Chapter 1: The Database Approach

# Data vs Information

**Data** are raw facts that have not been processed to reveal their meaning. They are the basic building blocks from which information is derived.

**Information** is what you get when you process data to uncover its significance. It involves organizing or analyzing data to identify patterns, make predictions, or draw conclusions.

The transformation from data to information may involve summarizing raw data into a more understandable format, such as a report or a graph, which can then be used for decision-making.

For effective use, raw data must be formatted correctly for storage and processing. This could mean converting survey responses into a standardized format or handling complex data types like multimedia.

Accurate, relevant, and timely information is crucial for making informed decisions, which are essential for the survival and success of organizations in a competitive global market.

Information serves as the foundation for knowledge, which is an informed understanding of a particular subject area. Knowledge allows for the creation of new insights from existing information.

In essence, data needs to be collected, processed, and formatted into information, which can then be used to support decision-making and generate knowledge within an organization or environment.

## Data Quality & Data Governance

The accuracy, relevance, completeness, timeliness, uniqueness, and clarity of data are crucial for making sound business decisions. **High-quality data** can lead to the development of strategies that enhance an organization’s income generation.

Organizations must comply with data protection laws such as GDPR in Europe and POPIA in South Africa, which regulate the collection, processing, and protection of personal information.

**Data Governance** refers to the policies and procedures established by an organization to manage the availability, usability, integrity, and security of data. It includes defining data ownership and authorization for data manipulation.

As part of data governance**, Master Data Management (MDM)** ensures consistency and accuracy of data across all systems and supports auditing, reporting, and compliance.

Implementing a data governance strategy is complex and requires involvement across the organization. It requires time to ensure data compliance with the strategy, and ongoing monitoring is essential to maintain its relevance and effectiveness.

In essence, maintaining high-quality data and implementing a robust data governance strategy are essential for organizational success and compliance with legal standards. Regular monitoring with data profiling and quality tools helps keep track of data and ensures continuous alignment with business objectives.

# Data vs Information

Efficient data management is often facilitated by the use of a computer database, which serves as a shared and integrated structure for storing two main types of data:

* **End-User Data**: These are the raw facts that are of interest to the end user. They represent the actual content that the database is designed to manage.
* **Metadata**: This is data about data. Metadata describes the characteristics of end-user data and the relationships among them, enabling the integration and management of the user data.

Metadata acts as a guide to the database, much like a card catalog in a library. It helps users understand what data is available and how it can be accessed or related to other data.

Metadata in a database management system (DBMS) refers to data that describes other data within the database. It provides context and additional information about the data stored, such as the structure of the database, the data types, and the relationships between different data elements. Here are some examples of metadata in a DBMS:

* **Table Metadata**: Information about database tables, including table names, sizes, and the number of records.
* **Column Metadata**: Details about the columns within a table, such as column names, data types, and any constraints applied.
* **Index Metadata**: Descriptions of the indexes on tables, which can include the index name, type, and the columns it affects.
* **Constraint Metadata**: Information about constraints like primary keys, foreign keys, unique constraints, and check constraints.
* **View Metadata**: Details about database views, including view definitions and the tables they reference.
* **Stored Procedure Metadata**: Information about stored procedures, such as their names, parameters, and underlying SQL code.
* **User Metadata**: Data about database users, including user privileges, roles, and permissions.
* **Backup Metadata**: Details about database backups, such as backup schedules, locations, and recovery plans.
* **Performance Metadata**: Information related to the performance of the database, like query execution plans and optimization statistics.

Metadata is crucial for effective data management, governance, and understanding the schema and components of a database. [It can be accessed and queried through SQL, specialized APIs, and system catalog views](https://databasetown.com/what-is-database-metadata-examples-types/).

A database can be likened to a well-organized electronic filing cabinet where the database management system (DBMS) is the software that manages the filing cabinet’s contents. The DBMS is a collection of programs that:

* Manages the structure of the database.
* Controls access to the data stored in the database.
* Facilitates the creation, retrieval, update, and deletion of the data.

In summary, databases and DBMSs are essential for organizing large amounts of data in a way that makes them easily accessible, manageable, and secure. They are fundamental tools in the information age, enabling efficient data management and utilization for various applications.

## Roles & Advantages of the DBMS

The role of a Database Management System (DBMS) is to act as an intermediary between end users and the database itself.

The DBMS acts as a middleman, receiving requests from application programs (which could be written in languages like Python, Visual Basic, C++, or Java) and translating them into the complex operations needed to fulfill those requests.

**The DBMS** abstracts the internal complexities of the database from the users and application programs, allowing them to interact with the database without needing to understand its underlying intricacies.

The DBMS allows data to be shared across multiple applications or users, facilitating a collaborative environment where data can be accessed and manipulated by authorized parties.

It integrates various user views of the data into a single, comprehensive data repository, providing a unified view of the organization’s operations.

By managing data centrally, the DBMS minimizes inconsistencies that might occur when the same data is stored in different locations or formats. For example, data inconsistency exists when a company’s sales department stores a sales representatives name as Thobile Cele, and the company’s personnel department stores that same person’s name as Bathobile M. Cele or when the company’s regional sales office shows the price of product X as R390.00 in South African currency and its national sales office shows the same products price as R350.00. The probability of data inconsistency is greatly reduced in a properly designed database.

The DBMS enhances the ability to access data quickly, especially for ad hoc queries, which are spontaneous questions that require immediate answers.

When a query is issued, the DBMS processes it and returns the result set to the application, enabling users to obtain the information they need promptly.

In essence, the DBMS simplifies and streamlines the interaction between end users and the database, ensuring data is managed efficiently, consistently, and accessibly. This makes it an essential component of modern data management and information systems.

## Types of Databases

There are various ways to classify databases within a Database Management System (DBMS). Let’s delve into each classification in detail:

1. **By Number of Users**:
   * **Single-User Database**: This type supports only one user at a time. If one user is accessing the database, others must wait their turn. When such a database is run on a personal computer, it’s referred to as a **desktop database**.
   * **Multi-User Database**: This type supports multiple users simultaneously. If it’s designed for a small group (usually fewer than 50 users) or a specific department, it’s called a **workgroup database**. If it’s intended for an entire organization with many users (often hundreds) across various departments, it’s known as an **enterprise database**.
2. **By Location**:
   * **Centralized Database**: This database stores all data on a single site.
   * **Distributed Database**: This type supports data distributed across multiple sites. The management and extent of distribution are complex topics, typically covered in advanced discussions like Chapter 14 on Distributed Databases.
3. **By Usage and Time Sensitivity**:
   * **Operational Database**: Also known as an **Online Transaction Processing (OLTP)**, transactional, or production database, this type is designed to support a company’s day-to-day operations. It handles transactions that require immediate and accurate recording, such as sales, payments, and purchases.
   * **Analytical Database**: This category includes databases designed for decision support and typically consists of two main components:
     + **Data Warehouse**: A specialized database that stores historical data from operational databases and other external sources in a format optimized for decision support.
     + **Online Analytical Processing (OLAP)**: A set of tools that provide an environment for advanced data analysis, allowing for the retrieval, processing, and modeling of data from the data warehouse.
4. **By Structure**:
   * **Structured Data**: Databases that store data in a fixed format, often in rows and columns like traditional relational databases.
   * **Unstructured Data**: Databases that handle data without a fixed format, such as text documents, images, and videos.
   * **Semi-Structured Data**: Databases that may not fit into traditional database structures but still contain tags or markers to separate semantic elements and enforce hierarchies of records and fields.

In recent times, this area of database application has grown in importance and usage, to the point that it has evolved into its own discipline: business intelligence. The term business intelligence describes a comprehensive approach to capturing and processing business data with the purpose of generating information to support business decision making. It leverages both operational and analytical databases to provide comprehensive insights into business operations.

Understanding these classifications helps in designing and selecting the appropriate database system based on the specific needs and context of an organization. Each type of database serves different purposes and offers various benefits, from transactional efficiency to analytical depth.

Unstructured and semi-structured data storage and management needs are being addressed through a new generation of databases known as **XML databases**. **Extensible Markup Language (XML)** is a special language used to represent and manipulate data elements in a textual format. An XML database supports the storage and management of semi-structured data XML data.

1. **XML Databases**:

* **XML (Extensible Markup Language)** is a flexible language used to represent and manipulate data elements in a textual format.
* **XML databases** are designed to store and manage semi-structured XML data. They are part of a broader category of NoSQL databases known as document-oriented databases.
* These databases can handle data that doesn’t fit neatly into traditional relational databases, such as deeply nested or sparse data[1](https://en.wikipedia.org/wiki/XML_database).
* XML databases can be **XML-enabled**, meaning they can store XML data within a traditional relational structure, or **Native XML databases**, which are optimized for working with XML data.

1. **Analytical Databases vs. Data Warehouses**:

* **Analytical databases** are used for interactive data exploration and advanced data analysis. They are ideal for performing calculations, aggregations, and data manipulation required for tactical or strategic decision-making[2](https://www.heliosz.ai/blogs/analytical-database-vs-data-warehouse-comparative-review/).
* **Data warehouses**, on the other hand, serve as centralized repositories for integrating and storing historical data. They support reporting, decision-making, and business intelligence by providing a structure that is different from transactional databases[3](https://www.coursera.org/articles/data-warehouse-vs-database).
* The primary focus of a data warehouse is to store data that helps generate information for making informed decisions based on historical data from operational databases and other sources.
* Data warehouses are updated periodically and may not reflect real-time data, but they allow for the analysis of large and diverse datasets to identify patterns and insights.

In summary, XML databases address the need for storing non-traditional data formats, while analytical databases and data warehouses provide the infrastructure for analyzing business data and supporting decision-making processes. Each has its unique structure and purpose, catering to specific data management and analysis needs within an organization.

A table with a number of data

Description automatically generated

The advent of the World Wide Web and the proliferation of internet technologies have given rise to a new era of social media, characterized by the ability to connect and interact seamlessly across the globe at any time. This has led to an explosion in the volume of data generated, as platforms like Google, Facebook, Instagram, Twitter, and LinkedIn continuously collect detailed information about their users’ activities, preferences, and interactions.

This surge in data has necessitated the development of specialized database systems capable of handling the scale and complexity of the information. Traditional relational databases, which organize data into predefined tables and columns, are not always well-suited to manage the variety and volume of data produced by social media interactions. As a result, a new category of databases, known as NoSQL databases, has emerged.

**NoSQL databases** are designed to store, manage, and analyze vast quantities of unstructured or semi-structured data. They are called “Not Only SQL” because they do not rely solely on the structured query language (SQL) used by traditional relational databases. Instead, NoSQL databases offer a more flexible and scalable approach to data management, which is particularly beneficial for social media applications where data structures can change rapidly and need to scale to accommodate millions of users.

NoSQL databases come in various forms, each optimized for specific types of data and use cases. For instance, **document-oriented NoSQL databases** are ideal for storing and querying data that is semi-structured, like **JSON documents** commonly used in web applications. **Key-value stores** are another type of NoSQL database that **pairs unique keys with corresponding values**, making them suitable for applications that require fast retrieval of data, such as user profiles or session informa-=tion.

The flexibility of NoSQL databases allows developers to iterate quickly and adapt to changing requirements without the constraints of a fixed schema. This agility is crucial for social media platforms, where user behavior and features can evolve rapidly. Moreover, NoSQL databases are designed to scale horizontally across multiple servers, providing the robustness and performance needed to support the high user loads and real-time data processing that social media demands.

In summary, NoSQL databases represent a significant shift in database technology, offering a solution to the challenges posed by the vast and diverse data landscape of social media. Their ability to handle unstructured data, scale efficiently, and provide flexibility in data modeling makes them an essential component of the modern data ecosystem. As social media continues to grow and evolve, NoSQL databases will remain at the forefront of enabling the storage and analysis of the digital interactions that define our connected world.

# Why Database Design is Important

**Database design** refers to the activities that focus on the design of the database structure that will be used to store and manage end-user data. It’s a deliberate and strategic activity, not a product of chance, and it’s critical to the success of any database implementation.

The design process begins with a clear understanding of the intended use of the database. For instance, a transactional database, which is central to day-to-day operations, must prioritize accuracy, consistency, and speed. On the other hand, a data warehouse, which is used for analytical purposes, focuses on historical and aggregated data. The requirements for these two types of databases are distinct, and so are their design approaches.

Moreover, the environment in which the database will operate also influences its design. A centralized, single-user database differs significantly from a distributed, multi-user database. The former may be simpler and more straightforward, while the latter requires a more complex design to ensure data integrity and performance across different locations and concurrent users.

The importance of database design cannot be overstated. A well-designed database ensures efficient data management and the generation of reliable information, which is crucial for informed decision-making. Conversely, a poorly designed database can lead to errors that are hard to detect and correct, potentially resulting in poor decisions that could jeopardize the success of an organization.

Given the stakes, database design is a skill that is highly valued and sought after. It’s a subject of study in universities, a topic for professional seminars, and a field where consultants can thrive. The investment in learning and applying sound database design principles is a testament to its significance in the realm of data management. It’s an investment that pays dividends by enabling organizations to leverage their data assets effectively and avoid the pitfalls of a subpar database structure.

# Historical Roots and File Data Processing

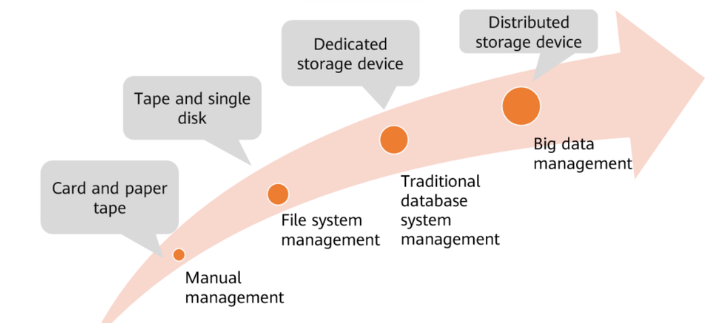
The evolution of databases is a journey from simplicity to complexity, from **manual systems** to sophisticated, **computerized structures**. Initially, data was managed through manual file systems, which involved physical folders and filing cabinets. This method was labour-intensive and time-consuming, as it required manual retrieval and organization of information.

* Manual file systems are accomplished through a system of file folders and filing cabinets.
* With computerized file systems, a **data processing (DP) specialist** created a computer-based system to track data and produce required reports.
* File System Redux: Modern End-User Productivity Tools
  + Business users widely use spreadsheet programs such as Microsoft Excel to enter data in a series of rows and columns to manipulate data.
  + Acommon misuse of spreadsheets is as a substitute for a database.

|  |  |
| --- | --- |
| **Term** | **Definition - Basic File Terminology** |
| Data | Raw facts, such as a telephone number, a birth date, a customer name, and a year-to-date (YTD) sales value. Data has little meaning, unless it has been organized in some logical manner. |
| Field | A character or group of characters (alphabetic or numeric) that has a specific meaning. A field is used to define and store date. |
| Record | A logically connected set of one or more fields that describes a person, place, or thing. For example, the fields that constitute a record for a customer might consist of the customer’s name, address, phone number, date of birth, credit limit, and the unpaid balance. |
| File | A collection of related records. For example, a file might contain data about the students currently enrolled at Alamo College. |

**Problems with file systems:**

* Lengthy development times
* The difficulty of getting quick answers
* Complex system administration
* Lack of security and limited data sharing
* Extensive programming



As the need for faster data processing grew, computerized file systems emerged. These systems allowed for quicker data retrieval and reporting but still had limitations. They often led to data redundancy, inconsistency, and were difficult to manage as the volume of data increased. The file terminology that evolved during this period included terms like ‘records’, which are collections of related fields, and ‘fields’, which are specific pieces of data within a record.

The advent of Database Management Systems (DBMS) revolutionized data processing by overcoming many of the limitations of file systems. DBMS provided a more structured approach to data management, allowing for better data integrity, reduced redundancy, and easier access to information. However, the transition to DBMS did not eliminate the challenges of data management; it merely provided tools to address them more effectively. Database designers and developers needed to understand the inherent problems of data management to avoid replicating the same issues within a DBMS environment.

The history of databases reflects the evolution of organizational needs and technological capabilities. From manual file systems to hierarchical and network databases, and eventually to the relational model, each step represented an advance in how data could be stored, accessed, and manipulated. The relational model, in particular, introduced by E.F. Codd in the 1970s, became the foundation for modern database systems, emphasizing the importance of data relationships and integrity.

Today, we have a plethora of database models, including NoSQL databases, which address the needs of large-scale, distributed systems and unstructured data typical of big data and social media applications. These systems are designed to handle the volume, velocity, and variety of data that characterize the digital age.

In conclusion, understanding the history of databases—from manual file systems to the latest NoSQL technologies—is crucial for database professionals. It provides context for the challenges of data management and the rationale behind the design and functionality of current database systems. This knowledge is essential for creating effective database solutions that leverage the strengths of modern DBMS while avoiding the pitfalls of the past.

## Structural and Data Dependence

Structural and data dependence are concepts related to the flexibility and adaptability of database systems, particularly in how they handle changes to the database schema.

**Structural Dependence**:

* Structural dependence refers to a situation where changes in the database schema affect the ability of the DBMS to access data. For example, if altering the structure of a table requires changes to all applications that access this table, the system exhibits structural dependence.
* This was a common issue in early file-based systems, where a change in the file format would necessitate rewriting all programs that accessed the file.

**Data Dependence**:

* Data dependence is a similar concept where changes in the schema at the logical level (like changing the data type of a field) would require changes in all programs that accessed the data.
* This creates a tight coupling between the data and the applications, making it difficult to evolve the database design without affecting the applications.

To overcome these limitations, modern database systems strive for **data independence**, which is the capacity to change the schema at one level of a database system without having to change the schema at the next higher level. There are two types of data independence:

1. **Physical Data Independence**: The ability to modify the physical schema without affecting the logical schema. For instance, changing how the data is stored on disk won’t affect the database’s tables, views, or other logical structures.
2. **Logical Data Independence**: The ability to change the logical schema without affecting the external schema or applications. This means that you can alter the logical design of the database—like adding a new column to a table—without needing to change the applications that use the database.

Achieving data independence is crucial for the longevity and adaptability of a database system. It allows the database to evolve and grow without imposing a heavy maintenance burden on the applications that use it.

## Data Redundancy

Data redundancy in databases refers to the occurrence of the same data existing in multiple places within the database system. This can happen for various reasons and can have both positive and negative implications.

**Reasons for Data Redundancy**:

* **Accidental**: It can occur unintentionally due to inefficient coding or complex processes that lead to the same data being entered or stored more than once.
* **Intentional**: In some cases, redundancy is deliberate, such as for backup purposes or to ensure consistency across different systems.

**Problems Caused by Data Redundancy**:

* **Data Inconsistency**: If the same data exists in multiple places and one instance is updated without updating the others, it can lead to inconsistencies[2](https://www.geeksforgeeks.org/the-problem-of-redundancy-in-database/).
* **Increased Storage Costs**: Storing the same data multiple times requires more storage space, which can increase costs.
* **Update Anomalies**: When data needs to be updated, all instances of the redundant data must be found and updated, which can be time-consuming and error prone.

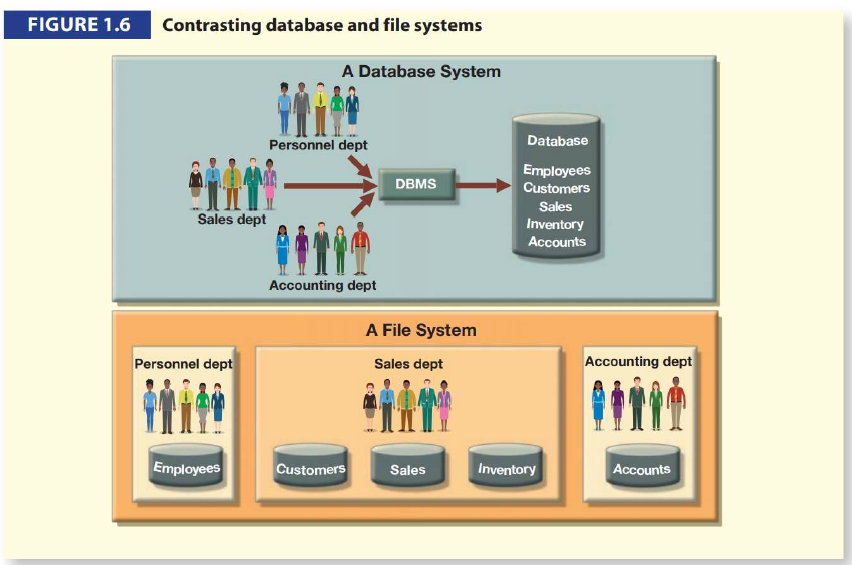
**Managing Data Redundancy**:

* **Normalization**: Structuring the database to minimize redundancy through normalization, which organizes data to reduce duplication.
* **Data Replication**: Using data replication to maintain consistent copies of data across different locations for accessibility and backup purposes.

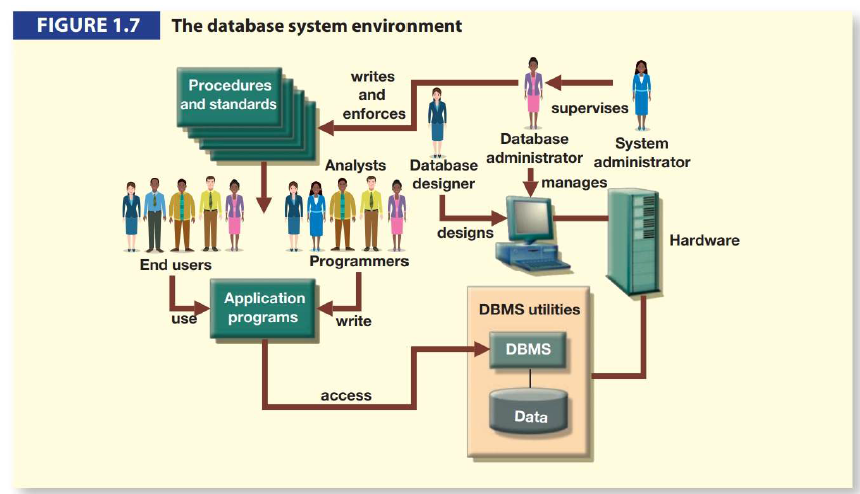
Data redundancy can be a challenge in terms of storage and consistency. The key is to manage redundancy carefully without leading to unnecessary complications.

# The DBMS

The term **database system** refers to an organization of components defining and regulating data collection, storage, management, and use within a database environment. Database solutions must be cost-effective as well as tactically and strategically effective.



* The database system is composed of the following five components:
  + **Hardware which** refers to all the systems physical devices for example, computers (microcomputers, mainframes, workstations, and servers), storage devices, printers, network devices (hubs, switches, routers, and fiber optics) and other devices (automated teller machines, ID readers, etc.).
  + **Software**. Although the most readily identified software is the DBMS itself, to make the database system function fully, three types of software are needed: operating system software, DBMS software, and application programs and utilities.
  + **People**. This component includes all users of the database system. Based on primary job functions, five types of users can be identified in a database system: systems administrators, database administrators, database designers, systems analysts and programmers, and end users.
  + **Procedures**. Procedures are the instructions and rules that govern the design and use of the database system. Procedures are a critical, although occasionally forgotten, component of the system. Procedures play an important role in a company because they enforce the standards by which business is conducted within the organisation and with customers. Procedures are also used to ensure that there is an organised way to monitor and audit both the data that enter the database and the information that is generated through the use of that data.
  + **Data**



## DBMS Functions

A DBMS performs the following important functions:

* Data dictionary management – The DBMS stores definitions of data elements and their relationships in a **data dictionary**
* Data storage management – The DBMS creates and manages the structures required for data storage.
  + **Performance tuning** ensures efficient performance.
* Data transformation and presentation – The DBMS transforms entered data to conform to required data structures.
  + Data is formatted to conform to the user’s logical expectations.
* Security management – The DBMS creates a system that enforces user security and data privacy.
* Multiuser access control – The DBMS uses sophisticated algorithms to ensure that multiple users can access the database concurrently without compromising its integrity.
* Backup and recovery management – The DBMS provides backup and data recovery to ensure data safety and integrity.
  + Recovery management deals with the recovery of the database after a failure
* Data integrity management – The DBMS promotes and enforces integrity rules, thus minimizing redundancy and maximizing data consistency.
* Database access languages and application programming interfaces – The DBMS provides data access through a query language.
  + A **query language** lets the user specify what must be done without having to specify how.
  + **Structured Query Language (SQL)** is the de facto query language and data access standard supported by many DBMS vendors.
* Database communication interfaces – A DBMS accepts end-user requests via multiple communication interfaces.