WHAT IS DBMS (DATABASE MANAGEMENT SYSTEM)?

Data is the cornerstone of any modern software application, and databases are the most common way to store and manage data used by applications.

With the explosion of web and cloud technologies, databases have evolved from traditional relational databases to more advanced types of databases such as NoSQL, columnar, key-value, hierarchical, and distributed databases. Each type has the ability to handle structured, semi-structured, and even unstructured data.

On top of that, databases are continuously handling mission-critical and sensitive data. When this is coupled with compliance requirements and the distributed nature of most data sets, managing databases has become highly complex. As a result, organizations require robust, secure, and user-friendly tools to maintain these databases.

This is where database management systems come into play—by offering a platform to manage databases. Let's take a look.

Introduction of DBMS

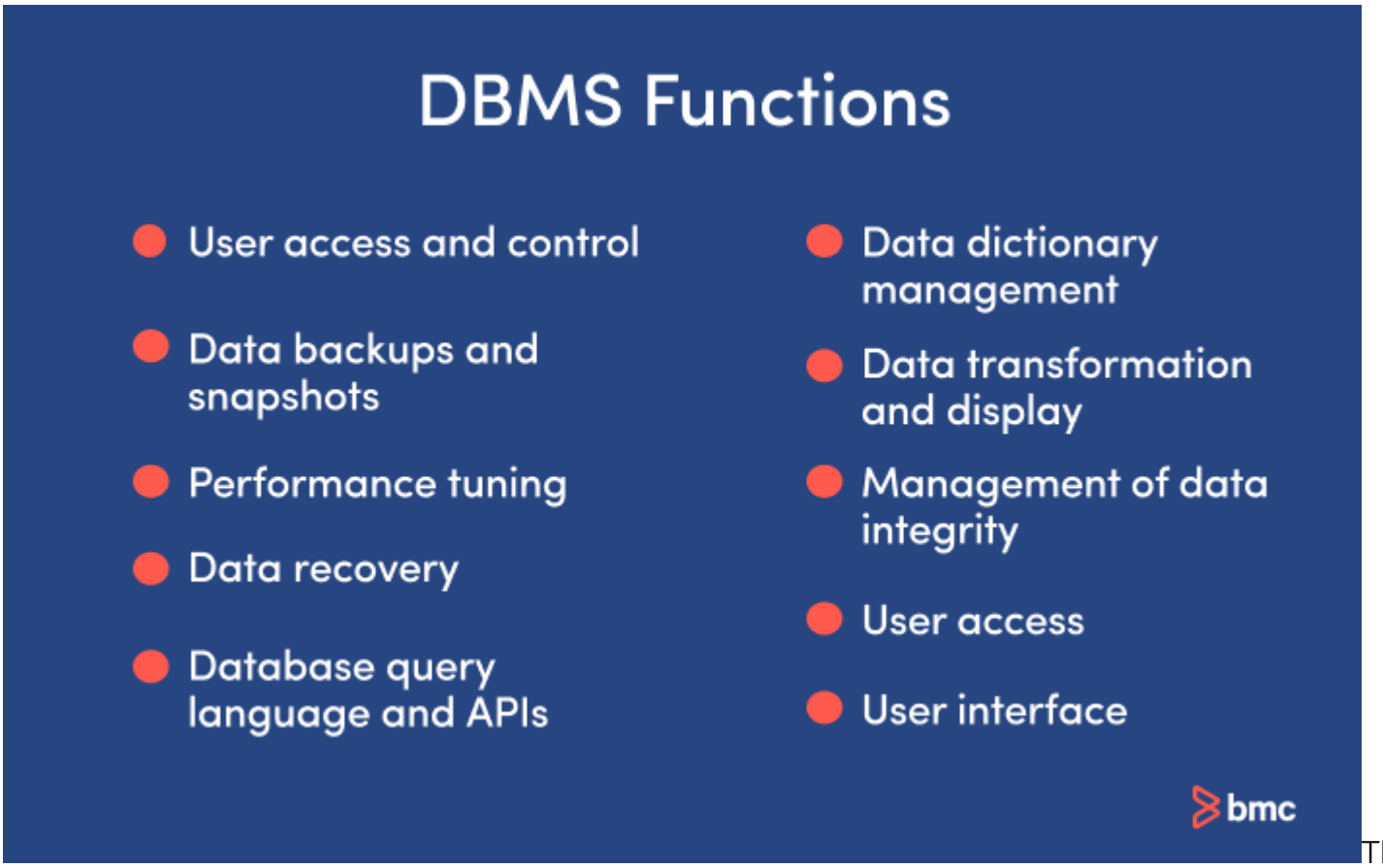
What is a database management system (DBMS)?

A database management system (DBMS) is a software tool for creating, managing, and reading a database. With DBMS, users can access and interact with the underlying data in the database. These

actions can range from simply querying data to defining database schemas that fundamentally affect the structure of DBMS.

Furthermore, DBMS allows users to interact with a database securely and concurrently without interfering with each user and while maintaining data integrity.

What are the functions of DBMS?



The typical DBMS tasks or functions include:

- **User access and control.** Administrators can easily configure user accounts, define access policies, modify restrictions and access scopes to limit access to underlying data, control user actions, and manage database users.
- **Data backups and snapshots.** DBMS can simplify the database backup process through a simpler and straightforward interface for managing backups and snapshots. For safekeeping, users can move these backups to third-party locations, such as cloud storage.
- **Performance tuning.** DBMS can monitor database performance using integrated tools. Users can tune databases by creating optimized indexes to reduce I/O usage and optimize SQL queries for the best database performance.
- **Data recovery.** DBMS provides a recovery platform and the necessary tools to fully or partially restore databases to their previous state—effortlessly.

- **Database query language and APIs.** Access and use data via a variety of query languages and API connections.
- **Data dictionary management**. Dictionaries include metadata about the structure of the data and relationships between data points so that functionality can rely on structural abstractions rather than complex coding.
- **Data transformation and display**. DBMS transforms data on command, such as assembling attributes for the month, day and year as December 14, 2024, or 12/14/24 or another specified display format.
- **Management of data integrity**. DBMS establishes and maintains data consistency and minimizes duplications.
- **User access**. This policy permits more than one user to access the database at a time and follows ACID to accommodate multiple users.
- **User interface**. Whether accessing data through a web form, a direct dashboard, or a third-party distributed network, a browser-based interface makes it easy.

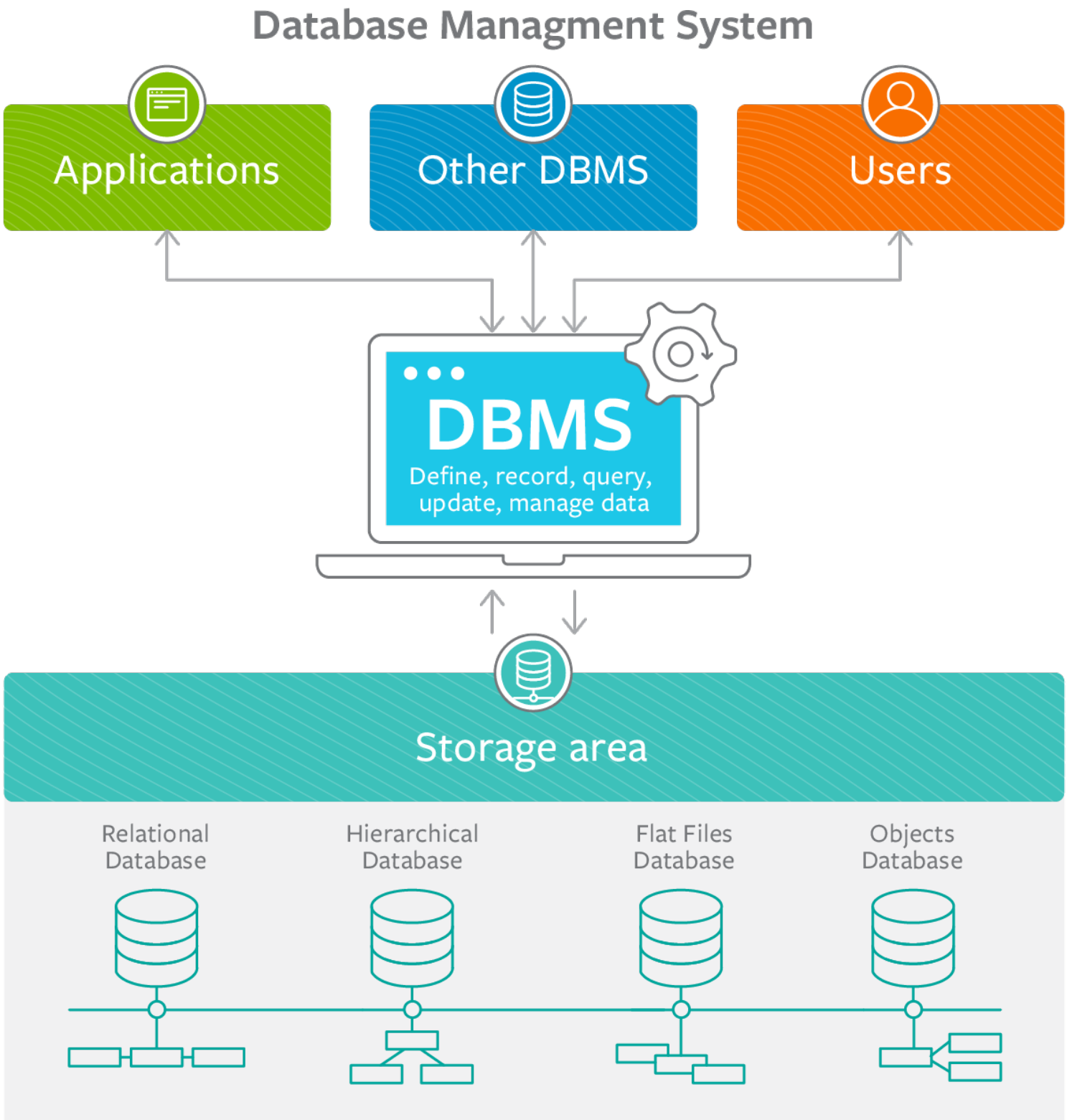
All these administrative tasks are facilitated using a single management interface. Most modern DBMS support handling multiple database workloads from a centralized DBMS software, even in a distributed database scenario. Furthermore, they allow organizations to have a governable top-down view of all the data, users, groups, locations, etc., in an organized manner.

(Explore the role of DBAs, or database administrators.)

How does DBMS work?

The various DBMS components work together to create an integrated system for structuring and storing data, supporting user queries and access, ensuring consistency and integrity, control, security, backups, and logging.

The following DBMS schematic illustrates how a DBMS system works:



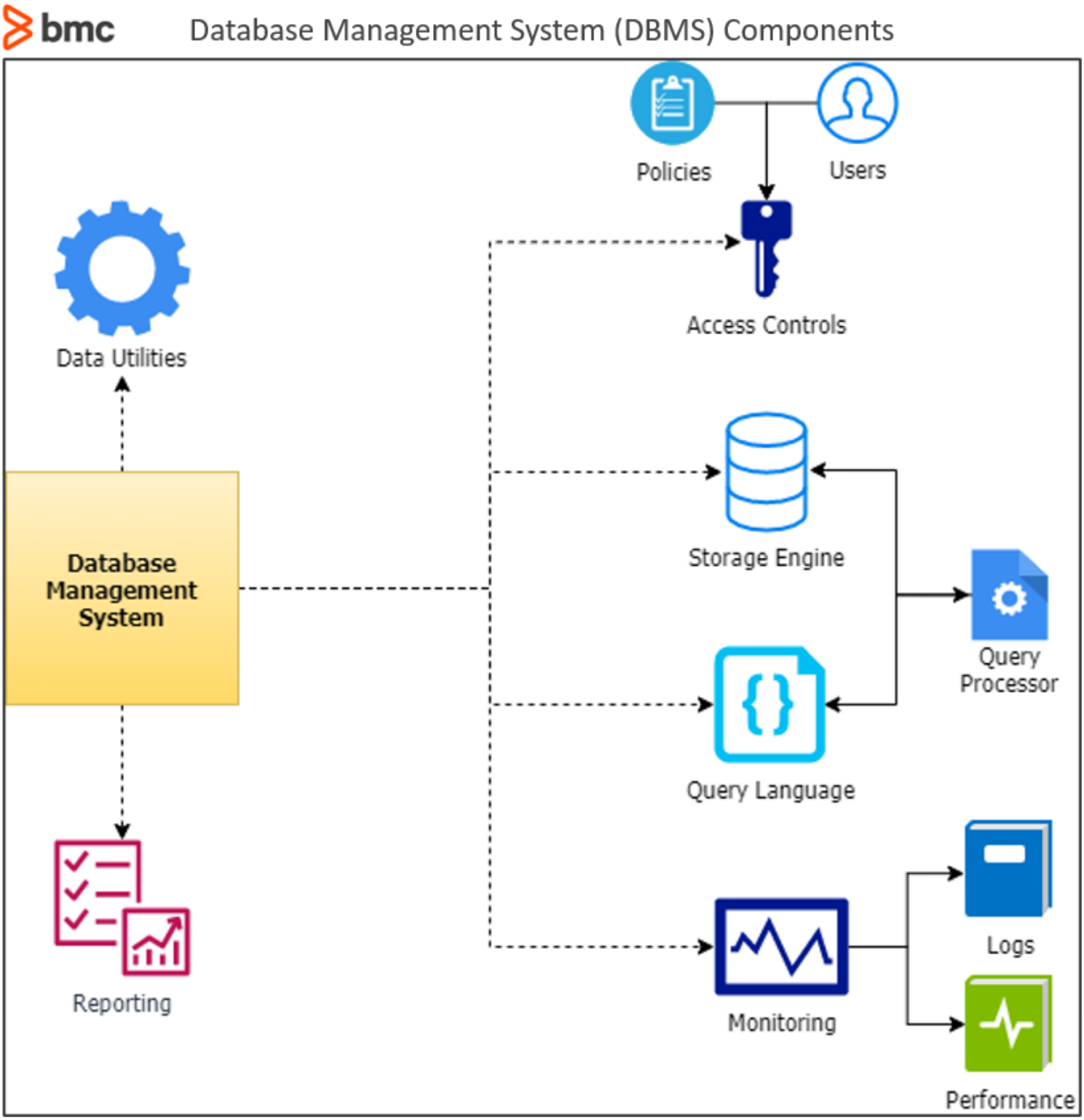
bmc

What are the components of a DBMS?

All DBMS comes with various integrated components and tools necessary to carry out almost all database management tasks. Some DBMS software even provides the ability to extend beyond the core functionality by integrating with third-party tools and services, directly or via plugins.

In this section, we will look at the common components of a DBMS that are universal across all database software:

1. Storage engine
2. Database query language
3. Query processor
4. Optimization engine
5. Metadata catalog
6. Log manager
7. Reporting and monitoring tools
8. Data utilities



1. Storage engine in a database

The database storage engine is the core component of the DBMS that interacts with the file system at an OS level to store data. All SQL queries which interact with the underlying data go through the storage engine.

Which storage engine is the best for a database?

The right storage engine depends on your data model. SQL engines supporting transactions work well with relational databases. Non-relational models, especially those that require scalability, work best with MongoDB or Cassandra.

2. Database query language

What is a database access language? A database access language is required for interacting with a database, from creating databases to simply inserting or retrieving data. A proper DBMS must support one or multiple query languages and language dialects. Structured query language (SQL) and MongoDB Query Language (MQL) are two query languages that are used to interact with the databases.

What are the 4 types of DBMS languages?

In many query languages, the query language functionality can be further categorized according to specific tasks:

- **Data Definition Language (DDL).** This consists of commands that can be used to define database schemas or modify the structure of database objects.
- **Data Manipulation Language (DML).** Commands that directly deal with the data in the database. All CRUD operations come under DML.
- **Data Control Language (DCL).** This deals with the permissions and other access controls of the database.
- **Transaction Control Language (TCL).** Command which deals with internal database transactions.

3. Query processor

The query processor is the intermediary between user queries and the database. In DBMS, query processing is the process of interpreting user queries, such as SQL, and making them actionable commands that the database can understand to perform the appropriate functionality.

What are the components of the query processor?

The query processor components each work together to extract data.

- **Parser**. This component translates a user query into a database language such as SQL, parses it for correct syntax, and verifies its logical meaning.
- **Optimizer**. This component converts the query into logical relational operations, identifies how much time and energy it will take to execute the query, and then specifies the exact operations and sequence for the most efficient execution.

- **Execution engine.** This is the component that carries out the query, implements algorithms and operators according to the optimized plan, and finally retrieves and formats the results.
- **Query cache.** Some systems include a component that stores frequently executed queries and results to save time and improve performance.

4. Optimization engine in DBMS

The optimization engine allows the DBMS to provide insights into the performance of the database in terms of optimizing the database itself and queries. When coupled with database monitoring tools, it can provide a powerful toolset to gain the best performance out of the database.

5. Metadata catalog

A metadata catalog, also referred to as a data catalog, is the centralized catalog of all the objects within the database. When an object is created, the DBMS keeps a record of that object with some metadata about it using the metadata catalog. Then, this record can be used to:

- Verify user requests to the appropriate database objects
- Provide an overview of the complete database structure

6. Log manager

The log manager is a component that will keep all the logs of the DBMS. These logs will consist of user logins and activity, database functions, backups and restore functions, etc. The log manager ensures all these logs are properly recorded and easily accessible.

(Compare logs to monitoring.)

7. Reporting & monitoring tools

Reporting and monitoring tools are another standard component that comes with a DBMS. DBMS reporting tools will enable users to generate reports while monitoring tools enable monitoring the databases for resource consumption, user activity, etc.

8. Data utilities

In addition to all the above, most DBMS software comes with additional inbuilt utilities to provide functionality such as:

- Data integrity checks
- Backup and restore
- Simple database repair
- Data validations
- Etc.

What are the different types of DBMS?

The evolution of data models, how data is structured, and the use cases of each has led to various types of DBMS. The most commonly used are:

1. Relational database management systems (RDBMS)

Relational Database Management Systems are the most common type of DBMS. Relational databases interact with databases that contain structured data in a table format with predefined relationships. Moreover, they use structured query language (SQL) to interact with databases. Some popular examples of RDBMS include:

- Microsoft SQL
- MySQL
- Oracle Database
- MariaDB
- PostgreSQL

2. NoSQL databases

NoSQL (nonrelational) databases are designed for semi-structured and unstructured data. They offer greater data modeling flexibility and often don't use a schema. They also support scaling across distributed systems.

Examples of nonrelational or NoSQL databases include:

- MongoDB
- Azure Cosmos DB
- Apache Cassandra
- CouchDB
- Amazon DynamoDB

3. Object-oriented DBMS (OODBMS)

This type of database stores data and data relationships as objects that can be used by object-oriented programming languages like C++ and Java in applications such as CAD systems, databases containing scientific research, and multimedia.

Examples of object-oriented databases include:

- ObjectDB
- Versant
- GemStone/S
- Objectivity/DB

4. Hierarchical DBMS

This type of database uses tree-like structures to organize data in parent-child relationships. A parent node can have many children and grandchildren, but each child node has only one parent. These DBMSs work well when data has well-defined relationships that can be organized into files and directories.

Examples of hierarchical databases include:

- IBM Information Management System (IMS)
- RDM Mobile
- Windows Registry
- XML data storage

5. Network DBMS

This type of database supports complex interconnections in many-to-many data relationships, with records that have multiple and complex links.

Examples of databases that use the network model include:

- IDMS (Integrated Database Management System)
- Oracle CODASYL

6. Columnar database management systems (CDBMS)

As the name suggests, CDMBS is used to manage columnar databases that store data in columns instead of rows, emphasizing high performance. Some databases that use columnar format are Apache Cassandra, Apache HBase, etc.

What are the advantages of DBMS?

DBMS was introduced to solve the fundamental issues associated with storing, managing, accessing, securing, and auditing data in traditional file systems. Software users and organizations can gain the following advantages of DBMS:

1. Increased data security

DBMS provides the ability to control users and enforce policies for security and compliance management. This controlled user access increases the database security and makes the data less vulnerable to security breaches.

2. Simple data sharing

DBMS enables users to access the database securely regardless of their location. Thus, they can handle any database-related task promptly without the need for complex access methods or worrying about database security. On top of that, DBMS allows multiple users to collaborate effectively when interacting with the database.

3. Data integration

DBMS allows users to gain a centralized view of databases spread across multiple locations and manage them using a single interface rather than operating them as separate entities.

4. Abstraction & independence

DBMS enables users to change the physical schema of a database without changing the logical schema that governs database relationships. As a result, organizations can scale the underlying database infrastructure without affecting the database operations.

Furthermore, any change to the logical schema can also be carried out without affecting applications that access the databases.

5. Streamlined backup & recovery mechanism

Most databases have built-in backup and recovery tools. Yet, DBMS offers centralized tools to facilitate backup and recovery functionality more conveniently and thereby provide a better user experience. Securing data has become easier than ever with functionality like:

- Automated snapshots
- Backup scheduling
- Backup verifications
- Multiple recovery methods

6. Uniform management & monitoring

DBMS provides a single interface to carry out all the management and monitoring tasks, thus simplifying the workload of database administrators. These tasks can range from database creation and schema modifications to reporting and auditing.

Why is DBMS important?

Considering the many advantages, DBMS is essential for any organization when managing databases. With different DBMS providing different feature sets, it is paramount that organizations rigorously evaluate the DBMS software before committing to a single system. However, a properly configured DBMS will greatly simplify the management and maintenance of databases at any scale. The scale, complexity, and feature set of a DBMS will depend on the specific DBMS and the organization's requirements.

Related reading

- BMC Big Data & Machine Learning Blog
- BMC IT Operations Blog
- Introduction To Database DevOps
- What Is DBaaS? Database-as-a-Service Explained
- CAP Theorem for Databases: Consistency, Availability & Partition Tolerance
- Data Ethics for Companies