

# WQD 7007

---


## Summary

---

---

---

---



# Chapter 1

## Characteristics of Big Data along with examples:

**Volume:** Big Data refers to extremely large data sets that are too large to be processed using traditional data processing methods. The volume of Big Data is typically measured in petabytes, exabytes, or even zettabytes. Examples of high-volume Big Data include social media data, scientific research data, financial transaction data, and IoT data.

**Velocity:** Big Data is generated and processed at a high velocity, meaning that it is constantly being updated and changed. For example, social media data is generated in real-time, financial transaction data is processed in milliseconds, and IoT sensor data is collected at a high frequency.

**Variety:** Big Data comes in many different forms, including structured, semi-structured, and unstructured data. Structured data is organized in a defined format, such as a database table. Semi-structured data has some structure, but is not organized in a defined format, such as JSON files. Unstructured data has no defined structure, such as text data, images, and videos. Examples of high-variety Big Data include social media data, email data, web data, and sensor data.

**Veracity:** Big Data may contain errors, inaccuracies, or inconsistencies, which can affect its quality and reliability. Veracity refers to the quality and reliability of the data. For example, social media data may contain fake news or spam, financial data may contain errors or fraudulent transactions, and IoT sensor data may contain noisy or incomplete data.

**Variability:** Big Data can vary in its format, structure, and content, which can make it difficult to process and analyze. Variability refers to the inconsistencies and variations in the data. For example, social media data can come in different languages, formats, and sources, financial data can vary in its format and structure, and IoT sensor data can have different sampling rates and data types.

**Visualization:** Big Data can be difficult to understand and interpret without effective data visualization techniques. Visualization refers to the use of graphs, charts, and other visual representations to make Big Data more accessible and understandable. For example, a heat map can be used to visualize social media data geographically, a scatter plot can be used to visualize financial data trends, and a line chart can be used to visualize IoT sensor data.

**Value:** Big Data is valuable when it provides meaningful insights and helps to drive decision-making. Value refers to the usefulness and impact of the data. For example, social media data can be used to understand consumer sentiment and improve marketing strategies, financial data can be used to detect fraudulent transactions and manage risk, and IoT sensor data can be used to optimize manufacturing processes and reduce costs.

# Differences between Big Data and Small Data in each of the following areas:

**Goal:** The goal of Big Data is to discover insights and patterns in large datasets, while the goal of Small Data is to understand specific phenomena in a particular context.

**Location:** Big Data is often collected from a wide variety of sources, including social media, IoT devices, and scientific instruments, while Small Data is typically collected from a smaller number of sources, such as surveys, interviews, and experiments.

**(Variety)** Data structure and content: Big Data is often unstructured or semi-structured, meaning it has no predefined schema or is not organized in a specific format. On the other hand, Small Data is typically structured and organized in a predefined format.

**Data preparation:** Big Data requires significant effort to prepare and clean the data before analysis can be performed. This is because Big Data is often noisy, incomplete, and inconsistent. In contrast, Small Data is typically preprocessed before analysis, with outliers and missing values removed.

**Longevity:** Big Data is often stored for long periods of time, as it can continue to provide insights and value over time. Small Data is often discarded once the analysis is completed.

Measurement: Big Data requires specialized tools and techniques to measure, store, and process the large volumes of data. Small Data can often be measured using more traditional data collection and analysis methods.

Reproducibility: Because Big Data is often collected from many sources, it can be difficult to reproduce the same results from different datasets. Small Data is often more reproducible, as the data is collected in a controlled environment.

Stakes/cost: Big Data projects often involve large investments in hardware, software, and personnel. Small Data projects can be completed with fewer resources and a smaller budget.

Introspection/observation: Big Data relies on observation and analysis of patterns and trends in the data. Small Data often involves more introspection and exploration of individual cases or instances.

Analysis: Big Data analysis often requires specialized tools and techniques, such as machine learning and data mining. Small Data analysis can often be performed using more traditional statistical methods.

(Complexity)

# Six phases of Big Data:

**Data Generation:** The first phase of Big Data is the generation of data, which can come from a variety of sources such as sensors, social media platforms, website clicks, and customer transactions. This data can be in the form of text, images, videos, or numerical data.

**Data Acquisition:** Once the data is generated, it needs to be collected and acquired. This involves identifying the data sources, selecting the appropriate tools and techniques to collect the data, and transferring it to a centralized location for storage and analysis.

**Data Storage:** The next phase is to store the data in a way that allows for efficient retrieval and analysis. This involves selecting the appropriate data storage systems and technologies, such as Hadoop Distributed File System (HDFS), NoSQL databases, or data warehouses.

**Data Analysis:** The analysis phase involves exploring and analyzing the data to identify patterns, trends, and insights. This can be done using a variety of tools and techniques, such as data mining, machine learning, and statistical analysis.

**Data Visualization and Interpretation:** Once the data has been analyzed, it needs to be visualized and interpreted to make it easier for decision-makers to understand. This involves creating visualizations such as graphs, charts, and dashboards that can be used to communicate the insights discovered during the analysis phase.

**Decision Making:** The final phase is to use the insights gained from the data analysis to make informed decisions. This can involve implementing changes to business processes, launching new products or services, or making strategic investments. The decisions made in this phase are based on the insights gained from the data, and can have a significant impact on the success of the organization.

# Key obstacles in Big Data applications:

**Data representation:** One of the biggest challenges in dealing with Big Data is how to represent and store the data in an efficient and meaningful way. This is because Big Data can be extremely diverse and complex, with a wide range of different data types, structures, and formats. Choosing the right data representation method can help ensure that the data is easily accessible, searchable, and usable.

**Redundancy reduction and data compression:** Big Data often contains a lot of redundant and irrelevant information, which can make it difficult and time-consuming to process and analyze. Redundancy reduction and data compression techniques can help to remove duplicate or unnecessary data, which can help to improve processing speed, reduce storage requirements, and increase the accuracy of analysis.

**Data life cycle management:** Big Data is typically generated and consumed at a rapid pace, which can make it difficult to manage and maintain over time. Effective data life cycle management involves implementing policies and procedures for data collection, storage, processing, analysis, and disposal. This can help to ensure that the data remains accessible, secure, and accurate over its entire lifespan.

**Analytical mechanism:** Analyzing Big Data requires powerful and scalable analytical mechanisms, such as machine learning, data mining, and predictive analytics. These mechanisms must be able to process and analyze massive amounts of data in real-time, while also providing accurate and actionable insights.

**Data confidentiality:** Big Data often contains sensitive or confidential information, such as personal or financial data, which must be protected from unauthorized access or disclosure. This requires implementing strong data security and privacy measures, such as encryption, access controls, and secure data storage.

**Energy management:** Big Data processing and storage can consume significant amounts of energy, which can be costly and environmentally unsustainable. Energy management techniques, such as data center consolidation, virtualization, and energy-efficient hardware, can help to reduce energy usage and costs.

**Expendability and scalability:** Big Data applications must be able to scale and adapt to changing data volumes and processing needs. This requires implementing flexible and scalable infrastructure, such as cloud computing, that can accommodate growth and expansion.

**Cooperation:** Big Data often involves collaboration and cooperation among different teams, departments, or organizations. This requires effective communication, coordination, and governance mechanisms, to ensure that all stakeholders are aligned and working towards a common goal.



# Chapter 2

## Five pillars to Hadoop:

**Data Management:** This pillar is concerned with the storage and management of data in Hadoop. It includes the Hadoop Distributed File System (HDFS) which allows for the distributed storage of large data sets across multiple nodes in a Hadoop cluster. This pillar also includes other tools such as Apache HBase for managing large, structured data sets and Apache Hive for querying and analyzing data.

**Data Access:** This pillar focuses on providing different ways to access data stored in Hadoop. This includes tools such as Apache Pig and Apache Spark for data processing, Apache Sqoop for importing and exporting data from relational databases, and Apache Flume for collecting and ingesting data from various sources.

**Data Governance and Integration:** This pillar focuses on ensuring the quality and consistency of the data stored in Hadoop. This includes tools for metadata management such as Apache Atlas, which provides a centralized metadata repository, and Apache NiFi for data integration and data flow management.

**Security:** This pillar is concerned with ensuring the security of data stored in Hadoop. It includes tools for authentication and authorization such as Apache Ranger, which provides a centralized security framework, and Apache Knox for secure access to Hadoop services.

**Operations:** This pillar focuses on the management and monitoring of the Hadoop cluster. This includes tools such as Apache Ambari for cluster management and monitoring, and Apache Oozie for workflow management and scheduling.

# Data Management in Hadoop:

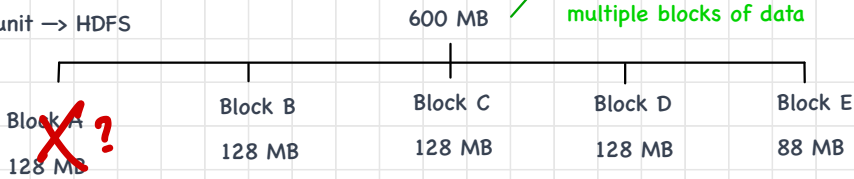
**YARN:** YARN (Yet Another Resource Negotiator) is a resource management framework in Hadoop that allows multiple data processing engines to run on the same Hadoop cluster. YARN is responsible for managing resources such as memory and CPU, and it ensures that applications running on the cluster have access to the necessary resources to run efficiently. YARN provides a flexible and scalable platform for running different types of applications in Hadoop, such as MapReduce, Spark, and Hive.

**MapReduce:** MapReduce is a programming model in Hadoop that allows for distributed processing of large data sets across a Hadoop cluster. MapReduce works by dividing the processing of data into two stages: map and reduce. The map stage takes in raw data and converts it into key-value pairs, which are then processed in parallel across multiple nodes in the cluster. The reduce stage takes the output from the map stage and combines the key-value pairs to produce a final output. MapReduce is a key component of Hadoop that allows for the distributed processing of large data sets.

**HDFS:** HDFS (Hadoop Distributed File System) is a distributed file system in Hadoop that provides a scalable and fault-tolerant storage system for large data sets. HDFS works by storing data across multiple nodes in a Hadoop cluster, with each node containing a part of the overall data set. HDFS is designed to handle large files and can replicate data across multiple nodes to ensure data availability and reliability. HDFS is used as the primary storage system for Hadoop and is a key component of Hadoop's data management capabilities.

# Hadoop:

1. Storage unit → HDFS



If 1 data node crashes?

Won't loss the data.

HDFS makes copies of data, stores it across multiple systems.



When Block A is created, it is replicated with a replication factor of 3 and store on different DataNodes.

Data is not lost at any cost; even if one DataNode crashes, making HDFS **fault-tolerant**.

Cost: Zero licensing and support costs

Reliability: HDFS copies the data multiple times

Speed: Hadoop clusters read or write more than one terabyte of data in a second

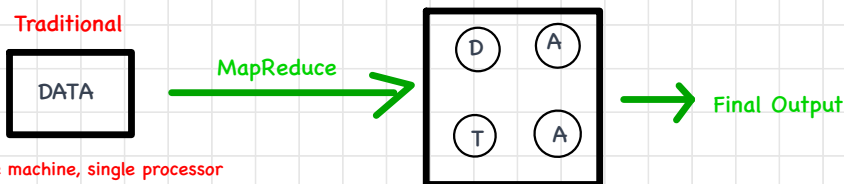
Characteristics of HDFS:

- Fault Tolerant
- Scalable
- Rack-Aware
- Support for heterogeneous clusters
- Built for large datasets

## 2. MapReduce → Data Processing

Traditional data processing method: process on single machine having single processor.

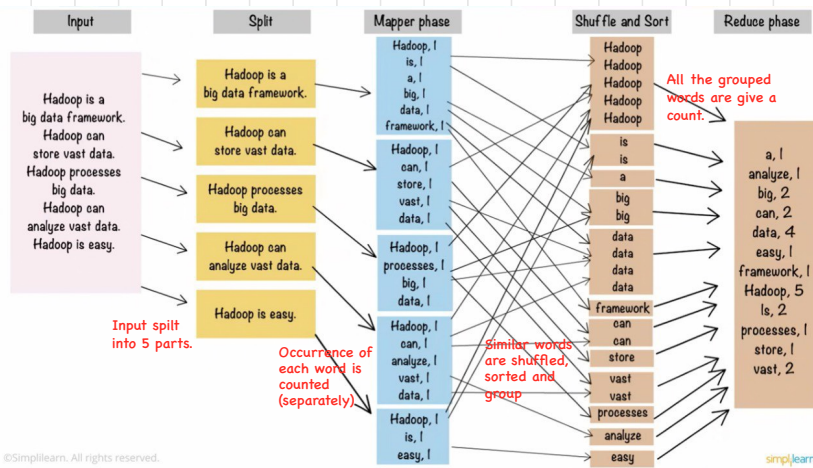
> consumed time and inefficient when processing large volume variety of data.



Single machine, single processor

MapReduce split data into parts, and processes each of them separately on different DataNodes.

Individual results are then aggregated to give the final output.



©Simplilearn. All rights reserved.

simplilearn

### Map Execution Phases:

Map Phase  
Partition Phase  
Shuffle Phase  
Sort Phase  
Reduce Phase

**Map process:** An initial ingestion and transformation step where initial input records are processed in parallel

**Reduce process:** An aggregation or summarisation step in which all associated records are processed together

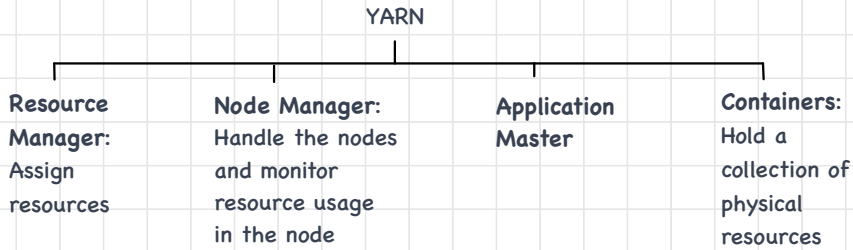
**Node Manager:** Keep tracks of individual map tasks and can run in parallel

**Application master:** Keeps track of a MapReduce job

### Characteristics of MapReduce:

- Handle large-scale data
- Works well on write once read many of WORM data
- Allows parallelism
- Operations are performed by the same processor
- Provides operations near the data
- Leverages commodity hardware and storage
- Attends to split and move data

### 3. YARN → Cluster Resource Management



#### Resource Manager (RM):

- 1 per cluster, is the master server.
- Knows the location of the DataNode and how many resources they have
- Run several services, most important is the resource scheduler that decides how to assign resources.

#### Application Master (AM):

- Framework specific process, negotiates resources of a single application.
- Each AM request resources from RM, then works with containers provide by NM.

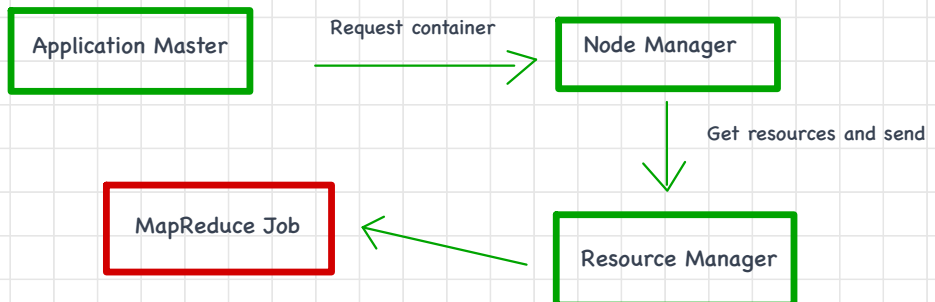
#### Node Manager (NM):

- Can be many in 1 cluster.
- Slaves of the infrastructure.
- When it starts, it announces itself to be the RM and offers resources to the cluster.

#### Container:

- A fraction of the NM capacity, used by the client for running the program.
- Takes instructions from the RM and reports and handles containers on a single nodes

YARN processes job requests and manages cluster resources.



# Data Access

The ability to access and process data stored in a Hadoop cluster. There are several tools and frameworks available for data access in Hadoop, including Hive, Pig, Spark, and HBase.

**Hive** is a data warehousing and SQL-like query language for Hadoop. It provides a simple and familiar interface for analysts and data scientists to perform ad-hoc querying and analysis on large data sets stored in Hadoop. Hive translates SQL-like queries into MapReduce jobs, allowing for the efficient processing of large data sets in Hadoop. Hive supports a wide variety of data formats and allows for the creation of tables, views, and partitions for efficient data organization.

**Pig** is a high-level scripting language for Hadoop that allows for the creation of data processing workflows. It provides a set of operators for data manipulation and transformation, allowing users to easily express complex data processing tasks. Pig translates these scripts into MapReduce jobs, allowing for the efficient processing of large data sets in Hadoop. Pig also supports the creation of custom UDFs (User Defined Functions) for more advanced data processing tasks.

**Spark** is a distributed data processing framework for Hadoop that provides in-memory processing capabilities. It allows for the efficient processing of large data sets in real-time, making it ideal for use cases such as machine learning and real-time data analytics. Spark supports a wide variety of programming languages, including Java, Python, and Scala, and provides APIs for data processing, streaming, and machine learning.

**HBase** is a NoSQL database for Hadoop that provides real-time access to large datasets. It provides a column-family-based data model that allows for the storage of unstructured and semi-structured data in Hadoop. HBase supports fast and efficient data retrieval and allows for the creation of tables and row keys for efficient data organization.

# Chapter 3

## Machine Translation:

Machine translation is the use of computer algorithms to automatically translate text from one language to another. It involves processing input text in one language, applying a translation model to generate a representation of the text in another language, and then generating output text in that language. Machine translation is often used in applications such as web-based translation services, language learning software, and global e-commerce platforms.

### How Machine Translation Works:

Machine translation involves using algorithms to automatically translate text from one language to another. The process typically involves several steps:

1. **Text Preprocessing:** The input text is first processed to remove any unnecessary elements, such as formatting tags or markup language.
2. **Language Analysis:** The input text is then analyzed to determine its language, and to identify the grammatical structures and linguistic features of the source language.
3. **Translation:** The input text is translated into the target language, using a variety of techniques such as statistical models, rule-based approaches, and neural machine translation.
4. **Post-Processing:** The translated text is then post-processed to improve its accuracy and readability, such as by adjusting the grammar, syntax, or word choice.

# Features of Identifiers:

**Completeness:** An identifier should be complete, meaning it should capture all relevant information about the data object it is identifying. This ensures that the data object can be easily found and accessed when needed.

**Uniqueness:** An identifier should be unique, meaning it should be distinct from all other identifiers in the system. This ensures that the data object can be uniquely identified and that there are no conflicts or errors in the data.

**Exclusivity:** An identifier should be exclusive, meaning it should not be reused for other data objects. This ensures that the data object can be accurately identified over time and that there are no ambiguities or errors in the data.

**Authenticity:** An identifier should be authentic, meaning it should accurately represent the data object it is identifying. This ensures that the data can be trusted and that there are no errors or inaccuracies in the data.

**Aggregation:** An identifier should be capable of aggregation, meaning it should allow for the grouping of related data objects together. This ensures that the data can be easily analyzed and understood at different levels of granularity.

**Permanence:** An identifier should be permanent, meaning it should persist over time and be able to be used to identify the data object consistently over time. This ensures that the data can be reliably accessed and analyzed over time.

**Reconciliation:** An identifier should support reconciliation, meaning it should be possible to match different identifiers for the same data object across different systems. This ensures that data from different sources can be accurately combined and analyzed together.

**Immutability:** An identifier should be immutable, meaning it should not be able to be changed once assigned to a data object. This ensures that the data object can be accurately identified over time and that there are no ambiguities or errors in the data.

**Security:** An identifier should be secure, meaning it should be protected from unauthorized access or modification. This ensures the integrity and confidentiality of the data.

**Documentation and Quality:** An identifier should have clear and accurate documentation to ensure that it is used correctly and consistently across the system. It should also be of high quality, meaning that it should be accurate, relevant, and up-to-date, and meet the requirements of the system and its users.



### Pros of Machine Translation:

- **Speed:** Machine translation can process large volumes of text quickly and efficiently, making it useful for applications such as online translation services or multilingual chatbots.
- **Cost-Effective:** Machine translation can be more cost-effective than hiring professional translators, particularly for large-scale projects or languages with a smaller pool of translators.
- **Accessibility:** Machine translation can make information more accessible to people who speak different languages, which can help to break down language barriers and promote cross-cultural communication.

### Cons of Machine Translation:

- **Accuracy:** Machine translation is not always accurate, particularly for languages with complex grammatical structures or idiomatic expressions. This can result in errors, misunderstandings, and even offense in some cases.
- **Contextual Understanding:** Machine translation may struggle to understand the context of the text, particularly in cases where there is ambiguity or multiple possible meanings. This can lead to mistranslations or incorrect translations.
- **Limited Vocabulary:** Machine translation may struggle with rare or specialized vocabulary, which can result in inaccurate translations or missing nuances in the text.

# Autocoding:

Autocoding is the process of automatically assigning codes to text data based on pre-defined rules or nomenclature. This process can be used to tag data with relevant metadata, classify documents into categories, or extract important information from unstructured text. Autocoding is often used in applications such as content management systems, digital libraries, and online archives.

Nomenclature refers to a system of naming or categorizing things according to a set of predefined rules or conventions. In the context of data analysis, a nomenclature may consist of a list of terms or categories that are used to classify or tag different types of data. For example, a nomenclature for a medical study might include categories such as patient demographics, disease diagnosis, and treatment outcomes, which can then be used to organize and analyze the data in a standardized way.

Key purposes of nomenclatures. Here's a brief explanation of each:

- **Enhance interoperability and integration:** By using a standardized nomenclature, different systems and applications can more easily exchange and share data, since they are all using the same terminology. This can help to reduce errors and improve efficiency, since there is no need to translate or reconcile different naming conventions.
- **Allow synonymous terms to be retrieved:** A nomenclature can help to address the issue of synonymy, where different terms may be used to refer to the same concept or entity. By mapping different synonyms to a common term or concept, it becomes easier to search and retrieve relevant information regardless of the specific terminology used.
- **Support comprehensive analyses of textual data:** By tagging or categorizing textual data using a nomenclature, it becomes possible to perform more detailed and comprehensive analyses. For example, by using a medical nomenclature to categorize patient data, it becomes possible to analyze trends in disease incidence, treatment effectiveness, and other factors across different patient populations.

### Autocoding algorithm steps:

1. **Preprocessing:** The input data is first cleaned and processed to remove any irrelevant content or formatting, and to ensure that the data is in a standardized format.
2. **Tokenization:** The data is then split into smaller units, such as words or phrases, known as tokens. This allows the algorithm to analyze the data at a more granular level.
3. **Rule Generation:** A set of rules or algorithms is created based on the desired coding scheme or classification system. This can involve using machine learning techniques to train the algorithm on a sample of the data, or creating a set of heuristics or decision trees based on expert knowledge.
4. **Data Classification:** The input data is then automatically assigned codes or categories based on the generated rules or algorithms. This can involve using techniques such as pattern matching, keyword analysis, or statistical inference to determine the appropriate code or category for each piece of data.
5. **Quality Control:** The assigned codes or categories are then reviewed and refined to ensure accuracy and consistency. This can involve manual review by human coders, or the use of automated checks to detect and correct errors or inconsistencies.
6. **Iteration:** The algorithm is then refined and updated based on the results of the quality control process, and the process is repeated until a satisfactory level of accuracy and consistency is achieved.

### Pros of Autocoding:

- **Speed:** Autocoding can process large volumes of data quickly and efficiently, making it useful for applications such as market research, social media analysis, and content analysis.
- **Consistency:** Autocoding can provide consistent and standardized coding across multiple sources of data, reducing the risk of human error or subjectivity.
- **Scalability:** Autocoding can be easily scaled up or down depending on the size and complexity of the data set, making it a flexible solution for data analysis.

### Cons of Autocoding:

- **Accuracy:** Autocoding can be prone to errors or inaccuracies, particularly when dealing with complex or ambiguous data. This can result in incorrect or inconsistent coding, which can affect the quality of the analysis.
- **Limited Scope:** Autocoding may not be able to capture the full complexity or nuance of the data, particularly in cases where human interpretation or judgement is required.
- **Dependency on Rules:** Autocoding is highly dependent on the quality and accuracy of the generated rules or algorithms, which can be challenging to create and maintain.

# Indexing:

Indexing is the process of creating an index or catalog of the contents of a collection of documents or data. This allows users to search for specific words or phrases and retrieve relevant documents or data quickly and efficiently. Indexing is often used in applications such as search engines, digital libraries, and e-commerce platforms.

Here's how indexing works:

1. **Document analysis:** The first step in indexing is to analyze the content of the document to identify key terms or concepts. This might involve using natural language processing techniques to identify named entities, relationships between words, and other relevant information.
2. **Term extraction:** Once the key terms have been identified, the next step is to extract those terms and create a list of index terms. This might involve removing stop words (common words such as "the" and "and" that don't add much meaning), stemming or lemmatizing words (reducing words to their base form), and eliminating duplicates.
3. **Index creation:** With the list of index terms in hand, the next step is to create an index that maps each term to the location of the document in the corpus. This might involve creating an inverted index, where each term is mapped to a list of documents that contain that term.
4. **Search and retrieval:** With the index in place, users can search for documents that contain specific terms or concepts. The search engine uses the index to quickly identify relevant documents, and returns them to the user in a ranked list.

## Pros of indexing:

- **Faster search and retrieval:** By creating an index of key terms, users can quickly find documents that contain specific information.
- **Improved accuracy:** Since the index is created based on the content of the document, it is more accurate than searching based on metadata alone.
- **Scalability:** Indexing can scale to handle very large document collections, making it useful for big data applications.

## Cons of indexing:

- **Index creation can be time-consuming:** Creating an index for a large corpus of documents can take a significant amount of time and resources.
- **Maintenance overhead:** The index must be kept up-to-date as new documents are added to the corpus, which can require additional maintenance overhead.
- **Difficulty with synonyms:** Since indexing relies on exact term matches, it can be challenging to handle synonyms or related terms that may be used interchangeably.

## Why not just use the "find" function?

While the "find" function can be useful for locating specific words or phrases within a document, it can become inefficient and time-consuming when dealing with large amounts of data. Indexing is a more efficient and scalable method for organizing and accessing large amounts of data.

When data is indexed, a separate file is created that contains a list of every unique word in the document or dataset, along with a pointer to every occurrence of that word within the dataset. This allows for much faster and more efficient searching and retrieval of data, as the search can be limited to only those documents or records that contain the specific indexed term or phrase.

In addition, indexing can also enable more complex search functions, such as searching for multiple terms within a certain proximity to each other, or searching for terms that are similar in meaning to a specified keyword.

However, indexing does come with some potential drawbacks. Building and maintaining an index can require significant computational resources and storage space, especially for very large datasets. Additionally, updating an index can be a time-consuming process, and the index may need to be rebuilt periodically to ensure accuracy and relevance.

# Term Extraction:

Term extraction is the process of automatically identifying and extracting key terms or concepts from a collection of text data. This can be done using natural language processing techniques such as part-of-speech tagging, named entity recognition, and text mining algorithms. Term extraction can be used to identify trends, patterns, and insights from large volumes of unstructured data, and is often used in applications such as business intelligence, market research, and social media analysis.

## Term extraction process:

1. **Preprocessing:** The first step is to preprocess the text by removing any unwanted characters or symbols, converting all the text to lowercase, and removing stop words (common words like "the," "and," "a," etc.) that do not provide much meaning.
2. **Tokenization:** The next step is to break down the text into individual words or phrases, known as tokens. This can be done using various techniques such as whitespace tokenization or n-gram tokenization.
3. **Part-of-speech (POS) tagging:** Once the text has been tokenized, each word or phrase is assigned a part-of-speech tag based on its grammatical function within the sentence (e.g. noun, verb, adjective, etc.). This helps to identify and extract only the relevant terms from the text.
4. **Term extraction:** The actual term extraction process involves identifying and extracting the most important and relevant terms from the text based on certain criteria, such as frequency of occurrence, relevance to the topic or domain, and importance in context. This can be done using various techniques such as statistical methods, machine learning algorithms, or rule-based systems.
5. **Evaluation:** Finally, the extracted terms are evaluated to ensure that they are relevant and meaningful in the context of the document or dataset. This may involve manual review by domain experts or automatic evaluation metrics such as precision and recall.

# Chapter 4

**Identification:** This refers to the process of identifying an individual or entity by linking their personal information to their identity, such as through a name, address, or other unique identifier.

Identification can be done directly, through the use of a unique identifier, or indirectly, by combining various pieces of personal information to identify an individual.

**De-identification:** This refers to the process of removing or obscuring personal information from a dataset in order to protect the privacy of the individuals whose data is included. This can involve removing direct identifiers such as names or addresses, or using techniques such as data masking, tokenization, or generalization to obscure identifying information.

**Re-identification:** This refers to the process of using information from a de-identified dataset to identify individuals or re-link their personal information to their identity. Re-identification can occur through various means such as data linkage, data mining, or inference based on other available information.

The purpose of de-identification is to protect the privacy of individuals and prevent their personal information from being used in ways that could be harmful or discriminatory. However, there is a risk of re-identification, particularly with the increasing availability of large and diverse datasets and advanced data analytics techniques. Therefore, it is important to consider the potential risks of re-identification and to implement appropriate safeguards to protect individuals' privacy.

# Features of Identifiers:

**Completeness:** An identifier should be complete, meaning it should capture all relevant information about the data object it is identifying. This ensures that the data object can be easily found and accessed when needed.

**Uniqueness:** An identifier should be unique, meaning it should be distinct from all other identifiers in the system. This ensures that the data object can be uniquely identified and that there are no conflicts or errors in the data.

**Exclusivity:** An identifier should be exclusive, meaning it should not be reused for other data objects. This ensures that the data object can be accurately identified over time and that there are no ambiguities or errors in the data.

**Authenticity:** An identifier should be authentic, meaning it should accurately represent the data object it is identifying. This ensures that the data can be trusted and that there are no errors or inaccuracies in the data.

**Aggregation:** An identifier should be capable of aggregation, meaning it should allow for the grouping of related data objects together. This ensures that the data can be easily analyzed and understood at different levels of granularity.

**Permanence:** An identifier should be permanent, meaning it should persist over time and be able to be used to identify the data object consistently over time. This ensures that the data can be reliably accessed and analyzed over time.

**Reconciliation:** An identifier should support reconciliation, meaning it should be possible to match different identifiers for the same data object across different systems. This ensures that data from different sources can be accurately combined and analyzed together.

**Immutability:** An identifier should be immutable, meaning it should not be able to be changed once assigned to a data object. This ensures that the data object can be accurately identified over time and that there are no ambiguities or errors in the data.

**Security:** An identifier should be secure, meaning it should be protected from unauthorized access or modification. This ensures the integrity and confidentiality of the data.

**Documentation and Quality:** An identifier should have clear and accurate documentation to ensure that it is used correctly and consistently across the system. It should also be of high quality, meaning that it should be accurate, relevant, and up-to-date, and meet the requirements of the system and its users.



# One-way hash

One-way hash functions are mathematical functions that take in input data of any size and produce an output, also known as a hash or message digest, of fixed size. The output is unique to the input data, and any small change in the input data will result in a vastly different output. This makes it difficult to reverse-engineer the input data from the output, hence the name "one-way" hash function.

**Input data:** The data to be hashed is input into the hash function. This data can be of any size or format, such as a password or message.

**Hash function:** The hash function takes the input data and performs a series of mathematical calculations to generate a unique output of fixed size. Some commonly used one-way hash functions are MD5, SHA-1, and SHA-256.

**Output:** The output of the hash function, also known as the hash or message digest, is a fixed-size string of characters that is unique to the input data. The output is typically represented in hexadecimal format, which is a combination of numbers and letters.

**Verification:** The hash can be used to verify the integrity of the input data. For example, if two sets of data produce the same hash, then the data is likely to be the same. This property can be used to compare passwords without storing the actual password, as only the hash is stored.

### Pros of one-way hash:

**Security:** One-way hash algorithms are designed to be **very difficult to reverse or decrypt**, which makes them a good choice for storing passwords or sensitive information.

**Efficiency:** One-way hash functions are computationally efficient, meaning they can be **performed quickly on large amounts of data**.

**Verification:** **Hash functions can be used to verify the integrity of data** by comparing the hash of the original data to the hash of the modified data. If the hashes are different, it indicates that the data has been tampered with.

### Cons of one-way hash:

**No reversal:** One of the biggest limitations of one-way hash algorithms is that they **cannot be reversed**. Once data is hashed, it **cannot be un-hashed or decrypted**, which can be problematic if the original data needs to be recovered.

**Vulnerability to attacks:** Although one-way hash functions are designed to be secure, they can still be **vulnerable to attacks such as brute force or rainbow table attacks**.

**Collision:** A collision occurs when two different pieces of data produce the same hash value. While the **likelihood of a collision is very low with modern hash functions**, it is still **a possibility** and can cause problems in some applications.

## Deidentification serves two purposes:

1. To protect the confidentiality and the privacy of the individual (when the data concerns a particular human subject)
2. To remove information that might bias the experiment (e.g., to blind the experimentalist to patient identities)

Because confidentiality and privacy concerns always apply to human subject data and because issues of experimental bias always apply when analyzing data, it would seem imperative that identification should be an irreversible process (i.e., the names of the subjects and samples should be held a secret, forever).

The process of de-identification should be irreversible, meaning that the original personal information should not be recoverable from the de-identified data. This is important to protect the privacy and confidentiality of individuals whose data is being analyzed. However, there is always a risk of re-identification, especially if the dataset is combined with other sources of data.

# Re-identification

Re-identification refers to the process of **matching de-identified data with identifying information to reveal the identity of an individual**. In some cases, re-identification is necessary for research or public health purposes, but it also poses a significant risk to the privacy and confidentiality of individuals.

Re-identification can occur through various means, including combining de-identified data with other data sets that contain identifying information, using advanced data mining techniques, or exploiting vulnerabilities in the de-identification process.

To **minimize the risk of re-identification**, researchers and organizations can implement various measures, such as **restricting access to de-identified data**, using statistical disclosure control techniques, and regularly reviewing and updating de-identification methods to account for emerging threats.

It is important to balance the need for re-identification with the risks to privacy and confidentiality. Re-identification should only be conducted when it is necessary for research or public health purposes and with appropriate safeguards in place to protect individuals' privacy and confidentiality.

# Ontology

An ontology is a formal and explicit specification of a shared conceptualization, which includes the categories, concepts, and relationships within a particular domain. It provides a common vocabulary and structure for representing knowledge and information, making it easier to share and reuse information across different systems and applications. Ontologies can be used in a wide range of applications, such as natural language processing, knowledge management, and semantic web technologies.

## Semantics

Semantics refers to the meaning of words and the relationships between them. It is concerned with the interpretation of words and phrases in a particular context, and it plays a crucial role in natural language processing and machine learning applications. In the context of the semantic web, semantics is used to represent and link data in a meaningful way, enabling machines to understand the relationships between different pieces of information.

The relationship between ontologies and semantics is that ontologies provide a structured framework for representing knowledge and information, while semantics provides the meaning and context needed to interpret that information. By using ontologies and semantics together, it is possible to create more intelligent and effective information systems that can reason and infer meaning from the data they process.

## Classification

Classification is a process of categorizing information into pre-defined classes or categories. The categories are typically hierarchical, with broader categories at the top and more specific categories at the bottom. Classification is used to make information easier to find and retrieve, and it can be used in a variety of fields such as library science, biology, and computer science.

# Ontologies:

Constructions that permit an object to be a direct subclass of more than one class.

Unrestrained classifications: Ontologies are predicated on the belief that a single object or class of objects might have multiple different fundamental identities and that these different identities will often place one class of objects directly under more than one superclass.

- Ontologies do not have the same constraints as traditional classifications, and can allow for objects to have multiple identities and be classified under multiple superclasses.

Data analysts sometimes prefer ontologies over classifications because they permit the analyst to find relationships among classes of objects that would have been impossible to find under a classification.

- Ontologies can help reveal relationships and connections between classes of objects that may not be immediately apparent in a traditional classification system.

Data object can be an instance of many different kinds of classes; thus, the class does not define the essence of the object as it does in a classification.

Classifications were created and implemented at a time when scientists did not have powerful computers that were capable of handling the complexities of ontologies.

Ontology model is overly complex:

1. Can be **difficult to understand and use**, particularly for non-experts. This can make it challenging to share and communicate knowledge effectively.
2. Can be **challenging to maintain and update over time**. As new knowledge is gained, the ontology may need to be revised and expanded, which can be difficult and time-consuming if the ontology is overly complex.
3. May also suffer from **issues of inconsistency or ambiguity**. If the ontology is too complex, there may be overlapping or contradictory definitions, which can lead to confusion and errors.

Classification model is overly simple:

1. May **not capture the nuances and complexities of the information** being categorized. This can lead to inaccuracies and errors in classification, and make it difficult to retrieve information.
2. May **not be flexible enough to adapt to new information or changing needs**. This can limit its usefulness over time, particularly in dynamic fields where new information is constantly being generated.
3. May also suffer from **issues of inconsistency or ambiguity**, particularly if the categories are too broad or overlapping. This can lead to confusion and errors, and make it difficult to retrieve information accurately.

## Common Pitfalls:

### 1. Don't build transitive classes: Puppy->dog?

Defining "Puppy" as a subclass of "Dog", can create inconsistencies in the ontology. This is because the superclass "Dog" may have additional subclasses that do not share the same characteristics as "Puppy". It is better to define "Puppy" as a subclass of "Young Dog" or "Juvenile Dog".

### 2. Don't build miscellaneous classes.

Building miscellaneous classes can create confusion and ambiguity in the ontology. It is important to ensure that each class has a clear and distinct definition and purpose within the ontology.

### 3. Don't invent classes and properties if they have already been invented.

Reusing existing classes and properties whenever possible is important to maintain consistency and avoid duplication in the ontology. This can also facilitate integration with other ontologies and data sources.

### 4. Do not confuse properties with your classes: Is a leg a subclass of the human body?

Confusing properties with classes can lead to incorrect modeling and interpretation of the ontology. In the example given, "Leg" is a property of the class "Human Body", rather than a subclass.



# Chapter 5

## Database Management System (DBMS)

- collection of interrelated data
- set of programs to access the data
- convenient and efficient to use
- Can be very large
- Touch all aspects of our lives

### Drawbacks using file systems to store data:

**Data redundancy and inconsistency:** File systems often lead to data duplication, as multiple copies of the same data are created and stored in different locations. This can lead to inconsistencies and errors, as changes made to one copy may not be reflected in others.

**Difficulty in accessing data:** File systems can make it difficult to access data, particularly when dealing with large or complex data sets. It can be challenging to locate specific data files or to extract relevant information from them.

**Data isolation:** File systems can also result in data isolation, where data is stored in separate files or directories that are not easily linked together. This can make it challenging to analyze data across different sources or to identify patterns and relationships in the data.

**Integrity problems:** File systems can also lead to integrity problems, as data may become corrupted or lost due to issues such as hardware failures or power outages. It can be challenging to recover lost or corrupted data, and the risk of data loss increases as the volume of data stored on file systems grows.

**Atomicity of updates:** Atomicity refers to the ability to make a set of updates to a database or system as a single, indivisible operation. File systems do not provide this level of atomicity, which can lead to partial updates and inconsistent data.

**Concurrent access by multiple users:** File systems do not provide mechanisms for managing concurrent access by multiple users, which can lead to conflicts and data corruption.

**Security problems:** File systems are typically not designed with security in mind, which can make them vulnerable to unauthorized access, data theft, and other security issues.

## Data Models

- Relational model
- Entity-relationship model – database design
- Object-based data model: Object-oriented and object-relational
- Semi structured data model: XML
- Older model: Network model, Hierarchical model

### Keys:

Primary key: one of the candidate keys

Foreign key: Value in one relation appear in another

### Updates to table:

Insert

Delete

Drop Table

Alter

### Modification in database:

Deletion

Insertion

Updating

## Advantages of Traditional File Access:

**Data Security:** Traditional file systems are more secure than electronic systems because the files are kept physically in a specific location, making it difficult for unauthorized personnel to access them.

**Complexity:** Traditional filing systems are less complex than electronic systems, which can make it easier for untrained people to access and manipulate data. It is easy for anyone to look through alphabetized filing cabinets to find a file, whereas locating and manipulating an electronic database requires technical training, and user error can result in unintended alterations or data loss.

## Disadvantages of Traditional File Access:

**Access Time:** It can take a long time to locate a few files in a large paper filing system. Searching through physical files is a tedious process that can take hours, whereas electronic databases allow for almost instantaneous access to information.

**Editing and Communication:** Traditional file systems are cumbersome in that they do not allow users to easily edit files or send information to others. Paper files often cannot be edited directly, forcing users to make new copies to update old files. Electronic databases, on the other hand, allow for easy editing and communication of data.

**Order of Data:** Traditional filing systems are prone to human error, which can lead to data getting out of order. If someone accidentally puts a file in the wrong place, or takes a file out of a cabinet and forgets to put it back, it can lead to lost data or the creation of additional copies of files. Electronic databases are more reliable in maintaining the order of data.

# Electronic database

## Advantages:

Clearly defined fields and relationships between different fields make it easier to organize and search for specific data.

Schema-on-write ensures that data is validated before being written to disk, reducing the risk of errors and data corruption.

Structured formats make it easier to analyze and extract insights from the data, as well as integrate it with other systems and applications.

Loading data into memory can improve processing speed and reduce access times.

## Disadvantages:

The upfront effort required to design and structure the data can be time-consuming and expensive, delaying the time before business value can be realized from the data.

The architecture of loading data from disk to memory can be inefficient when processing large volumes of data, leading to slow performance and increased costs.

The fixed structure of electronic databases can make it difficult to accommodate changes in the data or adapt to new use cases.

The size of the data can be very large, requiring significant storage and processing resources. This can lead to high costs for hardware and infrastructure.

# Chapter 6

## Distributed computing

A type of computing system in which multiple computers work together to achieve a common goal. In a distributed computing system, individual computers, also known as nodes, are connected to each other via a network and communicate with each other to coordinate their activities. This enables them to work together to solve problems that would be difficult or impossible to solve with a single computer.

### Characteristics of distributed computing

**Decentralization:** Distributed computing systems are decentralized, meaning there is no single point of control or failure. Instead, each computer in the network is responsible for a portion of the overall workload.

**Scalability:** Distributed computing systems can be scaled up or down depending on the size of the problem or task. Additional computers can be added to the network to increase processing power, and removed when they are no longer needed.

**Fault tolerance:** Because there is no single point of control or failure, distributed computing systems are typically more fault-tolerant than centralized systems. If one computer in the network fails or goes offline, the rest of the network can continue to operate.

**Heterogeneity:** Distributed computing systems can be composed of computers with different hardware, software, and operating systems. This allows organizations to leverage existing infrastructure and hardware, and to use the best tools for each part of the task.

**Interoperability:** Distributed computing systems require standards and protocols to ensure that different computers can communicate with each other and share data. Common standards and protocols include TCP/IP, HTTP, and XML.

**Security:** Distributed computing systems are vulnerable to security threats such as data theft, tampering, and denial of service attacks. Security measures such as encryption, authentication, and access control are needed to protect the network and the data it contains.

**Performance:** The performance of a distributed computing system depends on factors such as network bandwidth, latency, and the number and speed of the computers in the network. To optimize performance, distributed computing systems often use load balancing, caching, and other techniques.

## 2 Types of distributed systems:

1. Computers are physically close together (connect by LAN)
2. Computers are geographically distant (connect by WAN)

### Advantages of distributed computing:

**Increased speed:** By dividing a task into smaller pieces and distributing it across multiple computers, the overall speed of processing can be increased.

**Scalability:** Distributed computing allows for easy scalability by adding or removing nodes as needed.

**Fault tolerance:** If one node fails, the overall system can continue to function as other nodes take over the failed node's workload.

**Cost-effective:** By using existing hardware, distributed computing can be more cost-effective than purchasing new hardware.

### Disadvantages to distributed computing:

**Network dependency:** Distributed computing systems rely heavily on the network connecting the nodes. If the network is slow or unreliable, the performance of the system may suffer.

**Complexity:** Distributed computing systems can be complex to set up and maintain, requiring specialized knowledge and skills.

**Security risks:** With multiple nodes connected to a network, there is a greater risk of security breaches, and it can be more difficult to secure the system as a whole.

**Data consistency:** With multiple nodes processing data, maintaining data consistency can be a challenge.

## Distributed

Many processing steps but divided into several computers/nodes

## Parallel

Many processing steps, each step completed at specified time  
Supercomputer-suitable for high performance computing

## Grid

Manage/pooling hundreds/thousand computer system which individual are more limited in memory and processing power

More secured  
Loosely-coupled  
Location-specified: organisation/firm  
Efficiently utilise of a pool of heterogeneous systems, create large pools of resources

## Cloud

Remote location/virtual facility  
Scalable & flexible (over IT-related capabilities)  
Paid services  
Minimal flexible (over functions of software and hardware)

## Cluster

More heterogeneous of computers, located in different geographic locations  
Communication over network, can have latency

Tightly-coupled computers/systems/nodes work together but impression of single system/resource  
High availability - use of implicit redundancy to the system