25-05-2023

**INTRODUCTION TO DBMS, RDBMS AND NOSQLDB**

The database is a collection of inter-related data which is used to retrieve, insert and delete the data efficiently. It is also used to organize the data in the form of a table, schema, views, and reports, etc. Using the database, you can easily retrieve, insert, and delete the information.

* Database management system is a software which is used to manage the database. For example: MySQL, Oracle, etc are a very popular commercial database which is used in different applications.
* DBMS provides an interface to perform various operations like database creation, storing data in it, updating data, creating a table in the database and a lot more.
* It provides protection and security to the database. In the case of multiple users, it also maintains data consistency.

**DBMS allows users to perform the following tasks:**

1. **Data Definition:** It is used for creation, modification, and removal of definition that defines the organization of data in the database.
2. **Data Updating:** It is used for the insertion, modification, and deletion of the actual data in the database.
3. **Data Retrieval:** It is used to retrieve the data from the database which can be used by applications for various purposes.
4. **User Administration:** It is used for registering and monitoring users, maintain data integrity, enforcing data security, dealing with concurrency control, monitoring performance and recovering information corrupted by unexpected failure.

**Characteristics of a Database Management System:**

* A database management system is able to store any kind of data in a database.
* The database management system has to support ACID (atomicity, consistency, isolation, durability) properties.
* The Database management system allows so many users to access databases at the same time.
* Backup and recovery are the two main methods which allow users to protect the data from damage or loss.
* It also provides multiple views for different users in a single organization.
* It follows the concept of normalization which is helpful to minimize the redundancy of a relation.
* It also provides users query language, helpful to insert, retrieve, update, and delete the data in a database.

**Advantages of using a DBMS:**

1. **Data organization**: A DBMS allows for the organization and storage of data in a structured manner, making it easy to retrieve and query the data as needed.
2. **Data integrity**: A DBMS provides mechanisms for enforcing data integrity constraints, such as constraints on the values of data and access controls that restrict who can access the data.
3. **Concurrent access**: A DBMS provides mechanisms for controlling concurrent access to the database, to ensure that multiple users can access the data without conflicting with each other.
4. **Data security**: A DBMS provides tools for managing the security of the data, such as controlling access to the data and encrypting sensitive data.
5. **Backup and recovery**: A DBMS provide mechanisms for backing up and recovering the data in the event of a system failure.
6. **Data sharing**: A DBMS allows multiple users to access and share the same data, which can be useful in a collaborative work environment.
7. Disadvantages of using a DBMS:
8. **Complexity**: DBMS can be complex to set up and maintain, requiring specialized knowledge and skills.
9. **Performance overhead**: The use of a DBMS can add overhead to the performance of an application, especially in cases where high levels of concurrency are required.
10. **Scalability:** The use of a DBMS can limit the scalability of an application, since it requires the use of locking and other synchronization mechanisms to ensure data consistency.
11. **Cost**: The cost of purchasing, maintaining and upgrading a DBMS can be high, especially for large or complex systems.
12. **Limited use cases**: Not all use cases are suitable for a DBMS, some solutions don’t need high reliability, consistency or security and may be better served by other types of data storage.

**Disadvantages of using a DBMS:**

1. **Cost of Hardware and Software**: It requires a high speed of data processor and large memory size to run DBMS software.
2. **Size**: It occupies a large space of disks and large memory to run them efficiently.
3. **Complexity**: Database system creates additional complexity and requirements.
4. **Higher impact of failure**: Failure is highly impacted the database because in most of the organization, all the data stored in a single database and if the database is damaged due to electric failure or database corruption then the data may be lost forever.

**Key Features of Database Management Systems:**

1. **Data Definition**: A DBMS allows users to define the structure of the database, including the tables, fields, and relationships between tables.
2. **Data Manipulation**: A DBMS allows users to insert, update, and delete data in the database, as well as retrieve data using queries.
3. **Data Security**: A DBMS provides security features to prevent unauthorized access to the database and to protect the data from theft, loss, or corruption.
4. **Data Integrity**: A DBMS provides mechanisms to maintain the accuracy and consistency of the data in the database, including enforcing constraints, such as unique keys, foreign keys, and check constraints.
5. **Data Backup and Recovery**: A DBMS provides mechanisms to back up the data in the database and to recover data in case of data loss or corruption.

**Several types of DBMS:**

1. **Relational DBMS (RDBMS**): An RDBMS stores data in tables with rows and columns, and uses SQL (Structured Query Language) to manipulate the data.
2. **Object-Oriented DBMS (OODBMS)**: An OODBMS stores data as objects, which can be manipulated using object-oriented programming languages.
3. **NoSQL DBMS**: A NoSQL DBMS stores data in non-relational data structures, such as key-value pairs, document-based models, or graph models.

**Relational Database Management (RDBMS):**

The software used to store, manage, query, and retrieve data stored in a relational database is called a relational database management system (RDBMS). The RDBMS provides an interface between users and applications and the database, as well as administrative functions for managing data storage, access, and performance.

All modern database management systems like SQL, MS SQL Server, IBM DB2, ORACLE, My-SQL, and Microsoft Access are based on RDBMS. It is called Relational Database Management System (RDBMS) because it is based on the relational model introduced by E.F. Codd.

Data is represented in terms of tuples (rows) in RDBMS. A relational database is the most commonly used database. It contains several tables, and each table has its primary key. Due to a collection of an organized set of tables, data can be accessed easily in RDBMS.

**Key components and concepts of RDBMS:**

* Tables: The fundamental building blocks of an RDBMS are tables, which represent entities or concepts. A table consists of rows (also called records or tuples) and columns (also called fields or attributes). Each column represents a specific data attribute, while each row represents a unique instance or record.
* Primary Key: A primary key is a unique identifier for each record in a table. It ensures that each row is uniquely identifiable and helps establish relationships between tables. Usually, a primary key is formed by one or more columns with unique values.
* Relationships: RDBMS allows you to establish relationships between tables using common attributes. The most common relationship types are:
* One-to-One (1:1): Each record in one table is related to only one record in another table.
* One-to-Many (1:N): Each record in one table can be related to multiple records in another table.
* Many-to-Many (N:M): Multiple records in one table can be related to multiple records in another table. Achieved using a join table.
* Normalization: RDBMS follows normalization techniques to eliminate data redundancy and ensure data integrity. Normalization involves breaking down tables into smaller, more manageable pieces, reducing data duplication, and establishing relationships between them.
* SQL (Structured Query Language): RDBMS systems use SQL as the standard language for managing and querying databases. SQL provides a set of commands, such as SELECT, INSERT, UPDATE, and DELETE, to manipulate data within tables, define relationships, and retrieve information based on specific criteria.
* Data Integrity: RDBMS systems enforce data integrity rules to maintain the accuracy and consistency of data. These rules include:
* Entity Integrity: Each row in a table must have a unique primary key.
* Referential Integrity: Relationships between tables must be maintained, and foreign keys must refer to valid primary key values.
* Domain Integrity: Data in each column must adhere to predefined data types and constraints.
* ACID Properties: RDBMS systems ensure data consistency and reliability by following the ACID properties:
* Atomicity: Transactions are treated as atomic units of work that are either fully completed or fully rolled back if an error occurs.
* Consistency: The database remains in a consistent state before and after each transaction.
* Isolation: Transactions are executed in isolation from each other to prevent interference or conflicts.
* Durability: Once a transaction is committed, its changes are permanently stored and protected against data loss.
* Security: RDBMS systems provide robust security features to protect data from unauthorized access. These include user authentication, access control, encryption of sensitive data, and audit trails to track changes made to the database.
* Scalability and Performance: RDBMS systems are designed to handle large volumes of data and provide mechanisms to optimize query performance. Techniques such as indexing, query optimization, and caching are employed to enhance the efficiency of data retrieval operations.
* Concurrency Control: RDBMS systems manage concurrent access to the database by multiple users or applications. They use concurrency control mechanisms such as locking, transaction isolation levels, and multi-version concurrency control (MVCC) to ensure data consistency and prevent conflicts.

**Advantages of RDBMS:**

* **Data Integrity:** RDBMS systems enforce data integrity rules, ensuring that the data stored in the database is accurate, consistent, and reliable. The use of primary keys, foreign keys, and constraints helps prevent data duplication, maintain relationships between tables, and enforce data validation rules.
* **Structured and Organized Data:** RDBMS organizes data into tables with predefined columns and data types, providing a structured and systematic approach to data storage. This structure makes it easier to understand and manage the data, and allows for efficient querying and retrieval of information.
* **Data Consistency:** RDBMS systems follow ACID (Atomicity, Consistency, Isolation, Durability) properties, ensuring that transactions are processed reliably and consistently. ACID compliance ensures that the database remains in a consistent state, even in the presence of concurrent transactions or system failures.
* **Querying and Data Retrieval**: RDBMS systems use SQL (Structured Query Language) as a standard language for querying and retrieving data. SQL provides powerful and flexible commands for complex data manipulations, aggregations, filtering, and sorting operations. This makes it easier to extract meaningful insights and generate reports from the database
* **Scalability:** RDBMS systems are designed to handle large amounts of data and can scale vertically (by adding more resources to a single server) or horizontally (by distributing data across multiple servers). This scalability allows organizations to accommodate growing data volumes and support increasing user demands without sacrificing performance.
* **Data Security:** RDBMS systems offer robust security features to protect sensitive data. These features include user authentication, access control mechanisms, encryption of data at rest and in transit, and auditing capabilities to track and monitor data access and modifications. RDBMS systems provide a secure environment for storing and managing critical data.
* **Data Independence:** RDBMS provides a layer of abstraction between the physical storage of data and the applications that use the data. This data independence allows application developers to focus on the logical representation of data, rather than worrying about low-level storage details. Changes to the underlying database schema or storage technology can be made without impacting the applications that rely on the data.
* **Data Sharing and Integration:** RDBMS systems support data sharing and integration across multiple applications. Different applications can access and manipulate the same database concurrently, enabling data consistency and centralized management. This facilitates collaboration, data sharing, and avoids data silos within an organization.
* **Data Backup and Recovery:** RDBMS systems provide mechanisms for data backup and recovery. Regular backups can be taken to protect against data loss due to hardware failures, system crashes, or human errors. In the event of data loss, RDBMS systems offer recovery options to restore the database to a previous state.
* **Mature Ecosystem and Support:** RDBMS systems have been in existence for several decades, leading to a mature ecosystem with a wide range of tools, documentation, and community support. There is a wealth of expertise available for RDBMS systems, making it easier to find skilled professionals and resources for database development, administration, and troubleshooting.

**NOSQL Database:**

* NoSQL (Not Only SQL) databases are a category of database management systems that differ from traditional RDBMS (Relational Database Management Systems).
* NoSQL databases provide a flexible, scalable, and non-relational approach to storing and managing data.
* Instead of the typical tabular structure of a relational database, NoSQL databases, house data within one data structure, such as JSON document.
* Since this non-relational database design does not require a schema, it offers rapid scalability to manage large and typically unstructured data sets. NoSQL is also type of distributed database, which means that information is copied and stored on various servers, which can be remote or local.
* This ensures availability and reliability of data. If some of the data goes offline, the rest of the database can continue to run.
* NoSQL databases are commonly used in modern web applications, real-time analytics, content management systems, IoT (Internet of Things) applications, and scenarios requiring horizontal scalability and agility.
* It's important to note that while NoSQL databases offer benefits in certain use cases, they may not be suitable for all scenarios, especially when complex relational data structures and strict consistency requirements are essential.

**Characteristics and features of NoSQL databases:**

* Schema-less Structure: Unlike RDBMS, NoSQL databases do not enforce a predefined schema for data. They allow for dynamic and flexible data models, where each record (document) can have a different structure. This schema-less nature enables agility and accommodates evolving data requirements.
* High Scalability and Performance: NoSQL databases are designed to handle massive amounts of data and high traffic loads across distributed systems. They employ distributed architectures and horizontal scaling techniques, such as sharding and replication, to achieve high scalability and performance. This allows them to handle large-scale applications with ease.
* Variety of Data Models: NoSQL databases offer various data models to handle different types of data. The four main types of NoSQL databases are:
* Document Databases: Store data in flexible, self-contained documents (e.g., JSON or XML) that can vary in structure within a collection.
* Key-Value Stores: Store data as simple key-value pairs, providing fast retrieval by key but limited querying capabilities.
* Columnar Databases: Organize data in columns rather than rows, optimizing for analytical workloads and efficient storage and retrieval of specific columns.
* Graph Databases: Focus on representing and storing relationships between data entities, making them suitable for scenarios with complex interconnected data.
* Distributed Architecture: NoSQL databases are designed to operate on distributed systems, where data is partitioned and distributed across multiple servers or nodes. This architecture allows for horizontal scaling, fault tolerance, and high availability.
* Eventually Consistent: NoSQL databases often adopt an "eventual consistency" model rather than strict immediate consistency. This means that changes made to the data are propagated asynchronously and may take some time to synchronize across distributed nodes. This approach prioritizes availability and performance over immediate consistency.
* Agile Development and Iterative Prototyping: The flexible nature of NoSQL databases enables developers to iterate quickly during the development process. Schema-less structures allow for easy modifications to data models without the need for complex database migrations.
* Handling Big Data and Unstructured Data: NoSQL databases excel in handling unstructured and semi-structured data, such as social media feeds, log files, sensor data, and other forms of big data. They provide efficient storage and retrieval mechanisms for these data types, often leveraging distributed file systems and parallel processing.

**Types of NOSQL Databases:**

* Graph Databases: Examples – Amazon Neptune, Neo4j
* Key value store: Examples – Memcached, Redis, Coherence
* Tabular: Examples – HBase, Big Table, Accumulo
* Document-based: Examples – MongoDB, CouchDB, Cloudant

**When Should NoSQL be used?**

1. When a huge amount of data needs to be stored and retrieved.
2. The relationship between the data you store is not that important
3. The data changes over time and is not structured.
4. Support of Constraints and Joins is not required at the database level
5. The data is growing continuously and you need to scale the database regularly to handle the data.

**Drawbacks of NoSQL Databases:**

* Lack of Standardization
* Lack of ACID compliance
* Lack of support for complex queries
* Limited Querying Capabilities
* Data duplication and Data redundancy
* Learning curve and developer familiarity
* Limited tools and third-party integrations
* Data consistency challenges
* Limited support for complex transactions
* Increased complexity for data modelling