BI developer focuses on transforming raw data into meaningful insights through dashboards, reports, and visualizations. They use tools like Power BI, Tableau, or SQL-based reporting systems to help organizations make data-driven decisions. BI developers often work closely with data warehouses and perform data analysis and aggregation.

Additional Research Topics

1. Types of Databases

➢ Relational vs. Non-Relational Databases

Aspect	Relational (SQL)	Non-Relational (NoSQL)
Structure	Tables with rows/columns and strict schema.	Flexible schemas (documents, key-value, graphs).
Scalability	Vertical scaling (adding power to a server).	Horizontal scaling (adding servers).
Use Cases	Financial systems, inventory management.	Real-time apps, IoT, big data (e.g., social media feeds).
Examples	MySQL, PostgreSQL.	MongoDB (document), Cassandra (wide-column), Redis (key-value).

❖ Centralized vs Distributed vs Cloud Databases

Aspect	Centralized System	Distributed System	Cloud Databases
Control	Centralized control and authority	Decentralized control and authority	Managed by the provider (e.g., AWS, Azure), with user-configurable settings via dashboards/APIs
Resource Management	All resources managed centrally	Resources distributed across multiple nodes	Distributed but managed by the cloud provider (auto-scaling, load balancing)
Communication	Communication flows to central node	Direct communication between nodes	Hybrid: Nodes communicate via cloud infrastructure (e.g., virtual networks, APIs)

Fault Tolerance	Single point of failure	Redundancy, less vulnerable to single points of failure	High redundancy (automatic backups, multi-region replication)
Scalability	Limited scalability due to centralization	Highly scalable, new nodes can be added easily	Elastic scalability (auto-scaling based on demand)
Complexity	Relatively simpler to manage	More complex to manage	Low operational complexity (managed services), but requires cloud expertise
Use Case	Small business databases, legacy systems, single-branch banking.	Blockchain, IoT networks, global apps (e.g., Netflix)	SaaS applications, e-commerce platforms (e.g., Shopify), serverless architectures.

➢ Cloud Storage and Databases

What is Cloud Storage?

Stores data in remote servers (e.g., Amazon S3, Google Cloud Storage).

Relation to Databases: Cloud databases (e.g., Azure SQL) use cloud storage for data persistence.

Advantages of Cloud Databases

Scalability: Auto-scaling based on demand.

Cost: Pay-as-you-go pricing.

Managed Services: Automated backups, updates, and security.

Disadvantages

Latency: Network-dependent performance.

Security Risks: Data breaches in shared environments.

Vendor Lock-in: Difficulty migrating between providers.

❖ Database Engines and Languages

What is a Database Engine?

- Software component that stores, retrieves, and manages data (e.g., MySQL uses InnoDB).

Examples & Languages

Engine	Language	Key Feature
SQL Server	T-SQL	Proprietary extensions (e.g., TOP clause).
Oracle	PL/SQL	Advanced transaction control.
PostgreSQL	PL/pgSQL + ANSI SQL	JSON support, extensibility.
MySQL	ANSI SQL	Simplicity, open-source.

Relationship Between Engine & Language

- Each engine supports specific SQL dialects (e.g., T-SQL for SQL Server).
- Cross-Engine Compatibility: ANSI SQL works across engines but with syntax variations.

❖ Can We Transfer a Database Between Engines?

Is Migration Possible?

- Yes, but challenges exist (e.g., SQL Server → PostgreSQL).
- Tools: AWS Database Migration Service, manual scripts.

Challenges

- Syntax Differences: Stored procedures, triggers.
- Data Types: DATETIME in SQL Server vs. TIMESTAMP in PostgreSQL.
- Features: Proprietary functions (e.g., SQL Server's ROW_NUMBER()).

Considerations

- Schema compatibility.
- Data validation post-migration.
- Rebuilding non-standard features.

❖ Logical vs. Physical Schema

Logical Schema

- Definition: Abstract design (entities, relationships, constraints).
- Example:
  Student (StudentID, Name, EnrollmentDate)
  Course (CourseID, Title, Credits)
  Enrollment (StudentID, CourseID, Grade)

Physical Schema

- Definition: Implementation-specific details (storage, indexing).
- Example:

```
CREATE TABLE Student (
  StudentID INT PRIMARY KEY,
  Name VARCHAR(50),
  EnrollmentDate DATE
) STORED AS InnoDB;
```

Key Differences

- Logical: Focuses on *what* data is stored.
- Physical: Focuses on *how* data is stored/accessed.

Why Both Matter

- Logical schema ensures business rules; physical schema optimizes performance.