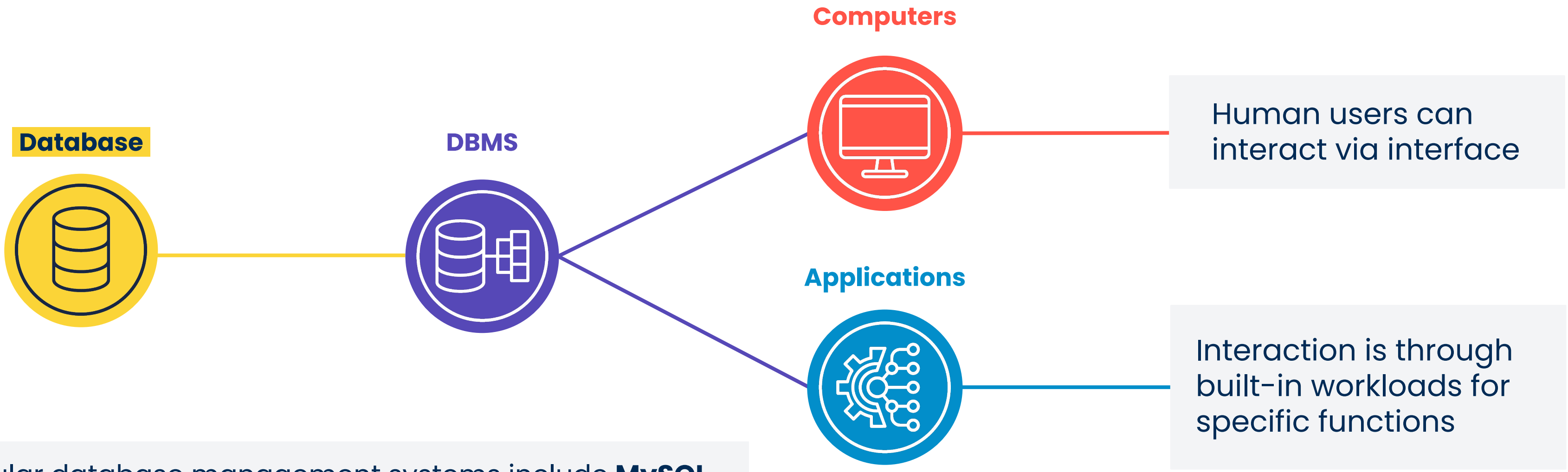
Database concepts

Relational database management systems

What is a database management system?

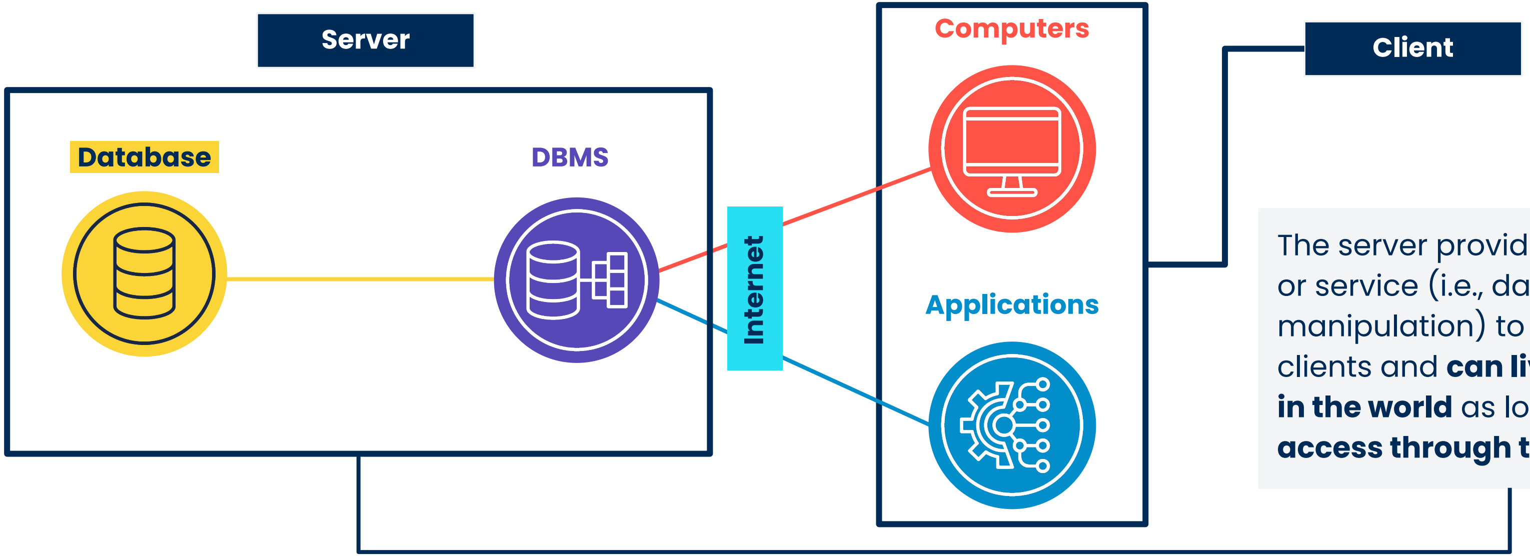
A database management system (**DBMS**) is system software that allows users to **create**, **store**, **retrieve**, and **run queries on data** stored in a database. It acts as an **interface** between an end user or an application and a database.



Popular database management systems include **MySQL**, **SQLite**, **SQL Server**, **PostgreSQL**, **Oracle database**, etc.

Client-server architecture for a DBMS

The client-server architecture is the **framework** in which the DBMS lives. The clients are represented by computers and applications, while the server is where the database and the DBMS live.



The server provides a function or service (i.e., data manipulation) to one or more clients and **can live anywhere in the world** as long as there is **access through the Internet**.

Client-server architecture pros and cons

Advantages

- Centralised data management
- Simultaneous access to a database by multiple clients and users
- Security and access control
- Allows for scaling as the need for a user base grows

Disadvantages

- As the server acts as a central point, an outage or failure can disrupt the whole system
- Maintenance and upgrades of the server infrastructure require resources
- Network dependency can disrupt effective communication
- Increased network latency can affect real-time or performance-sensitive applications

The purpose of database management systems

01. Data security

Incorporate security measures to **protect data** from **unauthorized access**, **manipulation**, or **breaches**.

02. Data manipulation and storage

Allow users to perform operations such as **inserting**, **deleting**, and **modifying** records. They manage the allocation of storage space, efficient **storage mechanisms**, and **data file management.**

03. Data backup and recovery

Allow for regular backups of the database, which can be used to **restore data in case of system failures**, **data corruption**, or other **unexpected events**.

04. Data integrity

Implement various constraints, ensuring that data remains **accurate**, **consistent**, and **reliable**.

05. Data sharing and collaboration

Enable **multiple users to access** and work with the same data concurrently, ensuring **data consistency**.

06. Data scalability

Handle **increasing volumes of data and growing user demands**. Scalability options include **partitioning or clustering** to distribute and manage data across multiple servers.

Types of database management systems

Database management systems can be **categorized based on various criteria,** such as the number of users or the data model. The most common types include:

01. Hierarchical DBMS

- Data are organized in a **tree-like structure**, where each parent node or record is linked to one or more child nodes, forming a **parent-child relationship**.
- For example: IBM's Information Management System (IMS)

02. Network DBMS

- Data are organized in a **graph-like structure**, where records are **connected by links** that represent their relationship. Child nodes or records can have **multiple parent** nodes or records, unlike the hierarchical model.
- For example: IDMS (Integrated Database Management System)

03. Relational DBMS

- Data are organized into **tables consisting of rows and columns**, where each table represents an entity or a relationship between entities.
- For example: Oracle, MySQL, Microsoft SQL Server, PostgreSQL, SQLite

Relational database management systems

Relational database management systems (RDBMSs) are the most widely used DBMSs because of their **efficiency** in data **standardization**, **querying**, and **relationships**.

Standardization

RDBMSs adhere to industry-standard query language **SQL (Structured Query Language)…** SQL provides a standardized and efficient way to interact with databases.

Data querying

Through SQL, users can query databases to **extract specific information** and perform **data manipulations** based on various conditions, filters, and sorting requirements.

Data relationships

They handle relationships between tables, hence allowing for the modeling of complex **associations**, and they support **efficient querying** and retrieval **of related data.**

RDBMSs support CRUD operations

CRUD is an acronym that represents the four basic operations (**create, read, update, delete**) that can be performed on data within an RDBMS.

Why CRUD operations?

Their support enables the standardization of the creation, retrieval, modification, and deletion of data, providing a robust foundation for managing and manipulating relational data.

CRUD operations are usually **supported by query languages like SQL**.

RDBMSs support CRUD operations

Create

- Enables the creation of new records by inserting data into database tables.
- Usually achieved through query statements, e.g. INSERT INTO for SQL.

Read

- Enables the retrieval of data from database tables using SQL statements such as SELECT.
- The RDBMS executes the query and retrieves the matching data.

Update

- Allows modification of existing data in a database through SQL UPDATE statements.
- Verifies the constraints and applies the changes to the matching records, updating the data in the database accordingly.

Delete

- Enables permanent removal of records from the database tables using SQL DELETE statements.
- Verifies constraints and deletes the matching records from the table, permanently removing them from the database.

ACID properties in RDBMSs

Relational database management systems should have **ACID** properties which ensure data validity and compliance. ACID is an acronym for **atomicity**, **consistency**, **isolation**, and **durability**.

A **transaction** is a **single unit** of work involving one or more operations, performed on a database with the aim of reading or modifying the data.

Transactions should follow ACID properties that guarantee the utmost data reliability and integrity.

For example, in the event of a power outage, the **absence of ACID properties** could mean that some of the **modifications made** to the database would **not be saved** causing inconvenience.

ACID properties of queries

A Atomicity

- Ensures that **all operations in a transaction or query** (to read, write, update, or delete data) are treated as a **single unit**.
- Meaning either the **entire query** is executed successfully if run**, or none of it is executed**.

C Consistency

- Ensures that transactions or queries only make changes to tables in **predefined, predictable ways**.
- This guarantees that errors or corruption in our data do not result in unintended consequences that **compromise the integrity of the database table**.

I Isolation

- Ensures that transactions or queries by multiple users on the same database **do not interfere with or affect one another**.
- Each query request can occur as though it were occurring one by one, even though they are simultaneously occurring.

D Durability

- Guarantees that modifications made to our data through successfully executed transactions will be **permanently saved**.
- This will remain the case in the event of a system failure.