DATABASE SYSTEMS - DCS II

Introduction to Database Systems

Overview of Databases

Definition of a Database: A database is an organized collection of data, typically stored and accessed electronically from a computer system. Databases are designed to efficiently store, retrieve, and manage large amounts of structured information.

Databases play a critical role in modern applications by allowing the storage, retrieval, and manipulation of data efficiently.

They support business operations, decision-making, and data analytics.

Examples of real-world database applications include online banking systems, inventory management systems, social media platforms, and customer relationship management (CRM) systems.

Historical Development:

- Early databases were flat-file systems, storing data in plain text files, which were difficult to manage and maintain.
- The introduction of hierarchical and network databases improved data organization but lacked flexibility.
- The development of relational databases in the 1970s, led by Edgar F. Codd revolutionized data management by introducing the concept of tables, which allowed for more flexible and efficient data handling.
- The emergence of NoSQL databases in the 2000s addressed the need for scalability and flexibility in handling unstructured and semi-structured data.

Types of Databases

Relational Databases:

Relational databases store data in tables with rows and columns, where each row represents a record and each column represents a field or attribute.

Key Features:

- Use of Structured Query Language (SQL) for data manipulation and retrieval.
- Support for complex queries, transactions, and data integrity constraints.
- Examples: MySQL, PostgreSQL, Oracle Database, Microsoft SQL Server.

NoSQL Databases:

NoSQL (Not Only SQL) databases are designed to handle large volumes of unstructured or semi-structured data, providing greater flexibility and scalability compared to traditional relational databases.

Key Features:

- Schema-less design, allowing for dynamic and flexible data models.
- Horizontal scaling to handle large datasets across distributed systems.

- Support for eventual consistency, which trades off immediate consistency for higher availability and partition tolerance.
- Examples: MongoDB, Redis, Apache Cassandra,etc.

Hierarchical Databases:
Organize data in a tree-like structure, with a single root and multiple levels of related records. Example: IBM's Information Management System (IMS).
Network Databases:
Use a graph structure, allowing more complex many-to-many relationships. Example: Integrated Data Store (IDS).
Object-Oriented Databases:
Store data as objects, similar to object-oriented programming. Example: ObjectDB.Key Concepts: Tables, Records, FieldsTables:Tables are the fundamental building blocks

Key Concepts:

- **Table**: A table is the basic unit of data storage in a relational database. It consists of rows (records) and columns (fields). Each row represents a set of data related to a specific item, and all rows follow the same structure but contain different data. The actual data is stored in the rows, with one or no items per row.
- **Primary Key**: A primary key is a unique identifier for each record in a table. It consists of one or more columns that must be unique for every row in the table and cannot contain null values. This ensures that every record can be distinctly identified.
- **Foreign Key**: A foreign key is a column or set of columns in one table that references the primary key in another table. While the primary key uniquely identifies a record in its table, the foreign key helps to establish relationships between tables by referring to these unique records.
- **Column**: A column represents a specific attribute or set of values in a table. Each column defines an aspect of the data stored in the table and contains a specific data type for all its entries, representing a particular characteristic of the records.
- **Field**: A field is a specific piece of data or information within a table that contains a homogeneous set of values. Fields are the individual cells in a table, and each one corresponds to a specific attribute of a record.
- **Record**: A record, also known as a row, is an individual entry in a table that consists of a group of related fields. Each record contains data pertaining to a particular item, and all records in a table share the same structure but hold different values.
- **Recordset**: A recordset is the collection of records and fields that results from executing a query. It represents a subset of data from a table or a combination of tables, retrieved based on specific conditions.
- **DBMS (Database Management System)**: A DBMS is software designed to help manage large collections of data. It provides tools and functions to store, retrieve, and manipulate data efficiently, while also ensuring data integrity and security.
- **Data Model**: A data model is a set of high-level constructs that describe how data is organized and stored. It abstracts the complexities of physical storage, enabling users to define and manipulate data in a consistent and logical manner through the DBMS.

- **SQL (Structured Query Language)**: SQL is the standard language for interacting with relational databases. It provides commands for querying, inserting, updating, and managing data. SQL is supported by all major relational databases, making it an essential tool for database administrators.
- **Constraints**: Constraints are rules applied to columns to enforce data integrity. They restrict the type of data that can be entered into a column. For example, a "not-null" constraint ensures that a column must always contain a value, preventing null entries.

Database Management Systems (DBMS)

A DBMS is software that provides an interface between users and databases, enabling the creation, management, and querying of databases.
DBMSs are responsible for ensuring that data is stored, retrieved, and manipulated efficiently and securely.

Core Functionalities of DBMS:

- Data Storage, Retrieval, and Update: DBMSs provide efficient mechanisms for storing large amounts of data, retrieving it as needed, and updating records while maintaining data integrity.
- Data Security and Integrity: DBMSs enforce security measures such as authentication, authorization, and encryption to protect sensitive data from unauthorized access. They also ensure data consistency through constraints and rules.
- Data Backup and Recovery: DBMSs support data backup and recovery operations to protect data against loss due to hardware failures, software issues, or other disasters. Automated backups and point-in-time recovery are common features.
- Multi-user Access and Concurrency Control: DBMSs allow multiple users to access and manipulate data simultaneously without causing conflicts or inconsistencies. Concurrency control mechanisms ensure that transactions are processed in a manner that preserves data integrity.
- Data Modelling and Design: DBMSs provide tools for designing and modelling databases, including the creation of schemas, tables, indexes, and relationships.
- Query Processing and Optimization: DBMSs use query processors to interpret and execute SQL queries efficiently. They also optimize query performance by choosing the best execution plan based on available indexes, statistics, and algorithms.
- Data Independence: DBMSs provide a level of abstraction between the physical storage of data and the logical representation, allowing changes to the physical structure without affecting application programs.

Advantages & Disadvantages of DBMS

Advantages:

- Efficient Data Management and Retrieval:
    - DBMSs provide a systematic approach to managing data, reducing redundancy and inconsistency, and ensuring that data is organized and easily accessible.
    - Powerful query languages like SQL allow users to perform complex queries and analytics on the data.

- Enhanced Data Security and Integrity:
    - DBMSs enforce access controls, encryption, and data integrity constraints, ensuring that only authorized users can access and modify the data.
    - Data integrity rules prevent invalid data from being entered, maintaining the accuracy and reliability of the database.
- Support for Concurrent User Access:
    - DBMSs enable multiple users to access and manipulate data simultaneously without compromising data integrity, thanks to features like transaction management and concurrency control.
- Reduced Data Redundancy and Inconsistency:
    - DBMSs reduce data redundancy by allowing data to be stored in a single location and shared across multiple applications, minimizing the risk of inconsistent data.
- Facilitated Data Sharing Across Applications:
    - DBMSs allow data to be shared across different applications and platforms, promoting data integration and collaboration across an organization.

Disadvantages:

- Complexity and Cost of Implementation:
    - Implementing a DBMS can be complex and costly, requiring significant investment in hardware, software, and skilled personnel.
    - The initial setup, configuration, and ongoing maintenance of a DBMS can be resource-intensive.
- Requires Specialized Knowledge and Skills:
    - Managing a DBMS requires specialized knowledge in database design, SQL, and system administration, which may necessitate training or hiring of experts.
- Potential for Performance Bottlenecks in Large Systems:
    - In large-scale databases, performance issues can arise due to the volume of data, number of concurrent users, or complexity of queries.
    - DBMS performance tuning and optimization are critical to avoid bottlenecks that could impact system responsiveness.
- Risk of Data Loss if Not Properly Maintained:
    - If backup and recovery procedures are not properly implemented, there is a risk of data loss in the event of system failures or disasters.
    - Poorly managed DBMSs can lead to data corruption, security vulnerabilities, or breaches.