introduction to database

The history of databases reflects the evolution of data storage and management systems, driven by technological advancements and the growing need for efficient handling of large data sets.

Flat File Systems:

- ✓ Somewhere in 1970s.
- ✓ Flat-file system => one file with the entire information.
- ✓ A system where data is stored in plain text files, with each line in the file representing a record, and fields in each record being separated by a specific delimiter (such as commas, tabs, or spaces). These files do not contain structured relationships between records, as relational databases do.

Drawback:

- ✓ Information was repeated and inconsistent.
- ✓ Data Redundancy leads to => data inconsistency.

Examples:

ID, Name, Age, Department

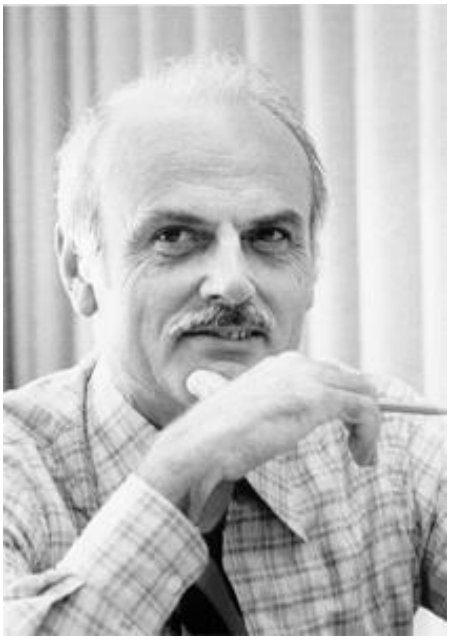
1, Alice, 28, HR

2, Bob, 34, IT

3, Charlie, 25, Marketing

Patient Id	Name	D.o.B	Gender	Phone	Doctor Id	Doctor	Room
134	Jeff	4-Jul-1993	Male	7876453	01	Dr Hyde	03
178	David	8-Feb-1987	Male	8635467	02	Dr Jekyll	06
198	Lisa	18-Dec-1979	Female	7498735	01	Dr Hyde	03
210	Frank	29-Apr-1983	Male	7943521	01	Dr Hyde	03
258	Rachel	8-Feb-1987	Female	8367242	02	Dr Jekyll	06

Dr. E. F. Codd



The major problem faced in the field of databases before the introduction of the relational model was the complexity of managing large, shared **databanks**. These databanks were typically stored in hierarchical or network models.

Shared databanks, in the context of early database systems, refer to large, centralized collections of data that were intended to be accessed and used by multiple users, departments, or applications within an organization. The concept originated in the 1960s and 1970s when organizations began to centralize their data storage to facilitate data sharing and improve efficiency in data management. Dr. E. F. Codd introduced the concept of relational model. He was mathematician

Relational Theory

The **relational model** is based on the idea of representing data in a **tabular form**, where:

- Data is stored in tables, called **relations**, which consist of rows and columns.
- Each **row** (also known as a tuple) represents a single record in the table.
- Each **column** (attribute) represents a specific data field.

The introduction of the relational model addressed the issues of **data sharing**, **scalability**, and **complexity**, transforming databases into more efficient, flexible, and reliable systems.

D	Database concept		D	Advanced SQL
1	Introduction to DB concept, DB Design and ERD.	✓ In general, what is idea behind DB. ✓ What is the benefit of having good DB design. (Some people start working on SQL server directly and this is wrong!). ✓ DB is like building architecture. ✓ What drives people to draw ERD and spend so much time on it?!	5	✓ **Database Schema**
	File Based system	✓ What is File System. ✓ Why people worked on it. ✓ Are all programs built on a database?		✓ **Ranking function** ✓ **Aggregation function** ✓ **Grouping** ✓ **Execution order** ✓ **Transaction** ✓ **ACID properties**
2	Database Design→ ERD, Database Mapping.	✓ Database Design Process Flow chart ✓ Is an ERD a database? ✓ This ERD needs to go through database mapping. ✓ DB Mapping: set of rules apply to ERD to display → tables	6	✓ **Variable:** Global, Local
	Database Normalization and SQL (Physical Schema implementation)	✓ Purpose of Normalization, Types of Normal Forms. ✓ if you work in SQL server, or Oracle you need to know SQL language. ✓ This is the language that use to contact with DB. ✓ DDL, DML, DQL, DCL, TCL.		✓ **Control of flow** (If, while continue, break, begin, end, if exists, if not exists). ✓ **Choose, wait-for** →this is for them to search :)
	Create Database [SQL Server]	✓ Start to create DB. ✓ There is no relation between DB engine		✓ **Functions:** Built-in functions (Scaler), User defined functions → Scalar, Inline,

		and the type of the language, even the type of the application. ✓ How can we backup & restore database to use.		Multi statement (Insert statement based on select) ✓ **Cursor**
3	**Joins, DB integrity**	✓ Very important because we are dealing with numbers of tables connecting with each other. ✓ To obtain information from the table we need to write query. ✓ Types of joins.	7	✓ **View and index**
	Database Constraints	✓ A database is not just a collection of related tables; it's also surrounded by several constraints. ✓ It is important because it ensures data integrity. ✓ DB integrity (correct → meet business rules.) ✓ To do this we use (Database Constraints: Domain integrity, Entity integrity, Referential integrity OR Database objects: Rules and triggers)		
4	**Subqueries**		8	✓ **Stored Procedure**
	Union			✓ Trigger ✓ SQL jobs ✓ **Performance optimization technique** → Indexing → Use Query Execution Plans →Partitioning table →Denormalization
	Users	✓ DB analyst ✓ DB designer ✓ DB developer ✓ DB admin ✓ Application developer ✓ BI developer		

Basic definition

Database:

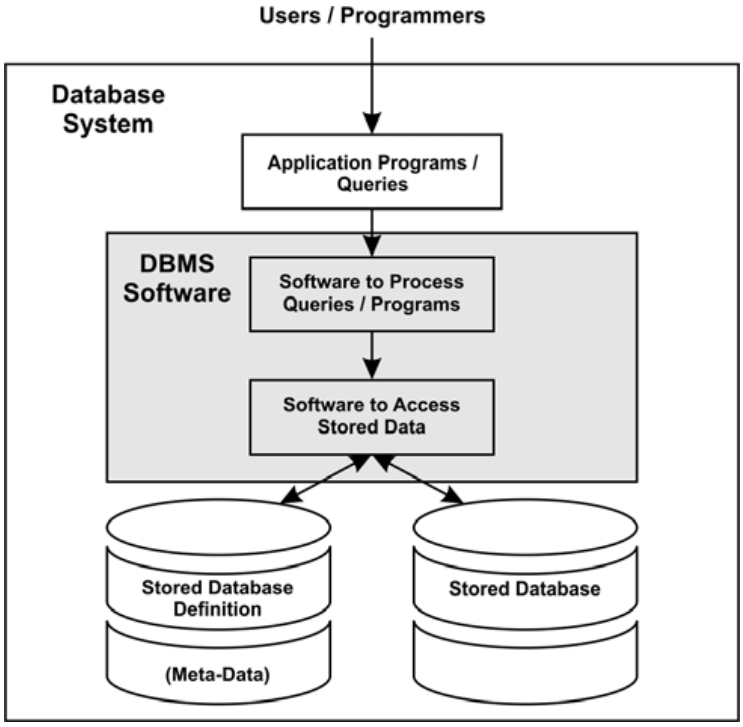
A collection of related data

Database Management System (DBMS):

A DBMS is software that provides an interface to interact with a database. It allows users to create, read, update, and delete data, as well as manage database structure and access controls. The DBMS serves as a mediator between the database and its users. Basically it's a tool to create Database like: (oracle , SQL Server .. etc)

Database system:

The DBMS software with the data itself, the term "database system" refers to the combination of the database itself, the DBMS, and the associated applications that interact with the database to perform various tasks.



Meta- data: For example in the table of student (Name, Id) of the student is the meta data (label).

Data is the value. For example, Mohammed, 112.

ERD contain the meta data there is no data, all the data will be on Physical schema.

A Database Management System (DBMS) offers several advantages:

1. Data Integrity and Consistency

- **Explanation:** DBMS ensures that data is accurate and consistent across the entire database. It uses constraints and rules to prevent invalid data from being entered.
- **Benefit:** Helps maintain the quality and reliability of data, which is crucial for decision-making and operational efficiency.

2. Data Security

- **Explanation:** DBMS provides robust security mechanisms to control user access to the data. It supports authentication and authorization features, allowing only authorized users to perform specific operations.
- **Benefit**: Protects sensitive information from unauthorized access, ensuring data privacy and compliance with regulatory requirements.

3. Data Redundancy Control

- **Explanation:** In a DBMS, data is stored in a centralized location, reducing duplication across different files. It allows for efficient data management by using relationships and normalization techniques.
- **Benefit:** Minimizes storage space requirements and eliminates data inconsistencies caused by duplicate records.

4. Efficient Data Retrieval and Querying

- **Explanation:** The DBMS uses indexing and advanced querying capabilities (e.g., SQL) to quickly retrieve and manipulate data.
- **Benefit:** Enhances the speed of data access and simplifies the process of generating reports, performing searches, and conducting analysis.

5. Backup and Recovery

- **Explanation:** DBMS supports automatic backup and recovery procedures. It can restore the database to a previous state in case of hardware failures, accidental deletion, or data corruption.
- **Benefit:** Ensures data availability and minimizes downtime, which is critical for business continuity.

6. Data Concurrency Control

- **Explanation:** The DBMS supports multiple users accessing the database simultaneously while managing transactions in a way that ensures consistency.
- **Benefit:** Allows multiple users to work with the database at the same time without conflicts or data integrity issues.

7. Data Independence

- **Explanation:** DBMS separates the data from the application programs that use it. Changes in the database structure (e.g., adding new fields or tables) do not require changes in the applications.
- **Benefit:** Simplifies maintenance and development, allowing for flexibility in modifying the database schema.

8. Scalability

- **Explanation:** DBMS can handle increasing amounts of data and users as an organization's needs grow. It allows for expansion without significant changes to the database structure.
- **Benefit:** Supports the growth of businesses and adapts to changing data requirements.

9. Standardization of Data Management

- **Explanation:** DBMS uses standardized methods for storing, retrieving, and managing data. SQL is a widely accepted standard for querying relational databases.
- **Benefit:** Ensures compatibility across different systems and reduces the learning curve for database administrators and developers.

10. Support for Complex Data Relationships

- **Explanation:** The DBMS can easily model complex relationships between different entities using tables and foreign keys, supporting hierarchical, network, and relational data models.
- **Benefit:** Facilitates the representation of real-world scenarios and supports complex data interactions, making the database more useful for various applications.

11. Reduced Development Time

- **Explanation:** The use of a DBMS allows developers to focus on application development without worrying about data management details. Features like indexing, security, and transactions are handled by the DBMS.
- **Benefit:** Speeds up the development process and reduces the time and cost associated with building and maintaining custom data management solutions.

12. Enhanced Data Sharing and Integration

- **Explanation:** A DBMS provides a central data repository accessible by multiple applications and users. It facilitates data integration across different systems and departments.
- **Benefit:** Promotes collaboration and data sharing within an organization, enabling better decision-making and streamlining business processes.

Database user:

System analyst

- Responsible for analysing and designing system requirements, including database requirements.
- Works on understanding business needs and translating them into functional specifications, which often include data models and database design requirements.
- May not directly manage or modify databases but collaborates closely with database designers and developers.

Database designer

- Focuses on designing the database schema, which includes defining tables, relationships, constraints, and data types.
- Ensures that the database design meets business requirements and is optimized for performance and data integrity.
- Lays out the blueprint for how data will be stored, accessed, and managed in the database.

Database developer

- Implements the database design by writing SQL code to create tables, views, indexes, stored procedures, and triggers.
- Works on optimizing database queries for performance and developing complex data processing tasks.
- May also collaborate with application developers to integrate database functionality into applications.

Database admin

- Manages and maintains the database, ensuring it is available, secure, and performing efficiently.
- Responsible for tasks like backup and recovery, user access control, monitoring performance, and troubleshooting issues.
- Plays a critical role in database security and disaster recovery planning.

Application developer

- Develops applications that interact with the database, such as web or desktop software.
- Uses programming languages (e.g., Java, C#, Python) to write code that connects to the database, retrieves data, and updates it based on user inputs.
- Collaborates with database developers to optimize the integration of database functionalities.

BI developer

- Specializes in using data from the database to generate reports, dashboards, and data visualizations.

- Works with tools like SQL Server Reporting Services (SSRS), Power BI, Tableau, or other BI platforms to create insights from data.
- Often designs data warehouses and ETL (Extract, Transform, Load) processes to consolidate data from various sources for analysis.

Database Life Cycle

consists of steps that guide the development, implementation, and maintenance of a database system, ensuring it meets the organization's requirements. The stages you mentioned can be viewed as part of this life cycle, with some specific activities involved in each. Here's an explanation of these stages:

Database Analysis

- **Objective:** Identify and understand the requirements of the database, including the data that needs to be stored, how it will be accessed, and what operations will be performed on it.
- **Activities:**

- Gather requirements by interviewing stakeholders, studying current processes, and understanding business needs.
- Identify entities (objects to be stored) and relationships between them.
- Analyse existing data and workflows to determine the data structure and access patterns.

Database Design

- **Objective:** Create a detailed design for the database structure, based on the analysis stage.
- **Activities:**

- Conceptual Design: Develop a high-level data model that captures the entities, attributes, and relationships.
- Logical Design: Refine the conceptual design into a more detailed data model, including tables, fields, primary and foreign keys, and normalization to eliminate redundancy.
- Physical Design: Translate the logical data model into a physical database structure, specifying how data will be stored, indexed, and accessed on the storage medium.

- **Output:** Database schema, Entity-Relationship Diagram (ERD), data dictionary, and other design documents.

Database Mapping

- **Objective:** Convert the logical and physical design into actual database structures using a Database Management System (DBMS).
- **Activities:**

- Map entities and relationships to tables, fields, and constraints in the DBMS.
- Define the mappings between high-level data models and the actual database schema.
- Create scripts or use tools to generate the required tables, views, and indexes.

➢ **Output:** Database schema defined in the DBMS, including tables, relationships, and constraints.

Database Implementation

➢ **Objective:** Build and populate the database according to the mapped design.
➢ **Activities:**

- Set up the DBMS, install software, and configure server settings.
- Execute scripts to create the database objects (tables, views, indexes, etc.).
- Load data into the database using ETL (Extract, Transform, Load) processes.
- Implement any triggers, stored procedures, or constraints required for data integrity.

➢ **Output:** A fully functioning database that is ready for use and testing.

GUI Application

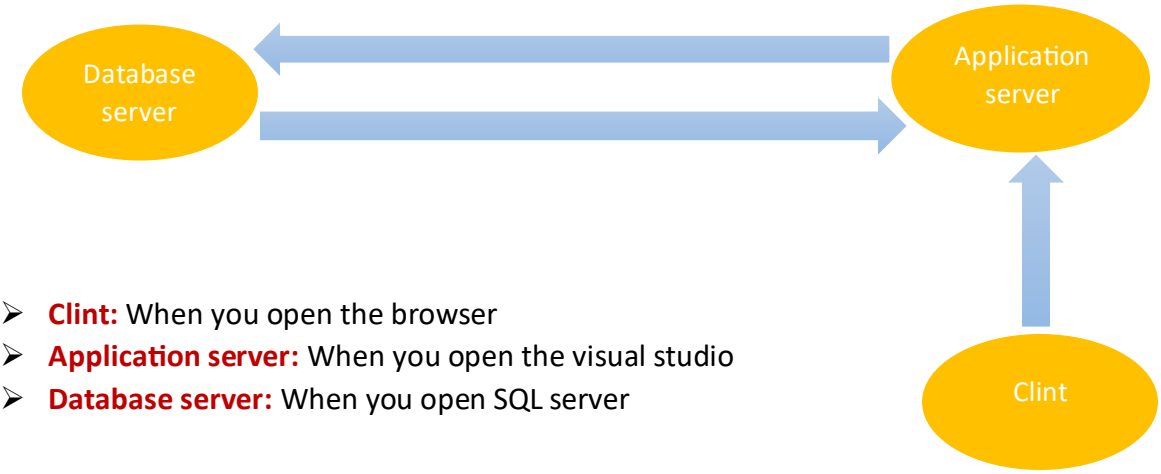
➢ **Objective:** Develop a graphical user interface (GUI) application that allows users to interact with the database.
➢ **Activities:**

- Design and develop forms, reports, and other user interfaces for data entry, updates, and retrieval.
- Implement business logic within the application to manage data operations (CRUD: Create, Read, Update, Delete).
- Test the application to ensure it integrates correctly with the database and provides a user-friendly experience.

➢ **Output:** An application that enables users to access and manipulate the database through a user-friendly interface.

Clint

➢ Open the browser and write the URL to start the application.



- **Clint:** When you open the browser
- **Application server:** When you open the visual studio
- **Database server:** When you open SQL server