HyperionDev

# Databases

## Task

Visit our website

# Introduction

In today's digital age, data has become one of the most valuable assets for individuals and organisations alike. The ability to efficiently store, manage, and retrieve data is crucial for making informed decisions, streamlining operations, and gaining competitive advantages. This lesson will discuss databases and database management systems, exploring their types, components, and importance in modern computing environments.

# Data systems

Storing data in a database is only one component of data management. Data systems are comprised of five main components:

1. **Data:** Stored in a database and organised according to the database's structure and rules.
2. **Software:** This includes the operating system, database management system (DBMS), as well as application and utility software necessary for database operations.
3. **Hardware:** The physical infrastructure that supports the database, including servers, storage devices, and networking equipment.
4. **People:** Individuals responsible for database design, use, and maintenance, such as database administrators, developers, and end-users.
5. **Procedures:** Documented processes and policies for database operations, maintenance, and security.

Depending on an organisation's needs, a data system may simply be a local machine with a single user, or a large complex system with a distributed network of servers with a team of developers and administrators.

We will focus on the types of databases and the database management systems used to manage them.

# Databases

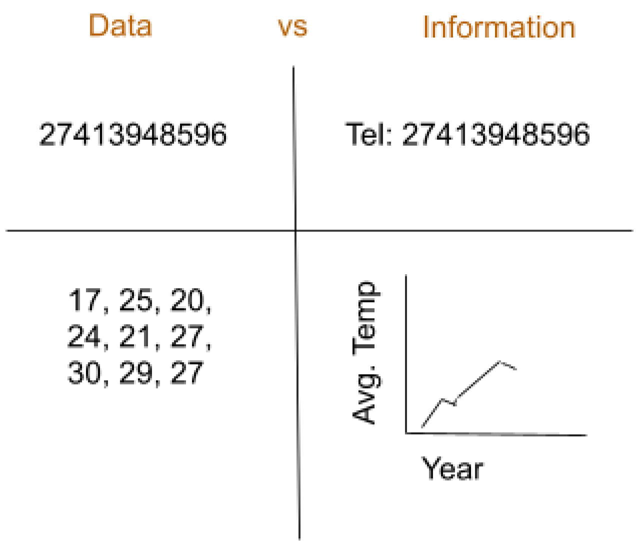
A database is a structured collection of data organised in a way that allows for efficient storage, retrieval, and manipulation. It goes beyond simple data storage by providing mechanisms for maintaining data integrity, such as defining specific data formats for data entry and storing metadata (or data about the data).

Data is the lifeblood of information systems, but it's not just about raw facts and figures. To fully understand the concept of databases, we must first distinguish between data and information, and recognise the role of metadata in organising and interpreting data.

## Data vs information

Data refers to raw, unprocessed facts, figures, or symbols that, on their own, may not have much meaning. For example, the number "42" is a piece of data. Information, on the other hand, is data that has been processed, organised, and given context to provide meaning. If we know that "42" represents the age of a person named John, it becomes information.

Data processing refers to a whole spectrum of activities: it can be as simple as organising data to identify patterns in it, or as complex as using statistical modelling to draw inferences from your dataset. Consider the following image:



Can you see how providing context and presenting data in a different format transformed the string and list of numbers into meaningful information? Recording metadata, or data that describes the data, is a simple form of data processing essential for extracting meaning from data. Metadata includes information such as units of measurement, the time period the data represents, or labels.

Decision-making relies on information, not just raw data. However, useful information requires accurate data, which must be generated properly and stored in a format that is easy to access and process. Just like any other basic resource, the data environment should be carefully managed. We will learn more about data management in this task.

# Types of databases

There are many different types of databases. These can be categorised according to the number of users who access the database, the physical location where data is stored, the type of data, the intended purpose of the data, and the degree to which the data is structured. We will focus on classifying data by purpose and structure.

## Purpose

In a business context, one of the most practical ways to define a database is based on how the data will be used and the urgency with which it needs to be accessed. Typically this results in two database categories: operational and analytical.

### Operational databases

Operational databases are designed to support the day-to-day operations of an organisation. They store and manage current, detailed data essential for running the business. Often referred to as **OLTP (online transaction processing)** databases, they are optimised for speed and efficiency in handling frequent updates and retrievals. Examples of where an operational database may be used are inventory management, order processing, and financial transactions.

### Analytical databases

Analytical databases, often referred to as **OLAP (online analytical processing)** databases, are designed for data analysis and business intelligence. Unlike operational databases, they store historical and aggregated data. These databases are optimised for complex queries and reporting, allowing businesses to gain insights into trends, patterns, and performance. Examples of analytical databases include data warehouses, data marts, and online analytical cubes, which support decision-making processes across various departments within an organisation.

## Structure

Databases can be classified based on their structure. **Structured** databases are highly organised with data arranged in a predefined format, similar to rows and columns in a spreadsheet. Relational databases are a common example of structured databases. **Semi-structured** databases have a degree of organisation but lack the rigidity of structured databases. Formats like **XML** and **JSON** are commonly used for semi-structured data. Finally, **unstructured** databases have no predefined format and can include various types of data such as text documents, images, and videos. Based on database structures, we can also group databases as relational or non-relational.

### Relational databases

**Relational databases** are the most widely used database type. They structure data into tables, composed of rows and columns, with related information linked between them. This means that different tables can be connected based on shared information, creating a structured network of data. For example, a "Customers" table might contain customer details, while an "Orders" table would hold order information. A common field, such as "Customer ID", could link these tables, allowing you to find all orders for a specific customer. **SQL** (Structured Query Language) is the standard language for interacting with and manipulating data in relational databases, enabling efficient data retrieval and interaction across related tables.

**Spot check 1**

Based on what you have learned about databases, would you define a spreadsheet as a type of database?

### Non-relational databases

**Non-relational databases**, often referred to as **NoSQL** (Not only SQL) databases, offer a more flexible approach to data storage and management. Unlike their relational counterparts, NoSQL databases don't enforce a rigid structure, allowing for diverse data models and scalability. Designed to handle massive amounts of unstructured or semi-structured data, NoSQL databases have gained prominence in today's data-intensive applications.

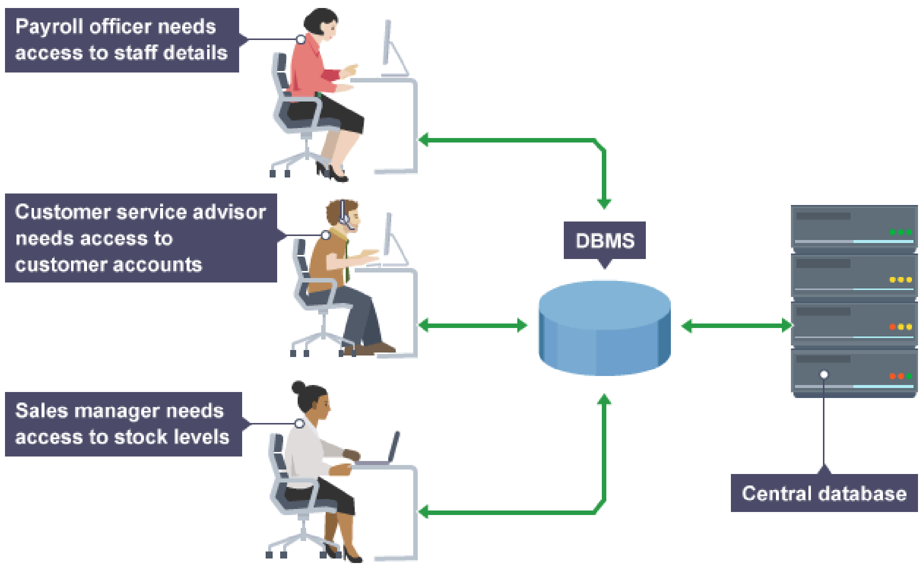
### Extra resource

You can find out more about **NoSQL databases** if you are interested in reading further.

To effectively manage and interact with these diverse database types, a specialised software system known as a database management system is employed.

## Database management systems

A **database management system (DBMS)** is a collection of programs that works to manage the database structure and control access to the data stored in the database. The DBMS serves as an intermediary between the user and the database, receiving all application requests and translating them into the complex operations required to fulfil those requests.


Payroll officer needs access to staff details
Customer service advisor needs access to customer accounts
Sales manager needs access to stock levels
DBMS
Central database

*Users accessing data through the DBMS (Gupta, n.d.)*

Much of the database's internal complexity is hidden from the application programs and end-users by the DBMS.

An important feature in most relational DBMSs are the built-in mechanisms to ensure data integrity and consistency through ACID properties. ACID stands for atomicity, consistency, isolation, and durability.

- Atomicity is when data can not be divided into smaller units.
- Consistency refers to the correctness of the database.
- Isolation means that parallel transactions should be able to happen without affecting each other.
- Durability implies that changes are permanent and can be recovered in the event of a system crash.

Essentially, ACID guarantees that database transactions are complete, accurate, independent, and permanent, even in the event of system failures. While some NoSQL databases offer ACID compliance for specific use cases, they are generally known for prioritising other characteristics like scalability and performance.

Ultimately, a DBMS helps make data management efficient and effective by providing:

- **Better data access and security:** By managing the database and controlling access to data within it, the DBMS makes it possible for end-users to have efficient access to data while enforcing data privacy and security policies.
- **Improved data integration:** By facilitating a variety of end-users to access data in a well-managed manner, the DBMS helps provide a clearer and more integrated view of the organisation's operations to the end-users.
- **Smooth data manipulation:** By making it possible to produce quick answers to spur-of-the-moment requests for data manipulation (e.g., to read or update the data).
- **High-quality information:** To make quicker and more informed decisions that increases end-user productivity.

As useful as data management systems are they do not come without challenges, especially where complex data systems are required. Increased complexity results in higher costs to maintain and update the hardware and software. Also, specialised personnel need to be hired or trained as vendors upgrade products with new features (Tutorialink, n.d.).

Next, let's have a look at some of the most widely used DBMSs.

## Popular DBMSs

While new DBMSs have entered the market and become popular for certain use cases, some have remained consistently popular. Some of these popular DBMSs are listed below.

- **MySQL** is a widely adopted open-source **relational database management system** (RDBMS) renowned for its ease of use, speed, and reliability. It's highly popular for web applications and smaller-scale projects due to its open-source nature, strong community support, and extensive documentation. MySQL is particularly favoured for its performance in read-heavy environments and its straightforward setup.
- **MariaDB** is an open-source RDBMS written in C and C++, and is a MySQL fork. It's particularly useful for web development and software applications where efficiently handling and storing data is essential. It has several features that are unavailable on other open-source RDBMSs, such as encryption at rest, compatibility with OraclePL and SQL, and columnar storage (DB-Engines, n.d.). These enhancements make it a robust choice for web development and enterprise applications where performance and security are critical.
- **PostgreSQL** is a powerful and open-source RDBMS renowned for its advanced features and support for complex data types. It includes capabilities such as full-text search, geospatial data handling, and extensive SQL compliance. PostgreSQL's strong emphasis on ACID compliance ensures data integrity and makes it suitable for a wide range of applications, from small projects to large-scale systems.
- **SQLite** is a lightweight, file-based relational database that does not require a server configuration. Its simplicity and self-contained nature make it ideal for mobile applications, embedded systems, and small-scale projects. Despite its compact size, SQLite offers reliability and speed, making it a popular choice for applications that require a straightforward, serverless database solution.
- **Oracle RDBMS** is a leading enterprise-level RDBMS known for its scalability, comprehensive feature set, and high reliability. It's designed to handle large-scale, complex workloads and offers advanced functionalities such as enhanced security features, data warehousing, and analytics. Oracle's robust support and extensive capabilities have made it a preferred choice for many large organisations and mission-critical applications.

- **MongoDB** is a NoSQL database. It's defined (MongoDB, 2017) as "an open-source database used by companies of all sizes, across all industries, and for a wide variety of applications." It's "an agile database that uses a flexible document data model so schemas can change quickly as applications evolve, [and] provides functionality developers expect from traditional databases, such as secondary indexes, a full query language, and strict consistency."

## Extra resource

Explore how the **historical trend of popular DBMSs** has changed over time, and compare system properties of different DBMSs, for example, **MariaDB vs MySQL**.

While a DBMS facilitates data integrity, accessibility, and performance, database design is crucial for effective database management. A poorly designed database can lead to performance issues, data anomalies, and increased maintenance costs. In the next task, we will focus on designing and normalising a relational database.

## Reflection

Consider what design choices you need to make to create **GDPR**-compliant databases.

## Extra resource

If you'd like to know more about database design, we recommend reading the book ***Database Design*** (2014) by Adrienne Watt.

## Practical task

Create a document called **MyDatabaseTheory** and answer the following questions:

1. Explain the difference between data and information.
2. What is metadata?
3. What is a DBMS, and what are its advantages?
4. Compare and contrast operational databases with analytical databases, and provide an example of each.
5. Explain the types of data and use cases where NoSQL databases are most effective.
6. Which DBMS does not require server configuration, and what are the advantages of using it?
7. Explain ACID properties in the context of a DBMS.

Convert your **MyDatabasesTheory** document to **PDF format** before submitting it.

**Important:** Be sure to upload all files required for the task submission inside your task folder and then click "Request review" on your dashboard.

## Spot check 1 answers

Spreadsheets, while capable of manipulating data in a tabular format, lack fundamental database functionalities. They do not support self-documentation through metadata, which is essential for understanding data structure. Additionally, spreadsheets cannot enforce data types or domains, leading to inconsistencies within data columns. Furthermore, they are unable to define relationships between different tables.

## Share your thoughts

Please take some time to complete this short feedback **form** to help us ensure we provide you with the best possible learning experience.

## Reference list

DB-Engines. (n.d.). *MariaDB system properties*. Retrieved July 10, 2024, from **https://db-engines.com/en/system/MariaDB**

MongoDB. (2017). *MongoDB overview*. Amazon AWS. **https://s3.amazonaws.com/info-mongodb-com/MongoDB_Datasheet.pdf**

**https://scalegrid.io/blog/2019-database-trends-sql-vs-nosql-top-databases-single-vs-multiple-database-use/**

Tutorialink. (n.d.). *Advantages and disadvantages of DBMS*. **https://tutorialink.com/dbms/advantage-and-disadvantages-of-dbms.dbms**