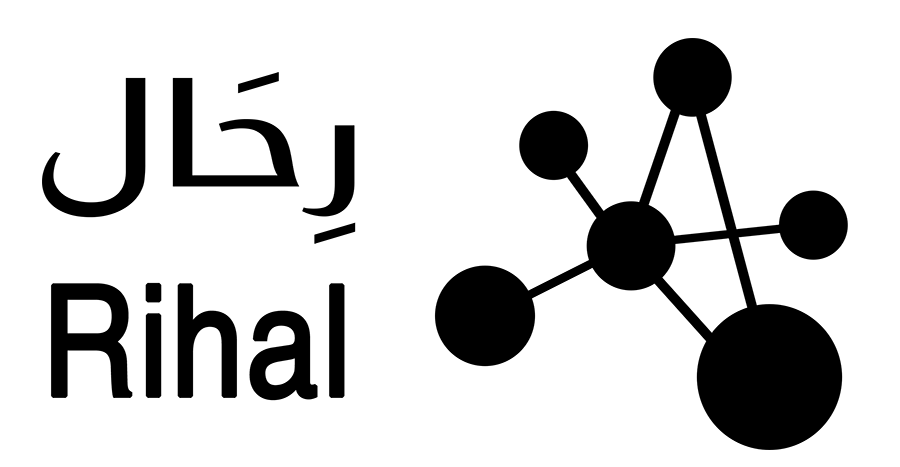
**1.0 Flat file systems vs relational Databases**



Database Report  
Qais Al Raisi

**1.1 What is a Database?**

* A database is an organized collection of data typically stored on a computer system; this data can be in the form of text, numbers, images, videos and files.
* A **database management system** is a software that manages a database. It allows users to create, read, update, and delete data. E.g.: Microsoft SQL, Oracle database and MongoDB.
* Data in a database is stored into tables, with rows representing individual records and columns representing specific attributes.
* columns.

**1.2 Flat File Systems:**

* A flat file system is a simple database system that stores data on a single, two-dimensional table.
* A flat file system lacks relationships between different records.
* They are typically stored as plain text files using delimiters to separate fields.
* **Data redundancy** is the duplication of data across multiple records within a single file or across multiple files. This occurs in flat file systems as they typically lack the structured relationships found in relational databases. This can cause issues such as wasted storage space, increased risk of data inconsistencies and potential difficulties in maintaining data integrity.
* **Benefits**:
  + Flat file systems are easy to create, read and modify and do not require database management software.
  + It can be easily shared, moved or backed up as a single file.
  + Require no setup.
  + Fast for small datasets.
* **Drawbacks:**
  + Data redundancy
  + No relationships between data meaning its harder to model real-word entities that are connected.
  + Poor scalability, they are useful when dealing with small datasets but not optimal when dealing with a large dataset that has complex relationships within the data.
  + They have limited security as there are no user permissions or access control since they are usually just files on a disk.
* **Example Usage:**
  + CSV files for data storage and transfer.
  + Configuration files. (e.g. env)
  + Log files

**1.3 Relational Databases:**

* A relational database is a structured database system that stores data in multiple related tables made up of records and attributes.
* A relational database supports relationships between different records using keys (primary and foreign keys).
* Data is typically managed using SQL (Structured Query Language) and stored in a Database Management System (DBMS) such as MySQL, PostgreSQL, or Oracle.
* **Benefits:**
  + Data integrity is maintained by separating data into normalized tables and linking them through relationships, which minimizes data redundancy. This ensures consistent, accurate, and non-repetitive data, making updates and maintenance easier and more efficient.
  + They support complex relationships, making it easier to model real-world systems accurately.
  + Enable powerful queries using SQL to search, sort, filter, and join data across tables.
  + Scalable and efficient for large datasets, with indexing and optimization features.
  + Offer advanced security, including user access controls and permission management.
* **Drawbacks:**
  + More complex setup, requiring a database management system and schema design.
  + Steeper learning curve for beginners or non-technical users.
  + Schema changes can be difficult once data and relationships are established.
  + Overhead for simple applications that don’t require complex data relationships.
* **Example Usage:**
  + Relational databases used to manage user accounts, orders, and inventory in e-commerce platforms.
  + Banking systems that organize customers, accounts, and transactions in interconnected tables.

| **Category** | **Flat File System** | **Relational Database** |
| --- | --- | --- |
| **Structure** | Single table (text file), no enforced schema | Multiple related tables with a defined schema |
| **Data Redundancy** | High — data often repeated across records | Low — normalized data reduces duplication |
| **Relationships** | Not supported | Fully supported using keys |
| **Example Usage** | CSV files, config files (.env), simple logs | Web apps, school systems, hospital databases, banking systems |
| **Benefits** | Easy to create and edit, portable, no setup needed | Data integrity, supports complex queries, secure, multi-user support |
| **Drawbacks** | No data integrity, not scalable, lacks security and linking | More complex setup, requires DBMS, harder to learn |

**Figure 1: Flat File vs. Relational Database**

**2.0 Roles in a Database System**

**System Analyst:**

* **Role:** Acts as a bridge between business users and technical teams.
* **Responsibilities:**
  + Gathers and analyzes business requirements.
  + Defines system specifications and workflows.
  + Ensures the database solution aligns with business needs.
  + Documents processes and communicates with stakeholders.

**Database Designer**

* **Role:** Designs the structure of the database.
* **Responsibilities:**
  + Creates data models and schemas.
  + Defines tables, attributes, keys, and relationships.
  + Ensures data normalization and integrity rules.
  + Plans how data will be stored and accessed efficiently.

**Database Developer**

* **Role:** Builds and implements the database according to the design.
* **Responsibilities:**
  + Writes SQL queries, stored procedures, and triggers.
  + Develops database scripts and programs.
  + Tests database functionality and performance.
  + Integrates the database with applications.

**Database Administrator (DBA)**

* **Role:** Manages and maintains the database system.
* **Responsibilities:**
  + Installs, configures, and upgrades DBMS software.
  + Performs backups, recovery, and security management.
  + Monitors performance and tunes database efficiency.
  + Manages user access and enforces policies.

**Application Developer**

* **Role:** Develops software applications that interact with the database.
* **Responsibilities:**
  + Designs and codes user interfaces and business logic.
  + Implements database connectivity and data handling.
  + Collaborates with DB developers and analysts.
  + Tests and debugs applications.
* **BI (Business Intelligence) Developer**
* **Role**: Creates reports and dashboards to help businesses understand their data and make better decisions.
* **Responsibilities**:
  + Collects and organizes data from different sources.
  + Cleans and prepares data for analysis.
  + Builds visual reports and dashboards.
  + Works with teams to show useful trends, patterns, and performance.

**3.0 Types of databases**

**3.1 Relational vs. Non-relational databases**

**Relational databases:**

Relational databases store data in structured tables with rows and columns. They use Structured Query Language (SQL) for querying and managing data. These databases are ideal for data that has clear relationships and a fixed schema.

**Examples**: MySQL, PostgreSQL, Oracle

**Non-Relational Databases (NoSQL):**

Non-relational databases store data in more flexible formats such as documents, key-value pairs, wide-columns, or graphs. They are well-suited for handling unstructured or semi-structured data, and they scale easily for large or fast-changing datasets

**Examples**:

* MongoDB – Document-based NoSQL database
* Cassandra – Wide-column store optimized for large-scale distributed data

**3.2 Centralized, Distributed and Cloud Databases**

**Centralized Database:**

A centralized database stores all data in a single location or server. It is easy to manage but can be a single point of failure.

Example: Small local businesses or schools using a single-location library management system.

**Distributed Database:**

A distributed database stores data across multiple servers, often in different geographical locations. This setup offers improved performance, fault tolerance, and availability.

Example: Large multinational companies needing fast and reliable access across regions.

**Cloud Database:**  
A cloud database is hosted and maintained on cloud computing platforms like AWS, Microsoft Azure, or Google Cloud. It offers automated backups, updates, scalability, and high availability.

**Example:** Web and mobile applications, SaaS platforms, and modern enterprise systems.

**5. Cloud Storage and Databases**

**5.1 What is Cloud Storage and How Does It Support Database Functionality?**

Cloud storage is an online service that allows users and systems to store data remotely on internet-connected servers. In the context of databases, cloud storage supports functionalities such as data persistence, backups, and data sharing across multiple services or applications. It enables modern databases to scale and perform without relying on physical hardware.

**5.2 Advantages of Cloud-Based Databases**

* **Scalability:** Resources can be scaled up or down automatically based on demand.
* **Security:** Includes built-in encryption, access control, and monitoring tools.
* **Maintenance-Free:** Cloud providers handle updates, patches, backups, and infrastructure.
* **Global Accessibility:** Accessible from anywhere with internet connectivity.
* **Cost-Effective:** Offers pay-as-you-go pricing, reducing upfront hardware costs.

**Examples of Cloud-Based Databases:**

* Amazon RDS
* Google Cloud Spanner
* Azure SQL Database

**5.3 Disadvantages or Challenges of Cloud-Based Databases**

* **Internet Dependency:** Requires a stable internet connection to access data.
* **Data Privacy Concerns:** Sensitive data stored off-site may raise compliance issues.
* **Ongoing Costs:** High usage can lead to increased operational costs over time.
* **Limited Low-Level Control:** Less flexibility in configuring the environment compared to self-hosted databases.

**Index**

* **Record**: a single row of data in a table, it represents one item, entity or instance and contains fields that store specific pieces of information about that item.
* **Attribute:** a single column in a table, it describes a property of the entity represented by the table and defines what kind of data each record can hold.
* **Table:** a table is a structured collection of related data organized into rows and columns.