Data Mining

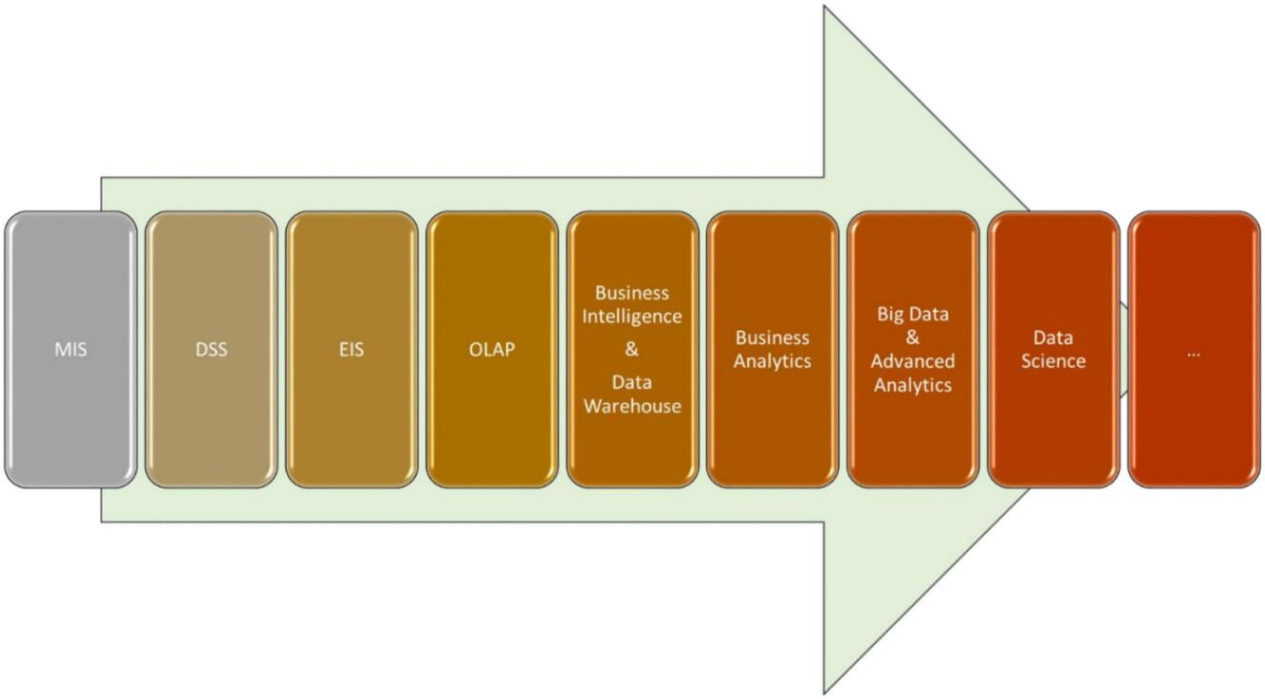
01 – Introduction

Data exists independently from data mining and machine learning but if we want to extract **insights** that are **actionable** from data, then we need those techniques, especially when it comes to **big data**.

We start by introducing a few definitions:

- **Data**: a collection of raw (or partly manipulated) value elements.
- **Information**: the result of collecting and organizing the data, typically showing the relationships between data items, giving context and meaning to the data (information = data + meaning).
- **Knowledge**: understanding information based on recognizing patterns.

Different tools and techniques allow to gather increasing insights on data by adding structure and interpreting it, as shown in the picture below:



We might now ask "where does data come from"? To answer this question, we must start by defining what is a **business process**: a set of activities that, once completed, will achieve an **organizational goal** (e.g. delivering a product to the consumer). When an **event** in the real world **changes the sate** of the enterprise, a **transaction** is executed to reflect the corresponding change in the **database** (by transaction we mean a **business event that generates or modifies data stored in an information system**).

We now start introducing the concepts that we previously saw depicted in the arrow. First, we will take a look at **OLTP (On-Line Transaction Processing)**: a class of software programs

capable of supporting transaction-oriented applications and data storage, designed to record the daily transactions necessary to run the business with goals of availability, speed, concurrency (multiple operators accessing certain data at the same time) and recoverability (we must be able to recover from crashes, losing as little as possible).

Many companies have **ERP (Enterprise Resource Planning)** systems built on top of OLTP. They are **integrated systems** that:

- **Can manage all the business processes of all the departments of a company within a single software product** (e.g. financial management, supply chain management, customer relationship management, human resource management, manufacturing resource planning, etc.).
- **Provide a common database to support all the applications**.
- **Operate in (near-)real time**.
- **Have a consistent look and feel across modules**.

MIS (Management Information Systems) are less complicated than ERPs, as they are **standardized and fixed reporting systems** built on top of existing OLTP solutions. They are built to **support structured and operational decision making** (the least difficult type, in which the requirements can be described in detail before the decision is made) and to be used by both managers and employees to generate various performance indicators.

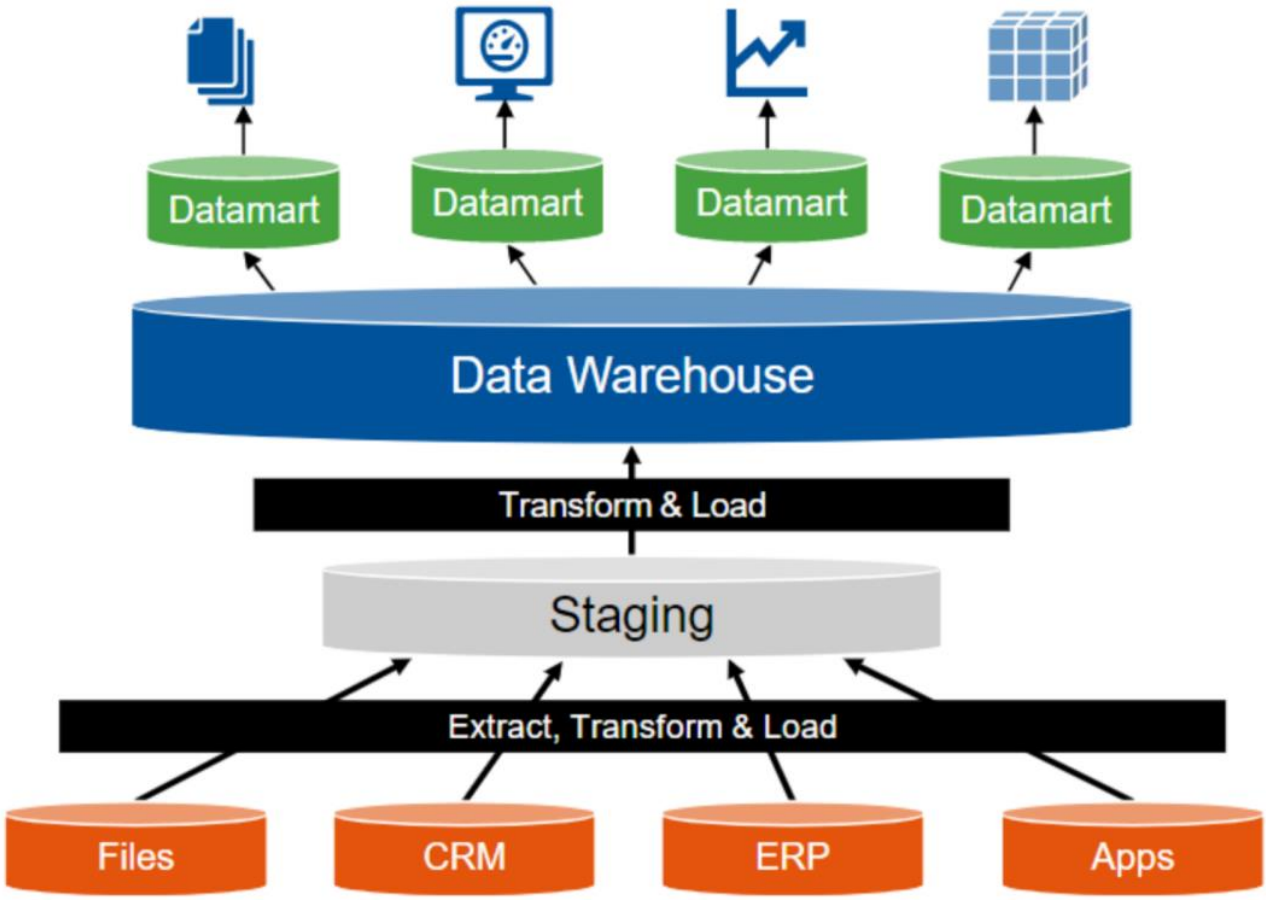
DSS (Decision Support Systems) are **analytical systems**, as in they analyze data at a higher level (e.g. by focusing on an entire department, rather than on a single transaction), and are meant to **provide support for complex and unstructured decisions** (ones that cannot be handled by MIS). They attempt to combine the use of models or analytic techniques with traditional data access and retrieval functions.

EIS (Executive Information Systems) support the executive level of management, allowing to **formulate high level strategic decisions that strongly impact the direction of the organization**. Given their target users, they need to have user friendly interfaces and the ability to extract summary data from both internal and external systems.

BI (Business Intelligence) systems are very important, so we will start by giving two definitions, one by Gartner and one by Forrester Research:

- "Applications, infrastructure, tools and **best practices** that enable access to and analysis of information to improve and optimize decisions and performance" (Gartner).
- "A set of methodologies, processes, architectures and technologies that transform raw data into meaningful and useful information used to enable more effective strategic, tactical and operational insights and decision-making" (Forrester Research).

An example of **business intelligence architecture** is shown in the picture below:



The idea behind business intelligence is that if we want to have a deep understanding of how our organization works and to guide it towards its intended results, we need to have an overall view of our whole company and the single applications are not enough. Business Intelligence, in fact, puts together the data coming from the various operational data sources to highlight the inter-relationships between departments. A central piece of this puzzle is called **data warehouse** and is the result of extracting data from their sources, putting them in an intermediate area **(staging)**, integrating it and giving it a structure. Ralph Kimball defines the data warehouse as "a copy of transaction data specifically structured for query and analysis".

The **OLAP (On-Line Analytical Processing)** concept is strictly related to the one of data warehouse: it is meant to allow an **interactive analysis of multidimensional data from multiple perspectives**. To clarify what "analyzing something from multiple perspectives" means, we can think of a manager from a company that sells goods: they might want to look at the sales trends from the perspective of time, or along a geographical dimension or based on the type of goods that are sold. Roughly speaking, doing OLAP means to **use extensively group by and summary functions with data structures designed to make for easier operations such as selections, projections and column exchanges** (refer to **data cube** later on).

It is important to note how OLTP-based management systems (like MIS), despite having the ERP module to make the various systems interact, are still fundamentally independent from

each other, while in OLAP-based analytical systems, the ETL (Extract, Transform, Load) process guarantees data consistency.

To clear up what **structured** and **unstructured decisions** are, we can refer to the following table, containing a few examples:

Structured		***Unstructured***	
Description	***Example***	***Description***	***Example***
Made under an established situation	Hiring a new employee	Made under an emergent situation	Fire breakout
Programmed	Start the monthly payment of salaries	Unplanned	Opportunity for financial investment
Fully understood	When a bank customer makes huge fund movements ask him the reason	Unclear or uncertain	Necessary to acquire information to understand which operation is to be performed
Routine task	Hiring new personnel in a given sector	Suddeen 0ne-shot situation	Dealing with a labor strike
Specified process	Manufacturing something	General processes	Managing security for IT equipment
Well defined methodology	Possible withdraw of funds from international accounts according to currency rates	Decisions relying on knowledge and/or expertise and on analysis of information	What new market segment could be targeted

To enable these types of decisions in a data-driven way, we usually refer to **analytics** for **structured decisions** and to **data mining** for **unstructured decisions** (through data mining we could also provide insights for structured decisions, of course).

While **organizations previously struggled to collect data to improve the decision process**, we are now in the **big data era**, in which the issue becomes how to extract value from such massive amounts of data, as we need to:

- **Process the growing VOLUME of data in a cost-efficient way**.
- **Respond to the increasing VELOCITY of data generation**.
- **Collectively analyze the broadening VARIETY of data**.

In addition to organizations, data is now being generated by machines (e.g. IoT devices, sensors, e-commerce sites, wearable devices, …) and by people (e.g. social media: Facebook, WhatsApp, YouTube, …). Another key enabler for the big data revolution has been technological progress, allowing for an increase in storage and computing solutions. In regard to this, huge importance is given to **cloud computing**, a **delivery model** for computing power that gives users **on-demand, pay-per-use access to a shared pool of configurable computing resources** (servers, storage, databases, software, network, …) **with fast provisioning and deployment times**. Cloud offerings are services, following the XaaS (X as a Service) model, including:

- **Software as a Service (SaaS)**: the consumer uses the provider's applications running on the provider's cloud infrastructure (e.g. Facebook, Gmail, etc).

- **Platform as a Service (PaaS)**: the consumer can create custom applications using programming tools supported by the provider and deploy them onto the provider's infrastructure (e.g. Google App Engine, etc).
- **Infrastructure as a Service (IaaS)**: the consumer can use computing resources within the provider's infrastructure, upon which they can deploy and run arbitrary software, including OSs and applications (e.g. Amazon EC2, etc).

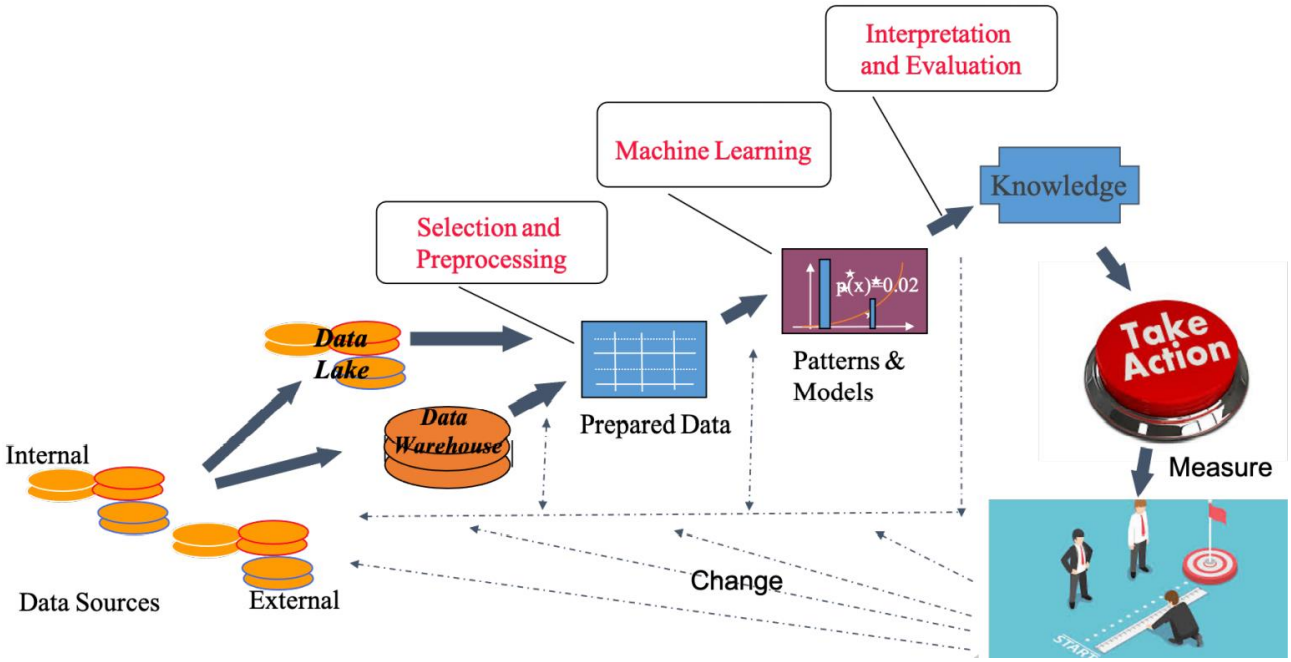
The more structured a service is, the less control a user has on it and vice versa. By using **cloud services**, we obtain significant **benefits**, among which we find:

- **Scalability, flexibility and elasticity**: resources are available when the client needs them and the delays for adding or removing capacity are minimal.
- **No investment in hardware (and maintenance)**: everything is set up and maintained by the cloud provider.
- **Pay-per-use**: the client will pay for only the amount of resources used, billed typically in seconds.
- **Updates are automated**: typically free of charge and deployed automatically by the provider.
- **Disaster recovery and security** are managed by the provider.
- **Accessibility**: the resources are accessible from any internet-connected device.

We now start introducing **big data**, meaning **a collection of data sets so large and/or complex and/or fast changing that they are difficult to process using traditional DBMSs or traditional data processing applications**. When we talk about big data, we usually refer to data that has three characteristics, the **three Vs**: **Variety**, **Velocity** and **Volume**. It is also important to introduce a high-level **taxonomy of data**, dividing it in three categories:

- **Structured data**: data with a fixed schema (e.g. relational tables, spreadsheets or data which could easily fit in them).
- **Semi-structured data**: data that may have a schema but that could also change over time (e.g. XML, JSON).
- **Unstructured data**: data that does not have an associated data model, making up roughly 80% of the total available data (e.g. audio, images and videos).

To solve big data problems we need **data mining**, a process which can be seen in the following picture:



We start from **data sources** that could be either **internal** or **external** to our organization, we store this data in a more or less structured storage facility, like **data warehouses** or **data lakes**, applying **selection and preprocessing** and obtaining **prepared data**. We can now use **machine learning** to find **patterns** and **models** that we can **interpret** and **evaluate** in order to gain **knowledge**. This will allow us to **take actions**, which will then be **measured** and **adjusted**.

It is important to keep in mind that in this course we refer to **data mining as the discovery process** found in the picture above, while we define **machine learning for data mining** as **the core of learning models and algorithms which allow us to extract actionable patterns from data**.

02 – Introduction to Business Intelligence, Data Warehouse and DFM

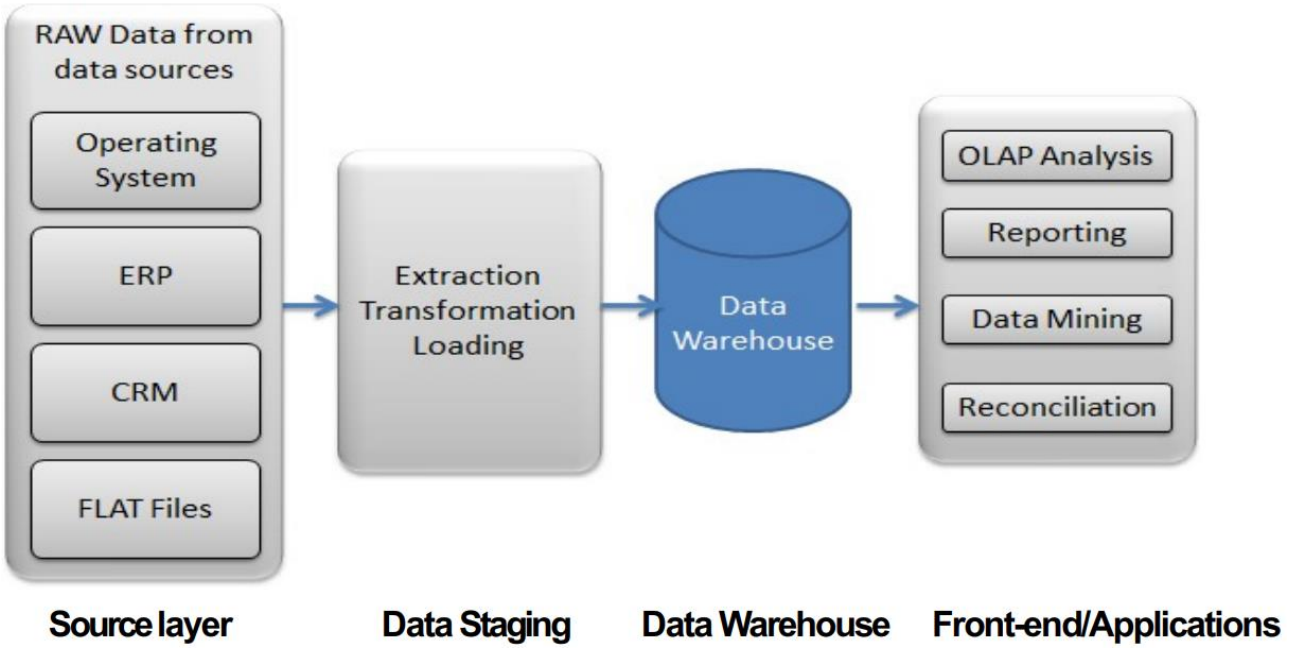
We start from the definitions of Business Intelligence that Gartner and Forrester Research gave (we have seen them in the previous chapter):

- "Business Intelligence (BI) is an umbrella term that includes the applications, infrastructure, tools and best practices that enable access to and analysis of information to improve and optimize decisions and performance" (Gartner).
- "Business Intelligence is a set of methodologies, processes, architectures and technologies that transform raw data into meaningful and useful information used to enable more effective strategic, tactical and operational insights and decision-making" (Forrester Research).

Starting from these two, we then give our own definition, which is **the process of**:

- **Transforming raw data into useful information to support effective and efficient business strategies**.
- **Capturing the business data and getting the right information to the right people, at the right time and through the right channel**.

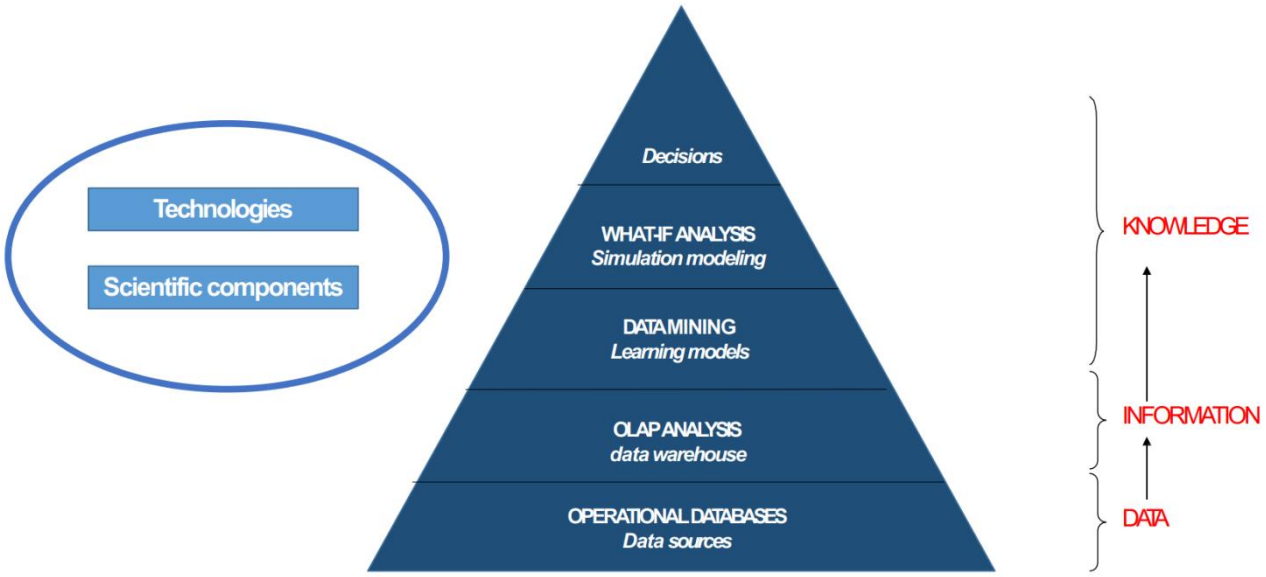
The **data warehouse (DWH)** is one of the **most important tools to support the Business Intelligence** process shown in the picture below:



To have a complete BI platform that can allow flexible and effective business analysis we need an ad-hoc infrastructure, both in terms of hardware and in terms of software, consisting of:

- Ad-hoc hardware.
- Network infrastructure.
- Databases.
- Data warehouse.
- Front-end software for data visualization.

Business Intelligence starts from huge amounts of data, reducing and refining it to capture its value and to derive decisions. This is typically represented in the form of a pyramid, shown in the picture below:



We now move on to the **data warehouse**, defined informally as an **optimized repository that stores information for the decision-making process** (it could also be seen as a specific type of Decision Support System, DSS). While at first glance they may seem like databases, they can support different kinds of workloads that DBMSs are not fit to do, for example they:

- **Can manage sets of historical data** (while databases typically represent data snapshots).
- **Can perform multidimensional analyses accurately and rapidly**.
- **Are based on a simple model that can be easily learned by its users**.
- **Are the basis for systems that calculate indicators.**

We commonly say that **a data warehouse is a collection of data that supports the decision-making process**. It also provides the following features:

- **It is subject oriented**: it focuses on enterprise-specific concepts (e.g. we may want to know what a customer has bought, when and how much they spent, etc.).
- **It is integrated and consistent**: it integrates data from different and heterogeneous sources, providing a unified view of it all.
- **It shows the evolution over time and it is not volatile**: changes to the operational data stores are tracked and recorded, allowing to create reports that show changes over a period of time. **Data in the data warehouse** is also **static**: **once committed it is never updated or deleted, it becomes read-only and it is retained for future use**.

As we said in the previous chapter, **OLAP (On-Line Analytical Processing)** gives the users the possibility to perform **multidimensional analysis**. This is done by means of the **data cube**, in which three dimensions are put together to give insights. For example, we can consider a data cube in which the dimensions are: date, product and shop. Each cell of the cube will then represent a certain product, in a certain shop, in a certain day and it will

contain a set of measures, for example the number of sales or the profit. Examples of OLAP queries are: "which products maximize the profit?", "what is the total revenue per product category and state?", "what is the relationship between profits gained by two different products?", "what is the revenue trend in the last three years?". It is also important to note that **data cubes support the idea of hierarchy**: the product "apple" may be part of the sub-category "fruit", which is part of the "edible" category of items (this is strictly related to the idea of **aggregation**).

It is worth to briefly compare **OLTP** with **OLAP**:

OLTP	OLAP
Interactive data processing system based on **transactions**	Interactive data processing system for **dynamic multidimensional analyses**
Each transaction **reads and writes a small number of records** from tables characterized by simple relationships	Each query involves **huge amounts of records** to process a set of numeric data, summing up the performance of the enterprise
OLTP systems have an essential **workload core "frozen" in application programs**	The **workload changes over time**

We can also compare **databases** and **data warehouses**:

Features	Operational Databases	Data Warehouses
Users	Thousands	Hundreds
Workload	Preset transactions	Specific analysis queries
Access	To hundreds of records, write and read mode	To millions of records, mainly read-only mode
Goal	Depends on applications	Decision-making support
Data	Detailed, both numeric and alphanumeric	Summed up, mainly numeric
Data Integration	Application-based	Subject-based
Quality	In terms of integrity	In terms of consistency
Time coverage	Current data only	Current and historical data
Updates	Continuous	Periodical
Model	Normalized	Denormalized, multidimensional
Optimizations	For OLTP access to a database part	For OALP access to most of the database

A few things to note:

- Quality in **databases** is defined in terms of **integrity**, as in "this field cannot be NULL" or "the balance must be positive".
- Quality in **data warehouses** is defined in terms of **consistency**: we may have inconsistent data across data sources and we must be able to solve these

inconsistencies (e.g. a person has two different addresses, we must find which of the two is valid).

Related to data warehouses we have **data marts**, defined as **a subset or an aggregation of the data stored to a primary data warehouse**. They **include pieces relevant to a specific business area, corporate department or a category of users** and:

- **Are used as building blocks** while incrementally developing data warehouses.
- **Mark out the information required by a specific group of users to solve queries**.
- **Can deliver better performance** due to their smaller sizes.

OLAP – On-Line Analytical Processing

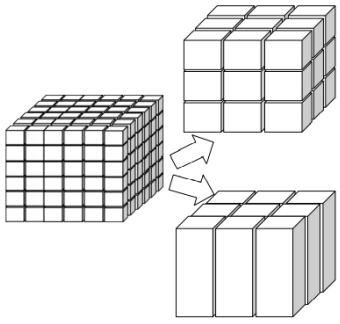
OLAP analyses allow users to interactively navigate the data warehouse information. Data is typically **analyzed at different levels of aggregation** (choosing to see more or less details)**, by applying subsequent OLAP operators**, each yielding one or more different queries **in a so-called OLAP session**.

There are a few different **OLAP Operators**, namely:

- **Roll-up**.
- **Drill-down**.
- **Slice-and-dice**.
- **Pivot**.
- **Drill-across**.
- **Drill-through**.

Roll-up

Roll-up causes an **increase in data aggregation** and **removes a detail level from a hierarchy** (we reduce the number of cubes in the data cube).



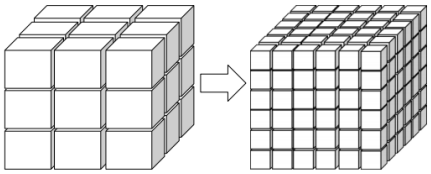
			2015		2014	
			Jan-15	Feb-15	Jan-14	Feb-14
Category	**Type**	**Product**				
Food and Beverages	Dairy products	White milk	90	90	60	80
		Chocolate milk	60	80	70	70
		Yogurt XY	20	30	30	35
	Beverages	Cola	20	10	35	30
		Orange Juice X	50	60	60	45

	2015		2014	
	Jan-15	Feb-15	Jan-14	Feb-14
Type				
Dairy products	170	200	160	185
Beverages	70	70	95	75

In this example, we apply roll-up, disregarding the "product" level, aggregating its values.

Drill-down

Drill down is the complement to the roll-up operator: it **reduces data aggregation and adds a new detail level to a hierarchy** (e.g. from category to subcategory).



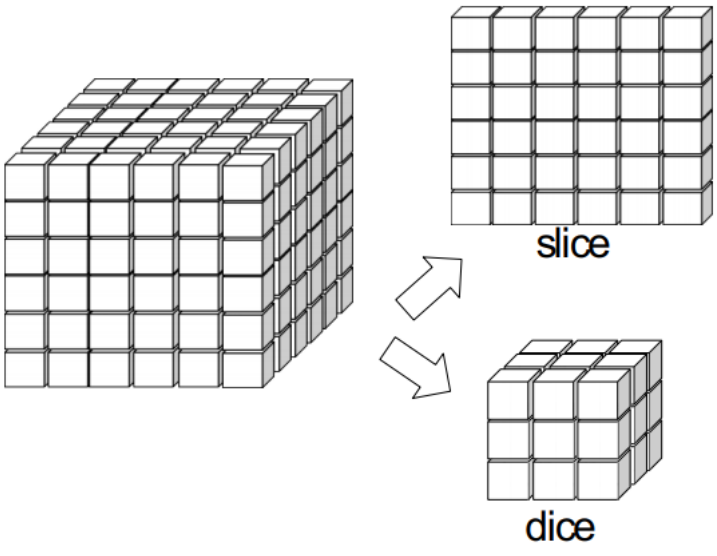
Type	2015	2014
Dairy products	370	345
Beverages	140	170

			2015		2014	
			Jan-15	Feb-15	Jan-14	Feb-14
Category	**Type**	**Product**				
Food and Beverages	Dairy products	White milk	90	90	60	80
		Chocolate milk	60	80	70	70
		Yogurt X	20	30	30	35
	Beverages	Cola	20	10	35	30
		Orange Juice X	50	60	60	45

In this example, we see that beverage sales have decreased, so we may want to see which products have lost more sales.

Slice-and-dice

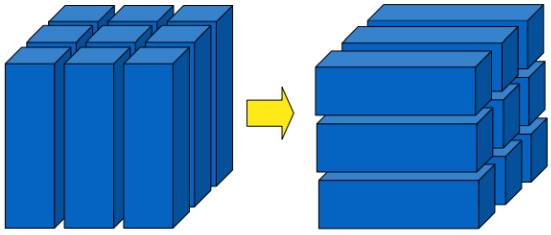
The **slicing** operator **reduces the number of cube dimensions after setting one of them to a specific value** (e.g. Category = "Food and Beverages"), while the **dicing** operator **reduces the set of data being analyzed by a selection criterion**.



Slicing means choosing a specific value for a dimension, while dicing means selecting a range among multiple dimensions.

Pivot

Pivoting implies a change in layouts, aiming at analyzing a group of data from a different viewpoint.

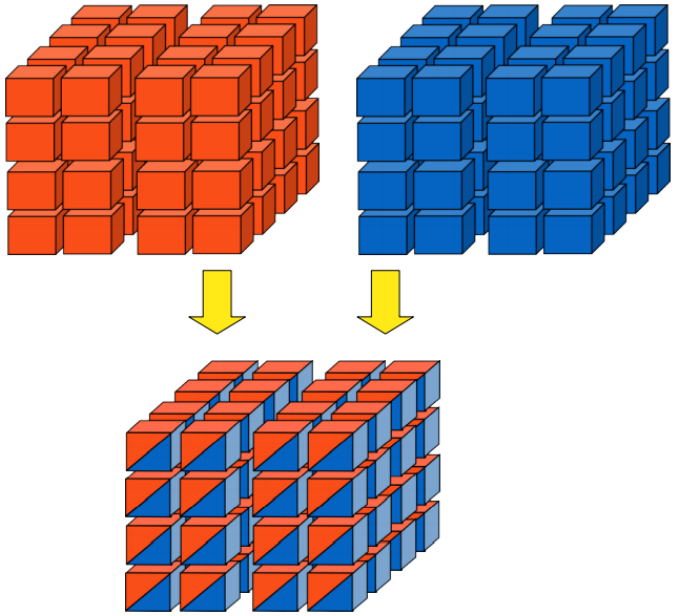


	2015	2014
Type		
Dairy products	370	345
Beverages	140	170

Type	Year	Quantity sold
Dairy products	2015	370
Dairy products	2014	345
Beverages	2015	140
Beverages	2014	170

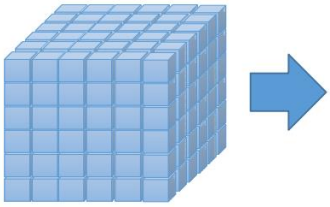
Drill-across

Drill-across allows to create a link between concepts in interrelated cubes, to make it easier to compare them.



Drill-through

Drill-through switches from multidimensional aggregate data to operational data in sources or in the reconciled layer.

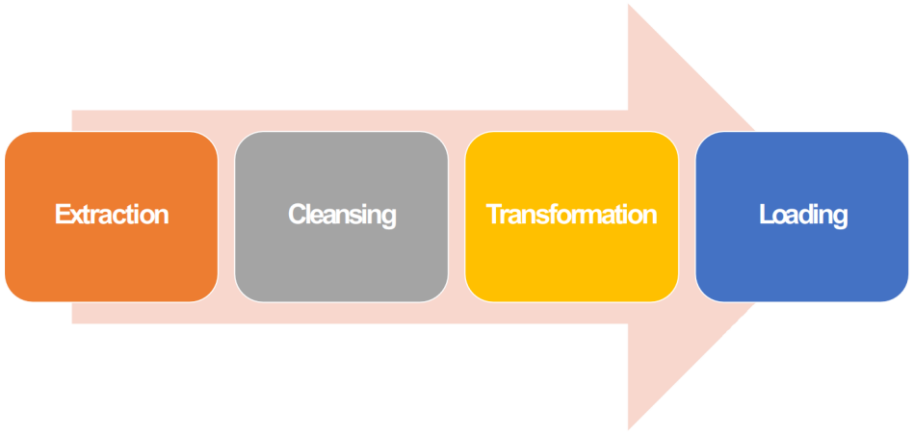


Order ID	Order Date	Ship Date	Ship Mode	Customer Name	Segment	City	State	Country
IT-2013-1191900	15/06/2013	15/06/2013	Same Day	Georgia Rosenberg	Corporate	Houilles	Ile-de-France	France
ES-2012-5315807	20/09/2012	23/09/2012	Second Class	Sonia Cooley	Consumer	Drancy	Ile-de-France	France
ES-2014-5488008	25/08/2014	31/08/2014	Standard Class	Karen Seio	Corporate	Magdeburg	Saxony-Anhalt	Germany
ES-2014-5488008	25/08/2014	31/08/2014	Standard Class	Karen Seio	Corporate	Magdeburg	Saxony-Anhalt	Germany
ES-2014-5488008	25/08/2014	31/08/2014	Standard Class	Karen Seio	Corporate	Magdeburg	Saxony-Anhalt	Germany
ES-2014-1668222	27/08/2014	02/09/2014	Standard Class	Vivek Grady	Corporate	Wetter (Ruhr)	North Rhine-Westpha...	Germany
ES-2014-1668222	27/08/2014	02/09/2014	Standard Class	Vivek Grady	Corporate	Wetter (Ruhr)	North Rhine-Westpha...	Germany
ES-2014-1668222	27/08/2014	02/09/2014	Standard Class	Vivek Grady	Corporate	Wetter (Ruhr)	North Rhine-Westpha...	Germany

We may want to see which is the origin of the data.

ETL – Extraction, Transformation and Loading

The **ETL process extracts, integrates and cleans data from operational data sources**, in order to feed it to the data warehouse layer. The process can be represented with an arrow:



The process consists of:

- **Accessing the data sources**.
- **Cleaning the data**, as there may be missing values or mistakes.
- **Transforming the data**, as it may be too detailed or there may be different ways of representing data (e.g. decimals separated with a dot or a comma).
- **Loading the data into the data warehouse**.

Extraction

Data extraction may be performed on **structured** and **unstructured data sources** in different ways. We have a:

- **Static extraction** when we **populate the data warehouse for the first time** with a snapshot of the operational data.
- **Incremental extraction** when we need to **append a set of changes** to the data warehouse regularly. This can be done by leveraging **timestamps associated with operational data** and **triggers associated with transactions**.

Cleansing

Cleansing is a **set of procedures to improve data quality** by **standardizing it**, **correcting mistakes** and **inconsistencies**, for example:

- **Duplicate data** (e.g. a customer is recorded many times in the customer database due to multiple registrations in different shops).
- **Missing data** (e.g. the customer's age).
- **Unexpected use of fields** (e.g. a free text field used to store a phone number).
- **Impossible or wrong values** (e.g. 30th February).
- **Inconsistent values for a single entity because of different practices used** (e.g. UNIBO and University of Bologna).

- **Inconsistent values for own individual entity because of typing mistakes** (e.g. Steet instead of Street).

This operation is very costly and time consuming but leads to better data, and thus to better insights. Each type of problem requires different techniques to be solved, we will see three main techniques:

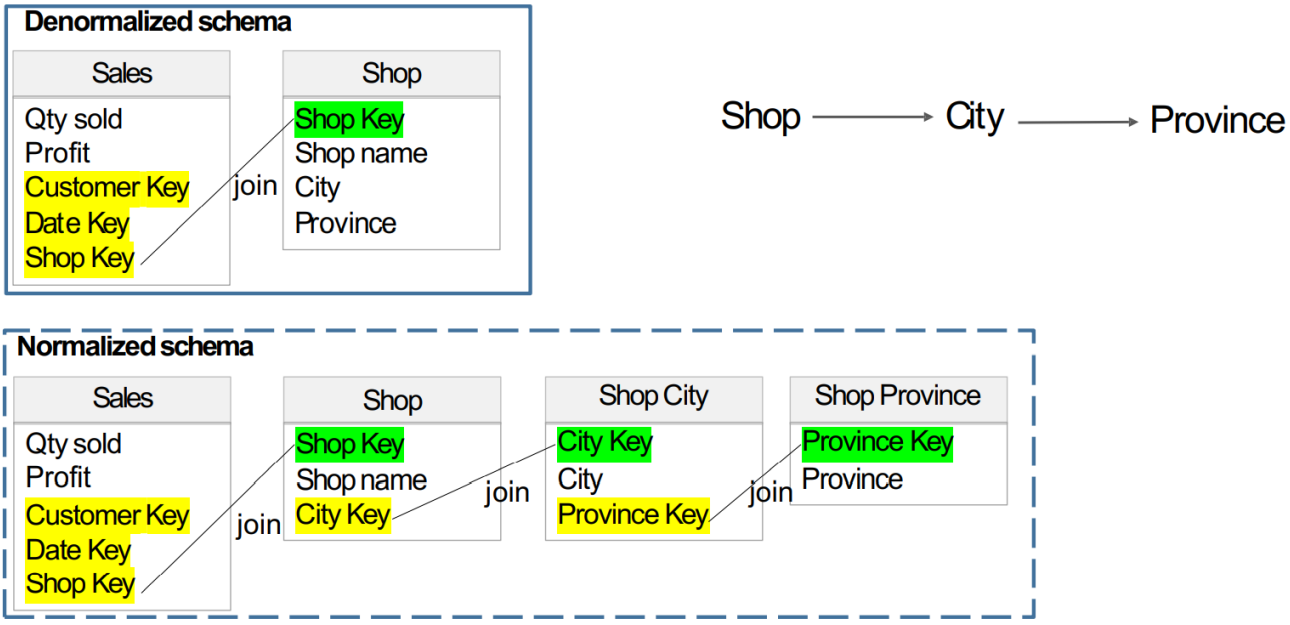
- **Dictionary-based techniques**: they are used to check the correctness of the attribute values via **lookup tables** and **dictionaries** that enable us to search for abbreviations and synonyms. These techniques are **only applicable to known and limited domains** and they are suitable for solving problems such as **typing mistakes** and **format discrepancies**.
- **Approximate merging**: we use this technique when we need to **merge data coming from different sources and we do not have a common key to identify matching tuples**. This is done via **approximate joins** (based on shared attributes like "customer surname" and "customer address") and **similarity functions** (like the edit distance).
- **Ad-hoc algorithms**: custom algorithms based on specific business rules.

Transformation

A few examples of data transformations can be found in the following table:

Categories of transformation	Examples
Conversion: changes on data types and format	• Date conversion: from date to number (12/11/2018 →20181112) • String conversion: lowercase to uppercase (unibo→UNIBO) • Naming convention transformation: short description to long description (IT→Italy)
Enrichment: combination of one or more attribute to create new information.	• Calculation of derived data Profit = Receipts - Expenses
Separation/Concatenation	• Attributes concatenation (e.g. customer surname || customer name) • Denormalization/Normalization process. Typically, in the DWH the data is denormalized.

It is important to note that having **denormalized data** makes it easier to perform operations like roll-up and drill down, although it introduces data redundancy. An example of a denormalized schema can be found in the figure below:



Loading

The **loading** operation can be performed in two different ways:

- **Refresh**: the data warehouse is completely rewritten (typically used in combination with static extraction).
- **Update**: preexisting data is not deleted or modified and changes are appended. It is used in combination with incremental extraction to regularly update the warehouse.

Data Warehouse Architectures

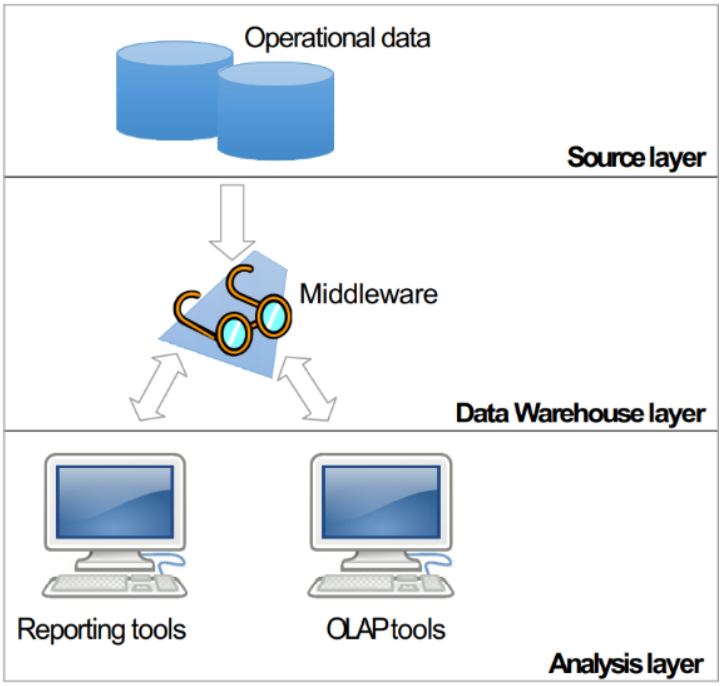
Data warehouses have a series of requirements, among which we find:

- **Separation**: analytical and transactional processing should be kept apart as much as possible, since OLAP queries might impact normal, day-to-day OLTP operations.
- **Scalability**: hardware and software architectures should be easy to upgrade as demands increase.
- **Extensibility**: the architecture should be able to host new applications and technologies without redesigning the whole system.
- **Security**: monitoring access is essential because of the strategic data stored in data warehouses.
- **Administrability**: data warehouse management should not be overly difficult.

We can also have different types of architectures: **single-layer**, **two-layer** and **three-layer**.

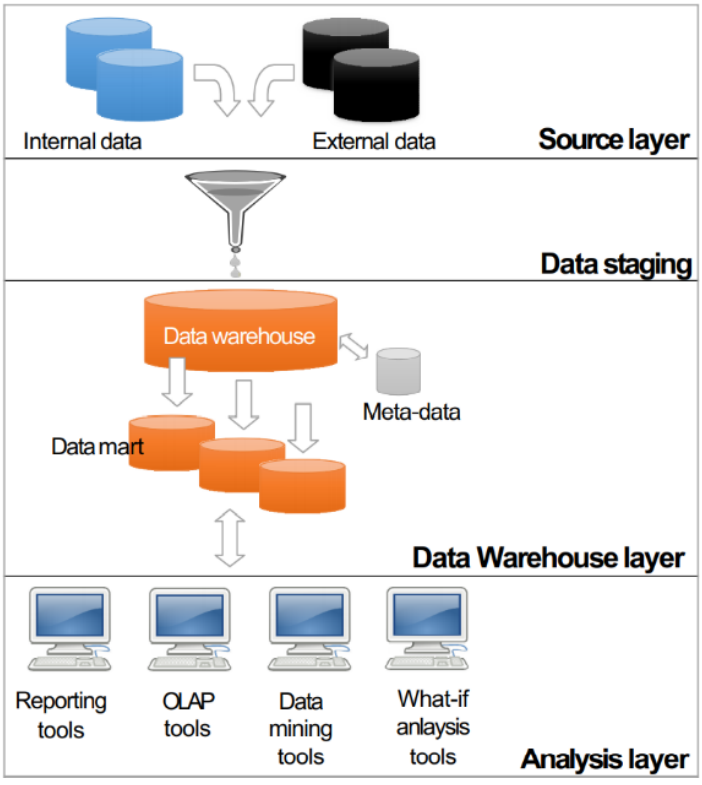
Single-layer architecture

The goal of this type of architecture is to minimize the amount of data stored, removing data redundancies. We achieve this by using a middleware acting as the data warehouse, while keeping the source layer as the only one "physically" available. Of course, this means that there is no separation between analytical and transactional processing (we do not meet the first two requirements).



Two-layer architecture

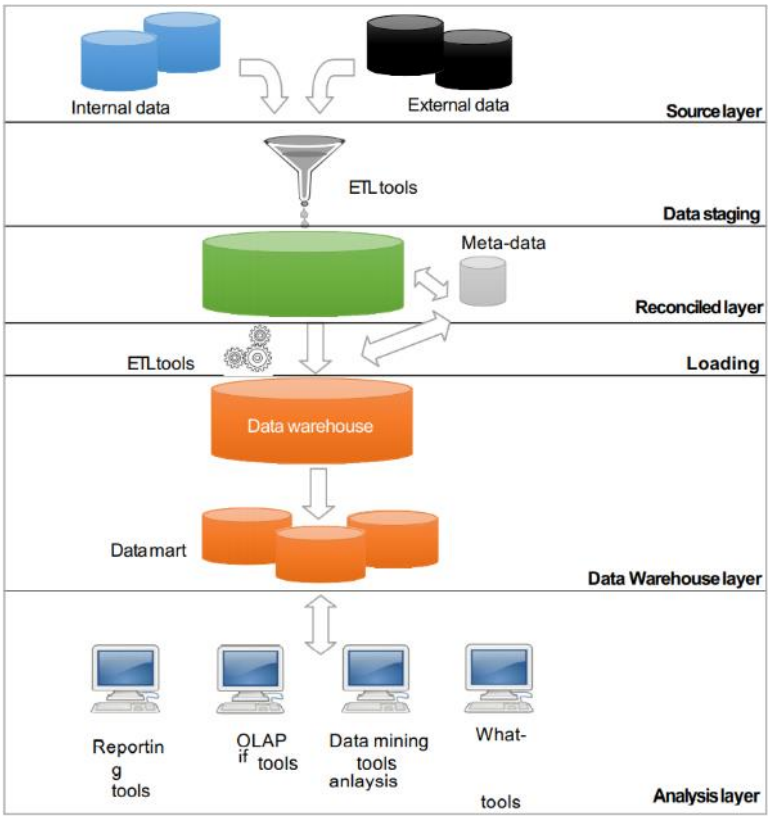
With a two-layer architecture we introduce a level of separation between the available sources and the data warehouse, called **data staging layer**. In the staging area we perform the ETL operations, extracting the data, cleaning it and integrating it into a common schema. Scalability and extensibility still suffer with this solution.



Three-layer architecture

It is the most complete solution. It has two repositories: the **reconciled layer**, in which data is stored in relational form (as tables) with most of the inconsistencies already solved and

metadata already available. The ETL process will then be applied to the data contained in this layer in order to be added to the data warehouse.



Conceptual modeling: The Dimensional Fact Model (DFM)

A **conceptual model** is a type of model which uses terms (such as "entity" and "relationship") which have a particular meaning and composition rules.

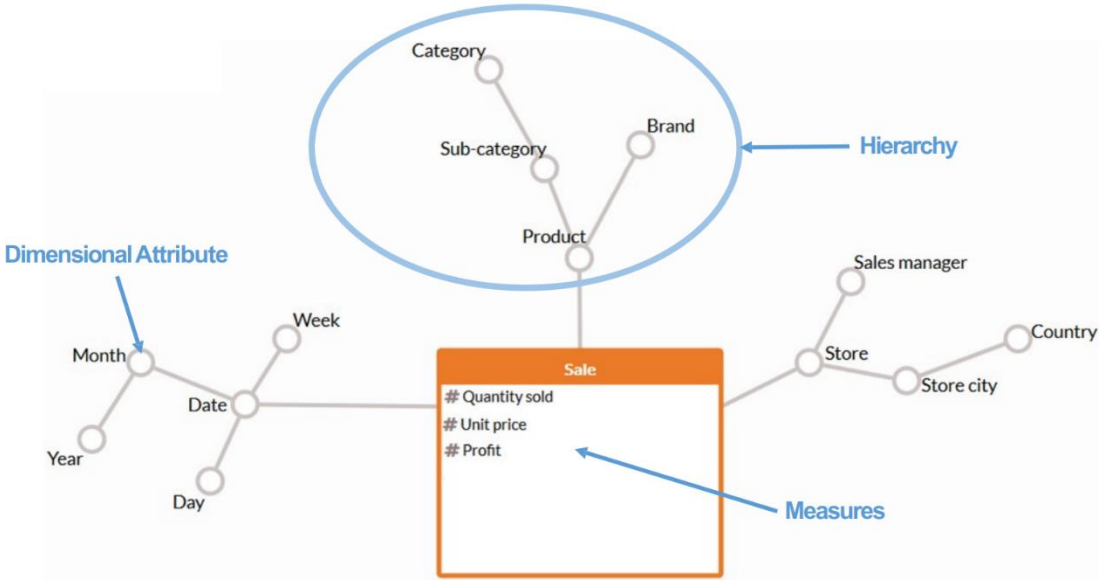
Let us suppose that we are tasked with helping a company "do something" with the data they have available. We want to start modeling the data (at a very high level) in order to store it in the data warehouse and to discuss the various concepts with the business owners. We can do so by following one of two approaches (plus a mixed one):

- **Requirement-driven approach**, based on a collaborative analysis with the user of the facts, measures and hierarchies involved (we simply ask the users what they want to do or to know). This is typical when there is no detailed information on data sources or it is too complex.
- **Data-driven approach**, in which the conceptual model of the data mart is based on the structure of operational sources.
- **Mixed approach**, a combination of the previous approaches, in which the result of the requirement analysis helps to refine the conceptual model derived from source analysis.

The primary tool for conceptual modeling in data warehouses is called **DFM**, **a conceptual model specifically created to function as a support for data mart design**. **It is graphic and based on the multidimensional model**. The objectives for this model are:

- **Providing support to conceptual design**.
- **Creating an environment in which user queries may be formulated intuitively**.
- **Favoring communication** between designers and end users to formalize requirement specifications.
- **Building a stable platform for logical design**.
- **Providing clear and effective design documentation**.

An example of dimensional fact schema built by means of the DFM is shown in the figure below:



The orange box represents a **fact**: something that happens in the real world that is relevant to our context. Inside of it we have **measures**, represented typically with numerical attributes. Each fact also has additional information connected to the fact box by means of lines and represented with circles (in this example: Date, Product and Store) called **dimensions** which, in turn, can have **dimensional attributes** (e.g. Day and Week). The branches that dimensional attributes form are called **hierarchies**: they organize dimensional attributes and complete the information brought in by the fact.

We sum these concepts up with the following table:

Concept	Description	Example
Fact	It is a concept relevant to decision-making processes. It typically models a set of events taking place within a company	Sales, purchases, orders
Measure	It is a numerical property of a fact and describes a quantitative fact aspect that is relevant to analysis.	Quantity, revenue, discount
Dimension	It is a fact property with a finite domain and describes an analysis coordination.	Date, product, store
Dimensional attribute	Dimensions and other possible attributes, always with discrete values, that describe them.	Category of product, month
Hierarchy	It is a directed tree whose nodes are dimensional attributes and whose arcs model many-to-one associations between dimensional attribute pairs. It includes a dimension, positioned at the tree's root and all of the dimensional attributes that describe it.	Date->Month->Year

There are also additional concepts shown in this table:

Concept	Description
Primary event	It is a particular occurrence of a fact, identified by on n-ple made up of a value for each dimension. A value for each measure is associated with each primary event.
Secondary event	Given a set of dimensional attributes, each n-ple of their values identifies a secondary event that aggregates all of the corresponding primary events. Each secondary event is associated with a value for each measure that sums up all the values of the same measure in the corresponding primary events.

To clarify these two concepts, we show an example that refers to the conceptual schema shown earlier:

Primary events

Date	Store	Product	Qty sold	Profit
01/03/15	Central store	Milk	20	60
01/03/15	Central store	Coke	25	50
02/03/15	Central store	Bread	40	70
10/03/15	Central store	Wine	15	150

SUM SUM

Secondary event

Month	Store	Category	Qty sold	Profit
March 2015	Central store	Food and Beverages	100	330

In this case, we can think of the secondary event as the result of a roll-up operation. The possibility of applying operators such as SUM should not be taken for granted, as it's related to the property of **additivity**: a measure is called **additive along a dimension** when you can use the SUM operator to aggregate its values along the dimension hierarchy. If this is not the case, it is called **non-additive**. **A non-additive measure is non-aggregable when you can use the no aggregation operator for it**.

For example, we can consider three types of measures:

- **Flow measures**: refer to a timeframe, at the end of which they are evaluated cumulatively (e.g. quantity sold).
- **Level measures**: are evaluated at particular times (e.g. number of products in inventory).
- **Unit measures**: are evaluated at particular times but are expressed in relative terms (e.g. unit price).

Depending on whether we consider the time dimension or not, we can apply a different set of operators (notice how some are not additive):

	Temporal Hierarchies	Non-temporal Hierarchies
Flow measures	SUM, AVG, MIN, MAX	SUM, AVG, MIN, MAX
Level measures	AVG, MIN, MAX	SUM, AVG, MIN, MAX
Unit measures	AVG, MIN, MAX	AVG, MIN, MAX

Aggregation operators can be:

- **Distributive** if we can calculate aggregates from partial aggregates (e.g. SUM, MIN, MAX).
- **Algebraic** if they require the usage of additional information in the form of a finite number of support measures to correctly calculate aggregates from partial aggregates (e.g. AVG).
- **Holistic** if they can calculate aggregates from partial aggregates only via an infinite number of support measures (e.g. RANK, in which you need all of the information to compute it).

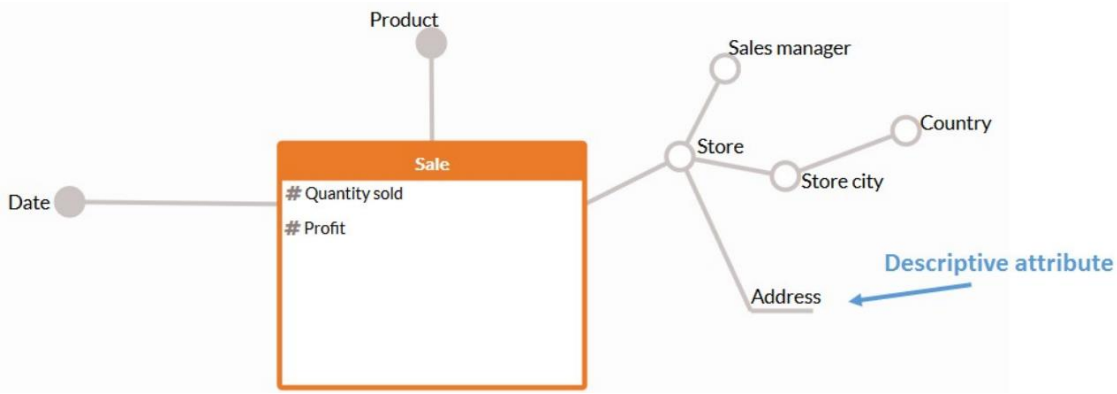
We can now consider **more advanced DFM concepts**, such as:

- **Descriptive attributes**.
- **Cross-dimensional attributes**.
- **Convergence**.
- **Shared hierarchies**.
- **Multiple arcs**.
- **Optional arcs**.
- **Incomplete hierarchies**.

- **Recursive hierarchies**.

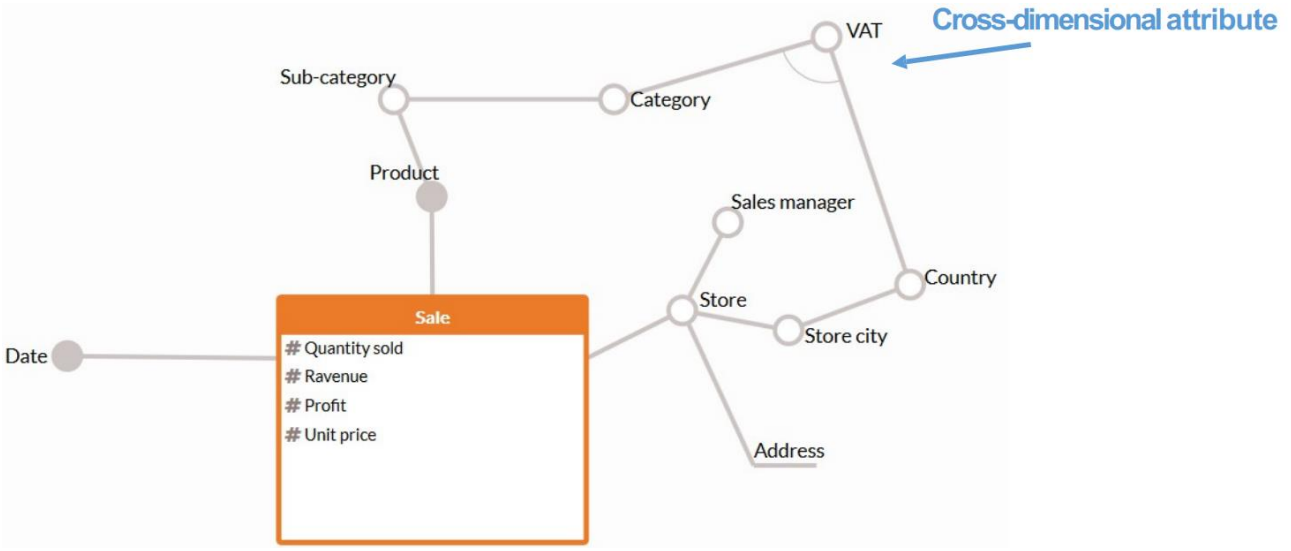
Descriptive attributes

Descriptive attributes are used to give additional information to a specific dimensional attribute but **they cannot be used as an aggregation criteria**. They are typically used to allow for more descriptive reports.



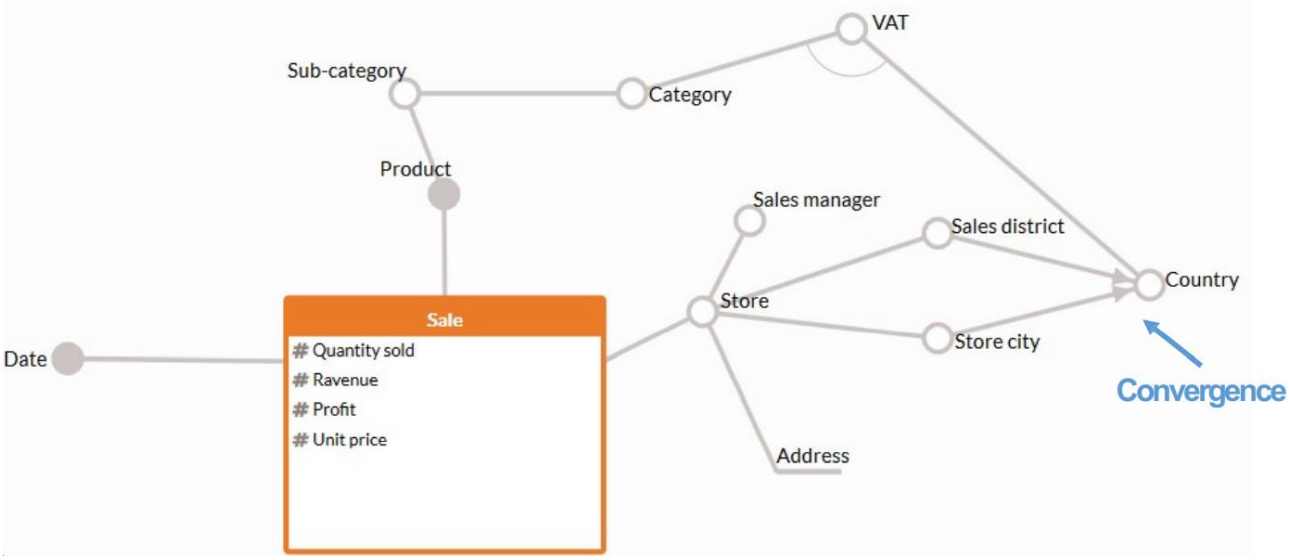
Cross-dimensional attributes

Cross-dimensional attributes are **dimensional** or **descriptive attributes whose value is defined by the combination of two or more dimensional attributes**, possibly belonging to different hierarchies.



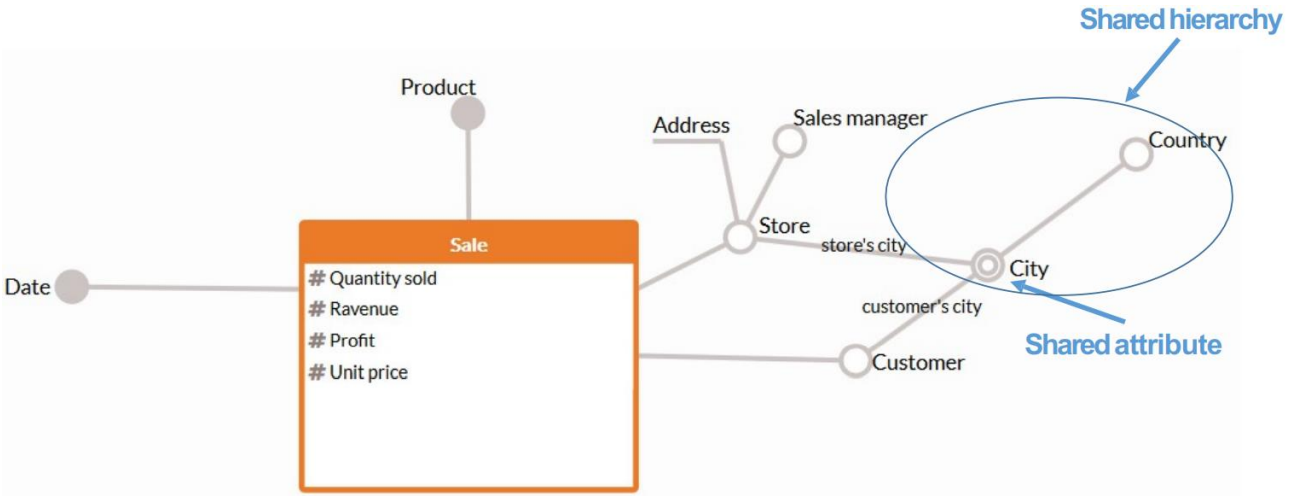
Convergence

We have a **convergence** when two or more arcs **belonging to the same hierarchy** end at the same dimensional attribute.



Shared hierarchies

Hierarchies could be **shared** and a double circle is used to emphasize the first attribute to be shared (City, in the example that follows). All descendants of the shared attribute are shared, too. For each incoming arc a role must be added (e.g. "customer's city").

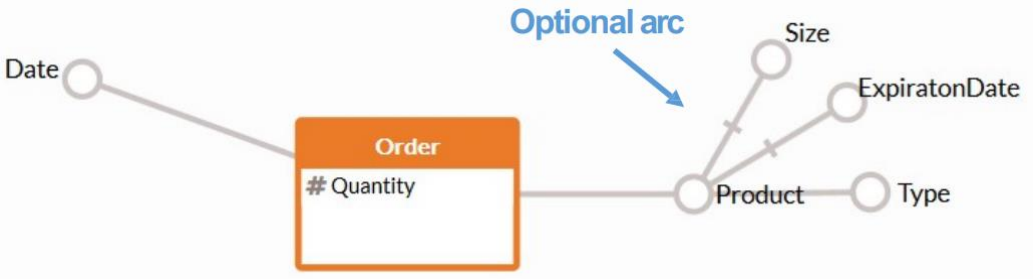


Multiple arcs

We can have **multiple arcs** going from an attribute *a* to an attribute called *b* to signify that there is a **many-to-many association** between the two.

Optional arcs

Optional arcs are used to model scenarios for which an association represented in a fact schema is not defined for a subset of events. For example, for one or more values of Product, the ExpirationDate and all the possible descendants in the hierarchy may be undefined.

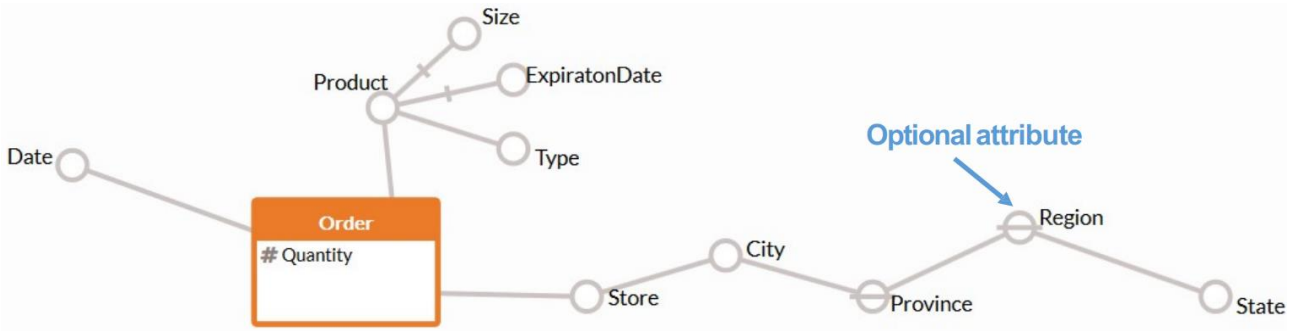


When one or more optional arcs exit from the same attribute *a* (e.g. Product), their **coverage** can be specified:

- The **coverage** is **total** if the value of at least one of the children is linked to each value of *a*. If, instead, values of *a* exist for which all of the children are undefined, the **coverage** is **partial**.
- The **coverage** is **disjoint** if there is a value for at most one of the children corresponding to each value of *a*. If, instead, values of *a* exist linking to values of two or more children, the **coverage** is **overlapping**.

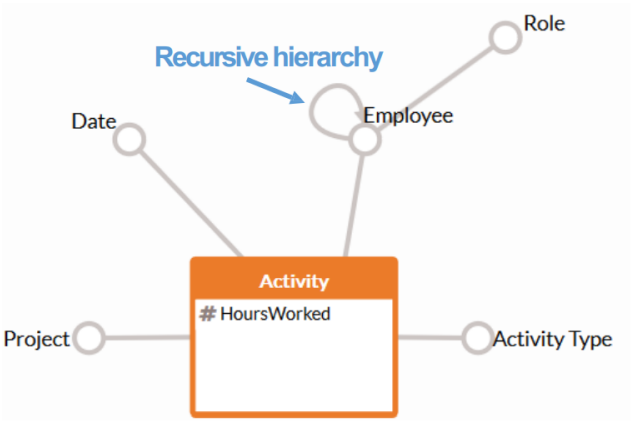
Incomplete hierarchies

An **incomplete hierarchy** is one in which **one or more levels of aggregation prove missing in some instances** (because they are not known or defined).

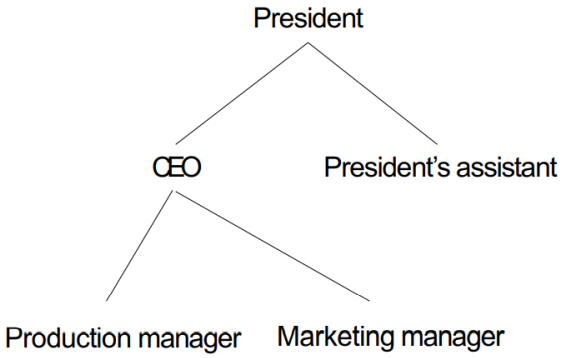


Recursive hierarchies

A **recursive hierarchy** represents a parent-child relationship among the levels.



Example: role hierarchy in an unbalanced company organization chart



Logical Design

Once the **Dimension Fact Model** has been completed, we must move on to the **logical design phase**, in which we **define the data structure** (e.g. set of tables and relationships between tables) that represent data marts according to the preselected logical model and optimize the performances by fine-tuning the structures. The logical design phase includes a set of steps that, starting from the conceptual schema, make it possible to define the logical schema of a data mart.

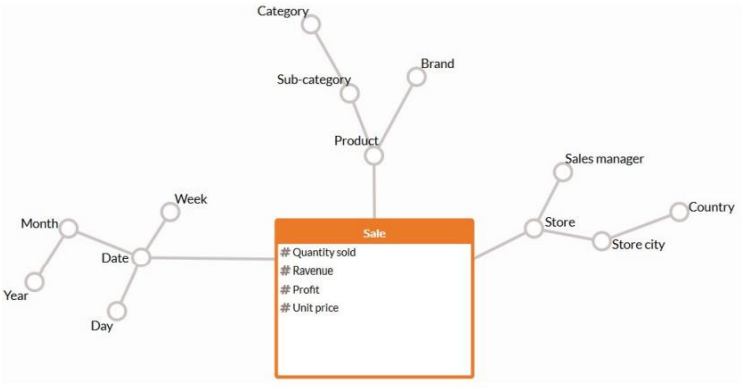
There are three main steps to implement a logical schema in a relational DBMS:

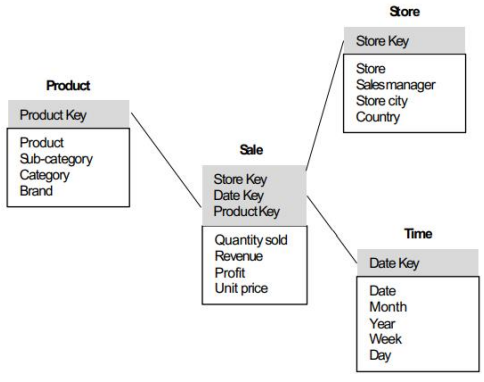
- **Translating the fact schemata into a logical schemata**, mainly using the **star** or the **snowflake schemata**.
- **Materializing views**: a set of secondary views that aggregate primary view data to improve query performance.
- **Fragmenting fact tables vertically and horizontally**.

We will now explain in further details the **star** and the **snowflake** schemata.

Star Schema

A **star schema** is characterized by **fact** and **dimension tables**. **A fact table contains all the measures and descriptive attributes linked to a fact. A dimensional table is created for each dimension and it includes all the hierarchy attributes**.

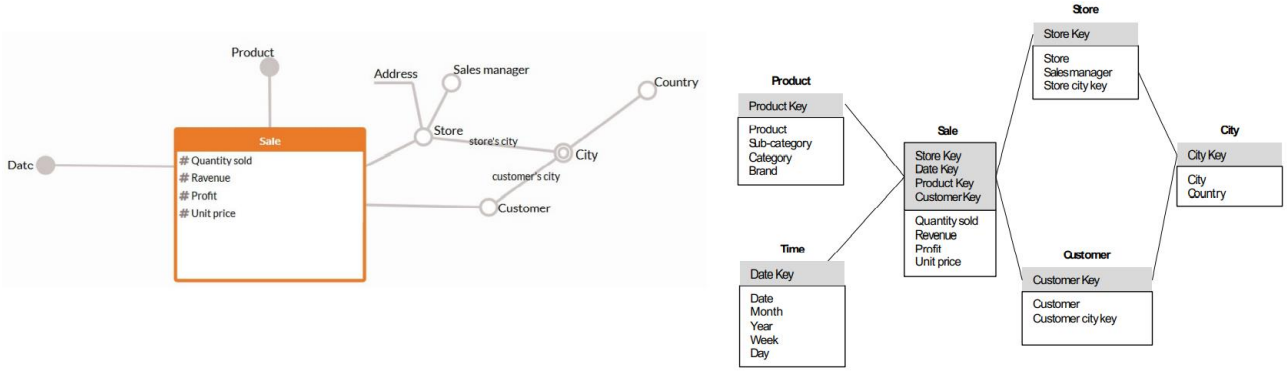




We have a table for the fact and one for each hierarchy. The fact table will have a foreign key for each of the hierarchy tables.

Snowflake schema

The **snowflake schema** is a star schema variant, with dimension tables partially normalized.



Some tables are split into different sub-tables (normalized) to reduce data redundancy and for increased query performance.

03 - Data Lakes

The digital era brought in new demands:

- **Granular personalization and individual treatment**.
- **Recommendations** (e.g. Amazon recommending us things to buy based on our buying history).
- **Instant, real-time processing and decision making** (strict time requirements).
- **360° view of a customer**, to better fulfill their requirements.
- **Insights**: understand things hidden among data, especially related to each specific customer-company **context** in a way that is **actionable** (allowing to define a strategy that can benefit both the customer and the company).
- **Smart, analytics**-enabled services.
- **Ability to adapt to changes in the business landscape**.

Along with new demands there have been new data types, for example **dark data**, meaning information assets collected, stored (but possibly not processed) by organizations that have not yet been used for analytics.

A solution for these challenges could be found in data lakes. They are:

- A **repository of data stored in its natural/raw format**, usually in the form of object blobs or files. As soon as data becomes available from any source, we store it, deferring the decision of what to do with it.
- **Dive anywhere, flexible access, schema on read**. We must be able to access data (less trivial than it sounds) in a flexible way (as in not limited by a query language, e.g. SQL). By having a "schema on read" we sacrifice ease of use a bit by requiring

additional processing to "frame" the data in order to have important gains in versatility (data is simply dumped into the lake). DBMSs and data warehouses operate following the opposite methodology, the "schema on write", requiring the schema to be defined before inserting the data.

- **Quickly ingest anything**: we do not enforce "schema on write", making **data ingestion** (the flow from the data sources to the repository) easier.
- **All data in one place**: all our data assets are logically centralized in one place, acting as a **single source of truth**.
- **Low cost, scalable storage**: computing and storage facilities are decoupled. While before we would buy computers with a certain amount of storage attached, we now use dedicated storage facilities (e.g. the cloud).
- **Future proof**: our models, storage systems and data management should be flexible enough to allow for future extensions and changes of perspective.

The huge amount of new data allows for new business opportunities:

- Massive personalization of products, services and market channels, enabling B2C in any industry.
- Real time and productive risk analytics.
- Operation optimization (e.g. predictive maintenance and supply chain optimization).

These new opportunities required changes in the traditional data analytics architecture, where previously operational systems (in charge of supporting day-by-day operations like ticketing buying, selling, supply chain management) would feed data into archives and perhaps into systems which could be used for analytics, data mining, data warehousing and business intelligence. This architecture had intrinsic **limits to scalability and high costs** due to computing and storage needing to be scaled together and completely **lacked support** for unstructured data, ad-hoc analytical queries, machine learning and data science workloads.

Our new requirements can be summed up as:

- **Support for a wide range of analytics use cases**.
- **Data should be made available** (at some level of aggregation) **to the business users**, as to put them in the best conditions to take decisions.
- **The data architecture should be flexible and scalable**, enabling the support for additional storage and computation.

- **Support for data warehousing, business intelligence and data science workloads**, typically 20% structured (DW, BI) and 80% unstructured.
- **Short time to value**: short time between figuring out how to satisfy a need and the actual result.
- **Holistic metadata management, governance, security monitoring and usage analytics**: these concepts should be managed at high level, without going into details. (reminder: metadata is "data about data". They describe the nature of the data, e.g. database schema, data cube schema, design of the ETL process, …).

Depending on the use cases, we can use different solutions:

Use cases	Tech solutions	Feature trends
Mission critical, low latency, insight apps	Data Warehouse / Hot	- More expensive HW/SW - Use case-specific data - Less latency - More governance - Higher data quality - Used by end-users and data analysts
Agile insight apps	Data Hub / Warm	↑↓
Staging area, data mining, searching, profiling, cataloging	Data Lake / Cold	- Less expensive HW/SW - All enterprise data - More latency - Less governance - Lower data quality - Used by data scientists

While for mission critical, low latency and insight applications we need data warehouses, where data is loaded in (semi-) real time and that require more expensive hardware, if we have less strict requirements, we can use the less expensive data lake. They are based on the concept of staging area, a space in which data is simply stored and in which we can perform searches, profiling, cataloging and even data mining. These systems need less governance, meaning that data may not be as consistent or high quality.

We now have a need for **insight-driven systems**:

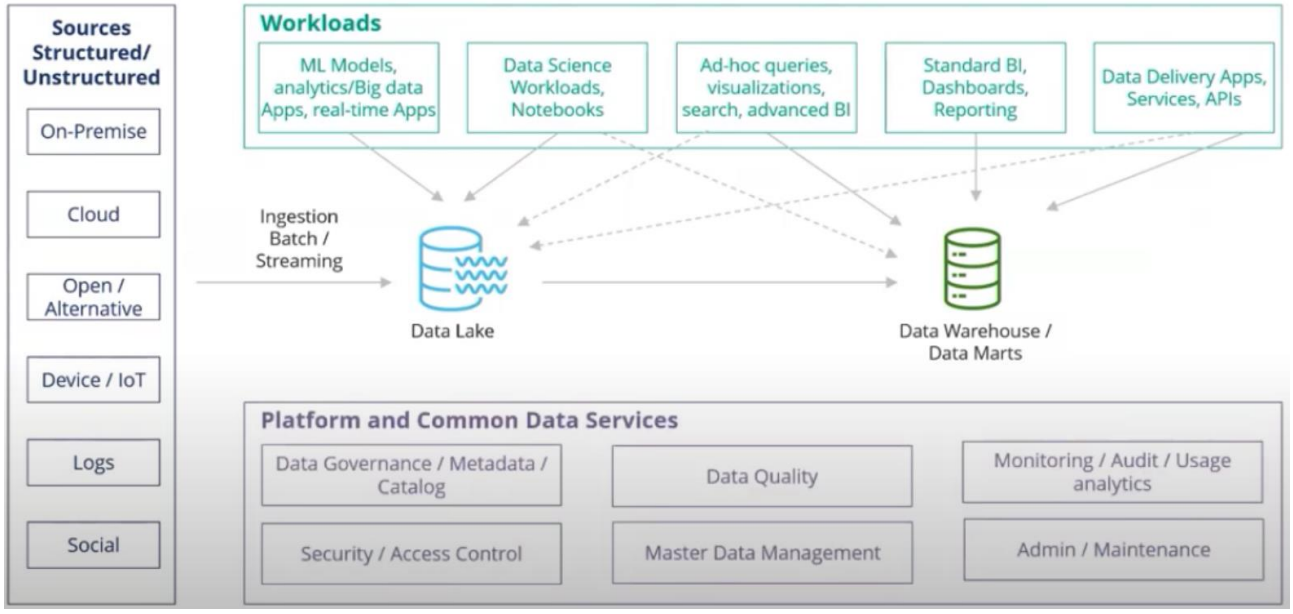
	Traditional Data Systems	Insight-driven Systems
Data Sources	Structured, relational data from transaction systems, relational and operational data stores	Traditional sources + semi and unstructured sources: logs, web sites, social media, alternative data providers

Data Movement (Ingestion)	Amount of data that could be moved is limited	Virtually unlimited amount of data that could be moved into the system in original form at a required latency
Storage	Limited volume of stored data	Unlimited volume of data. Source of truth for all source data
Data Structure	Schema is designed upfront, before data is captured. Data format dictated by storage/technology	Schema is not fixed when data is captured. Variety of supported open formats for analytics pipelines (relational document, graph, etc.)
Data Transformations	Upfront, time consuming data cleansing, enrichment, integration and transformation, "single source of truth" of trusted and widely usable data	Add transformation for an ad-hoc data querying and data science related feature engineering
Analytics	SQL Queries, BI tools, full text search	+ self-service BI, big data analytics, realtime analytics, machine learning, data exploration/visualization. Allow users to securely explore and query raw data. Easily introduce new types of analytics
Price/Performance	Highest cost storage/fastest query results	Low-cost storage + performance scale/speed/cost tradeoffs
Users	Business (analysts)	+ Data Scientists, Data analysts and engineers. Developers
Data Quality	High	Low and high, depending on use case. Data quality must be transparent
Data Sharing and Collaboration	Very limited, mainly via central DW/BI team	Rich. Raw and transformed data sets, analytical models, dashboards can be easily and securely shared

These new systems postpone the need for transformations and add new tools for data exploration, visualization and exploration. Since the design step is way less complicated, costs are also reduced. Depending on how much the company is willing to spend, these

systems can offer both low or high data quality to its users (typically data scientists, data analysts and engineers) and it is critical for them to be aware of it.

The modern data architecture can then be found in the following picture:



Data lakes can handle data flowing in from a variety of sources (on-premises, cloud, open data, IoT devices, logs, social streams, …) in both **structured** and **unstructured** forms. Plenty are also the workloads that can be executed on the data: data delivery, business intelligence and reporting, ad-hoc queries for analysts, data science workloads and machine learning models. A very important piece of this modern architecture is the possibility to offer a certain set of services in a common and shared way; these include metadata and catalogs (data governance), security and quality control, maintenance and monitoring.

Data architecture has evolved from traditional, on-premises data warehouses and big data solutions (where we simply had to have a way to deal with the big data that were pervasive in our company) to cloud-based data warehouses that are simpler to set up and use and cloud data lakes. The idea behind cloud data lakes is to have a place in which data that may or may not be used is stored, while having by default a series of additional standard services for data governance, access control, maintenance and auditing.

A comparison of the data architectures can be found in the following picture:

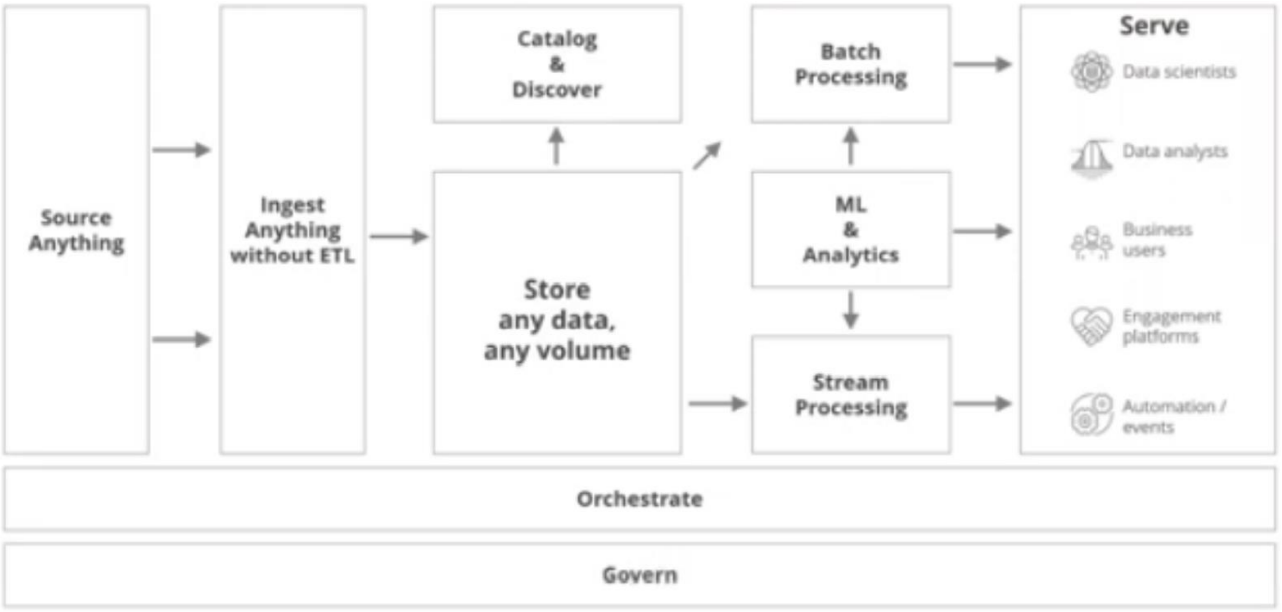
	Traditional DW/BI	Modern Cloud DW	On-prem Big data (e.g. Hadoop Cluster)	Cloud Data Lake
Data formats	Structured	Structured + Semi-structured	Any	Any
Schema	On-write	On-write + on-read for semi-structured	On-read	On-read
Independently scale Storage and Compute, Elasticity	No	Yes	No	Yes
Set-up and Maintenance cost	Very high	Low to Very Low	High to Very High	Low
Relational data support	Strong	Strong	Weak	Weak to Ok
New Data Ingestion	Slow to introduce	Fast if data format is supported	Fast	Fast
Data Quality	Very High	Very High + Any	Any	Any
Workloads	BI, Dashboards, Reporting	Advanced BI, Dashboards, Reporting, Ad-hoc queries	Ad-hoc queries, Data science and Machine Learning, Big Data Apps, Weak BI/Visualization	Ad-hoc queries, Data science and Machine Learning, Big Data Apps, Weak BI/Visualization
Data granularity	Often aggregated only	Aggregated + detail, and raw	Raw and calculated	Raw and calculated
Durability and Resilience	Limited, Expensive	Strong, Multi-cloud	Limited, Expensive	Strong, Can be multi-cloud
SLA for main workloads	Tight, optimized	Tight, optimized, elastic	Loose, unless specifically optimized	Loose, unless specifically optimized
Flexibility of architecture and tools	Low	Average	Average	High
Relational data support	Strong	Strong	Weak	Weak to Ok

We will focus on just a few items: while standard solutions could handle only structured or semi-structured data formats, new solutions should be able to handle any kind of data, even unstructured, enforcing the schema only on read, scaling well and with low setup and maintenance costs. Newer architectures also enable us to have increased flexibility when dealing with detailed data, allowing for increased data granularity. By leveraging cloud solutions, we also obtain greater durability and resilience, while their SLA may come a bit shorter in terms of guarantees compared to traditional on-premises solutions.

With data lakes, we have a variety of users accessing data, each with a different background and level of technical knowledge. Typical users are:

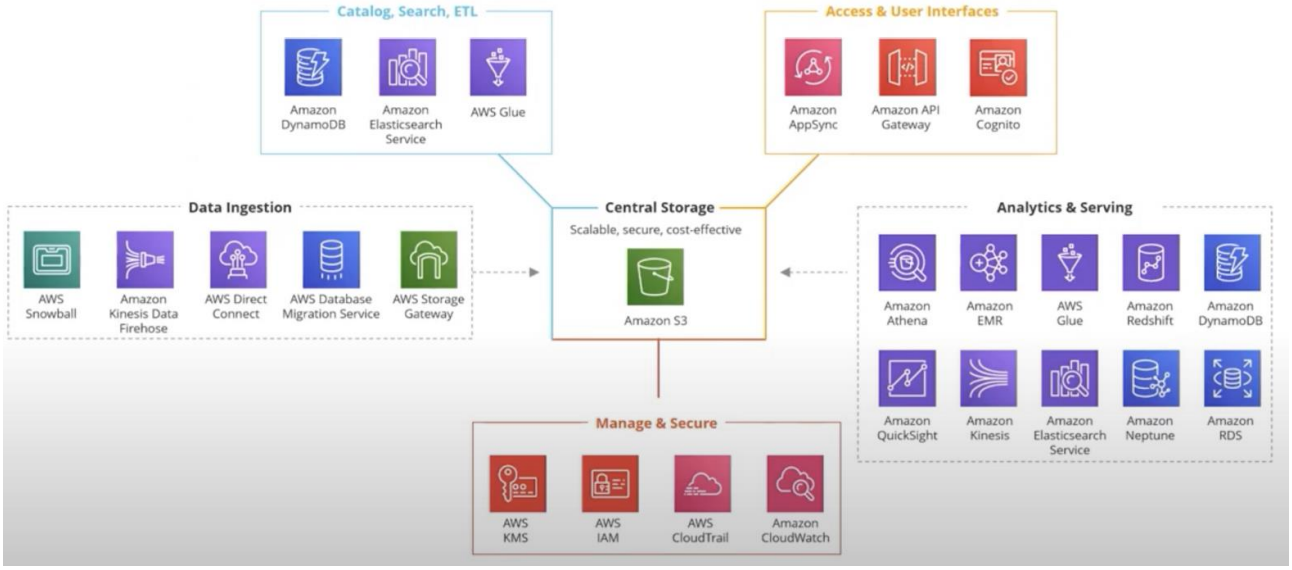
- **Business users**: using pre-configured dashboards and reports.
- **Business analysts**: using self-service BI and ad-hoc analytics, possibly building their own models to provide business insights.
- **Data scientists, engineers, developers**: performing statistical analysis and machine learning training, implementing big data analytics to identify trends, solve business problems and optimize performance.

To match all these needs, data lakes need to have a variety of components, as shown in the picture below:



We want to be able to source anything and ingest the data without having to go through the ETL (Extract, Transform, Load) process that data warehouses require before storing it. A very important piece of the puzzle is the **catalog**, meant to describe (in some way) the content of our data containers, available for consultation via the **discover** tool. On top of this data lake architecture, we may want to perform **stream processing**, as in continuously inspect the data flowing into the lake; **batch processing**, typically to have a synthesis of a subset (batch) of the data; **machine learning and analytics**. As a common background we have a data governance and orchestration layer through which we can manage disaster recovery, backups, authorization and have some automated procedures to be ran on data.

One of the most widely used architectures for these type of workloads is the one proposed by Amazon Web Services (AWS) shown in the picture below:



As we can see, everything is built around Amazon S3 (Simple Storage Service, an object storage) but it is not the only option: AWS, in fact, allows to have different data sources, like

relational and NoSQL DB instances. NoSQL (Not Only SQL) is an evolution of relational databases that rejects the "schema on write" approach, allowing the storage of less structured data that can be retrieved with operations different from standard SQL queries.

We now need to tackle the problem of **data ingestion** (the first two rectangles in the data lake architecture). We first start from data sources, which can be either internal or external: **internal sources** may include relational databases, data warehouses and file system data, while **external sources** indicate data that can be obtained via APIs or web crawlers. Once the data is ready, we may go down the **hot** or **cold path** for data ingestion: the hot path is reserved for data that usually has time constraints and that reaches the system as soon as it has been generated, while the cold path, instead, moves sets of data at a time. To further add to this, we define three types of **data ingestion options**:

- **Data workload migration**: data is bunched up and migrated to a different system.
- **Incremental ingestion**: after performing a migration, we apply the changes that have occurred on the migrated data.
- **Streaming data ingestion**: we accept data as soon as it has been produced.

An important concept for streaming data ingestion is **change data capture**, necessary to ingest data logs and application events as soon as they are available, for continuous ingestion in the data lake, capturing streaming data changes and to migrate databases to the cloud. This can be done by means of high-level tools or programs which operate between the data source and the cloud data lake.

When performing **data ingestion**, we should consider the following **best practices**:

- **Identify the business case**: what is our goal and what we want to obtain.
- **Identify the right method of ingestion** (favoring the simplest solution): do we need to migrate from time to time? Do we have time constraints?
- **Consider streaming and change data capture ingestion benefits** but use them only if strictly necessary.
- **Focus on near-term needs**: do not over-engineer the solution, data lake and ingestion tools can be set up quickly and are flexible enough to allow for changes down the line.
- **Compress the data before sending it**, to minimize bandwidth requirements.
- **Encrypt Personally Identifiable Information (PII)**, to avoid violating privacy and data protection regulations.
- **Minimize the number of files**, as they are a source of complexity.
- **Ensure exact processing**, taking into account all the possible sources of inconsistency.
- **Automate ingestion**.

It is now time to focus on the **data storage** (storage and catalog components). A base idea could be **partitioning data in different zones**:

- **Raw**, an **immutable store** that cannot/should not be modified after it has been written, typically self-descriptive by means of attached metadata and useful for disaster recovery

(although in this case, we must ensure that this part is isolated from the rest of the system).
- **Optimized**, a subset of the data, possibly partially aggregated, to speed up data access and query execution (it is the idea behind materialized views in databases).
- **Analytics**, BI and machine learning-ready data and tables (e.g. after feature engineering).

To further optimize data access, we should pay particular attention to the file formats in which we store data. In addition to the more classic and well-known formats like CSV (Comma Separated Values) and JSON (JavaScript Object Notation), other types are available, some choosing to **store data by columns**, instead of by rows. Choosing a **columnar storage** may be beneficial:

- **Homogeneous data is kept in a single block**.
- **Strategies can be used to compress data within the block**.
- **It reduces fragmentation**, compared to row-based storage.
- **Mining algorithms may perform better**, as they often consider columns rather than rows.

Despite this, we need to keep in mind that **insertions are slower** in these systems (we need to decide based on our workloads) and they are **best suited for machine learning workloads** while having poor performances in OLTP (transaction and row-based).

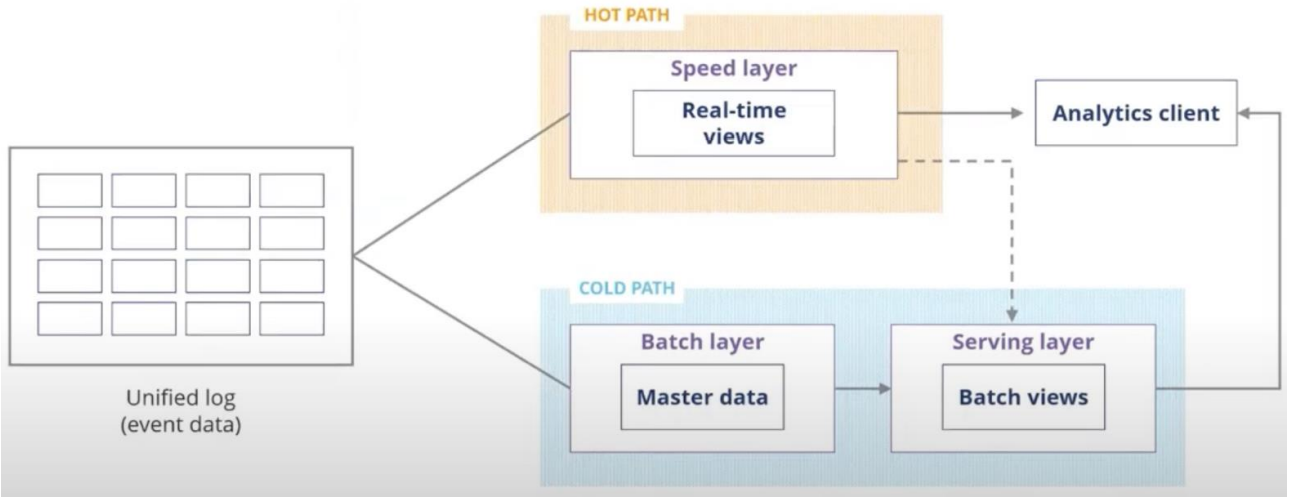
As we said before, having a **data catalog** is of utmost importance: the lack of a schema or descriptive metadata would in fact make it very hard to consume or query the data, while the lack of **semantic consistency** would impair pattern analysis and mining. To help us in defining the catalog, we can use **spiders** or **crawlers** that will make sure our data lake does not turn into a **data swamp**, a dumping area where no useful analysis is possible.

As a last note, we may perform cost optimization choosing to save important information in the so-called **hot storage**, which guarantees higher throughput and lower latency at increased costs, while storing less crucial information in the **cold storage**.

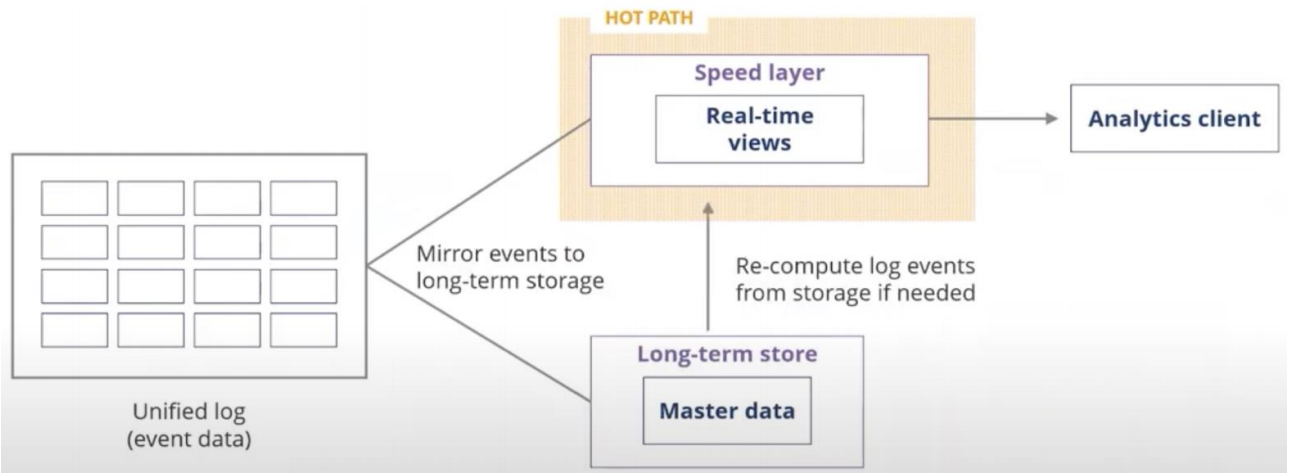
It is now time to focus on the **analytics part**. We have different types of analytics: **interactive analytics**, **big data analytics** and **real-time analytics**. With **interactive analytics** we refer to the ones in which a user issues interactive queries to large data volumes, expecting a result in a timely fashion, possibly saving back the result in the data lake. Typical users include businesses, business analysts, data scientists and developers. **Big data analytics** are interactive queries issued by data scientists and developers on large data volumes either for data aggregation, transformation or complex analytics. **Real time analytics**, instead, are used in streaming environments or in event detection, typically by data scientists and developers.

To conclude, we will now analyze three types of data lakes: the **lambda lake**, the **kappa lake** and the **delta lake**.

The **lambda architecture** was created to analyze data in different ways: the events coming from the data source, in fact, are fed to two separate pipelines, the hot and the cold path, respectively. In the hot path we perform quick computations and immediately send the results to the analytics clients, while in the cold path we set up buffers to allow for more complex and time-consuming batch processing. An example of lambda architecture is shown in the following figure:



The **kappa architecture** simplifies the previous architecture by replacing the cold path with a long-term store that allows the re-computing of log events if needed. These systems rely heavily on stream processing and offer high throughput and robust scalability. A view of a kappa architecture is shown in the picture below:



The **delta** lake, the last architecture we consider, offers a larger number of features, like:

- **ACID properties** (it is able to deal with consistency requirements).
- **Scalable metadata handling**.
- **Data versioning** (also called **time travel**, it allows the analysis of different snapshots).
- **Unified batch and streaming source and sink**.
- **Schema enforcement and evolution**.
- **DBMS-like operations**: updates, deletes, inserts, upserts (insert, on conflict update).

04 – The CRISP-DM methodology

We start from a question: can data mining be a "push-button" technology in which I get my answers just by pushing a button? The answer, unfortunately, is negative: **data mining is a process** that **involves various steps and complex choices**. It requires a mix of good tools and skilled analysts, a sound methodology (directions on how to carry out this process), project management techniques and a way to manage interactions along the process. By having a **standard process model for data mining**, we could have:

- A common reference point for discussions.
- A common understanding between the designers and the customers.
- A basis for **good engineering practice**.
- Checklists.
- Clear expectations.

The **CRISP-DM (CRoss Industry Standard Process for Data Mining) methodology** is a set of guidelines that describe common approaches used by data mining experts. It presents a **reference model**, consisting of **several phases**, each made up of **generic and specialized tasks** that materialize in **process instances**. It also provides a **user guide** with:

- **Checklists**.
- **Questionnaires**.
- **Tools and techniques** (mainly for machine learning).
- **A sequence of steps to take**.
- **Decision points**.
- **Pitfalls** (possible sources of mistakes).

The CRISP-DM methodology can be represented with a circle highlighting how, like every design process, it does not really "end": additional information and experience gained in a particular phase may give us a clearer idea about something that has been already dealt with previously, forcing us to go back and perform some adjustments. In addition to this, after the final step, consisting of the deployment phase, new requirements may arise and a new project could be started.

The first step of the CRISP-DM methodology is called **business understanding**. In this phase, data mining specialists need to have a close interaction with the business owners in order to try and understand what needs to be done with the data, **iteratively reformulating the problem in many different ways, refining the scenario** (what is the use case of the solution to the problem) **and our objectives**. The **tasks** that are involved in this phase are:

- **Determining:**
    - **Business objectives**.
    - **The background from which business objectives arise**.

    - **The Business Success Criteria (BSI)**, also called **Key Performance Indicators (KPI)**. In the beginning we should figure out what the client is expecting and have indicators that can quantitatively determine in what level they are met.
- **Assessing:**
    - **The resources available** (at a high level).
    - **The requirements, assumptions and constraints**.
    - **The risks and contingencies** that may affect our project and the **terminology**.
    - **Costs and benefits**.
- **Determining goals:**
    - **Data mining goals**.
    - **Data mining success criteria** (e.g. reaching a 95% accuracy).
- **Producing plans:**
    - **Project plan**.
    - **Initial assessment of tools and techniques to be used**.

In short, the business understanding step consists in generating reports of our current situation and objectives, while also planning on how and with what tools we want to achieve them. At the end of this phase, we will know which direction to take, what our target is, and the customer must be convinced of the value that this project will bring them.

The step that follows is called **data understanding**, in which we ask questions like:

- **What raw data is available?** It rarely matches the problem needs, as it has usually been collected for different purposes (or for no purpose at all). We may have multiple data sources, partially intersecting and with varying degrees of reliability due to the different way in which that data has been generated.
- **What is the cost of the available data?** Internal data may be free but external data typically is not. We may also need to collect further information by means of ad-hoc campaigns.

There may be instances where only after collecting and analyzing the data we can choose the direction of the project. The **tasks** to be completed during the **data understanding** process are:

- **Collecting the initial data**.
- **Describing the data**.
- **Exploring the data**.
- **Verifying the data quality**.

After each of these steps we should create reports, namely **initial data collection report**, **data description report**, **data exploration report**, **data quality report**.

The third phase is called **data preparation**: it is very important, since the results will heavily depend on how we perform this step. Some analysis techniques may require **data transformations** (like converting to tabular format, or from numeric to symbolic data types

and vice versa) or they could help **improve the quality of the results** (normalization, scaling, data cleaning and filling in missing data). We must also avoid having **data leaks**, where, during our mining process, we access data about our mining target that could invalidate our model, leading to conclusions akin to "it rains when it rains". The **tasks** involved in this phase are very expensive and time consuming (up to 80% of the total effort) and are:

- **Data set description**.
- **Data set selection**, explaining the rationale for including or excluding them.
- **Data cleaning**, generating a report on this step.
- **Data construction**, reporting which attributes have been derived or generated.
- **Data integration**, reporting what data has been merged.
- **Data formatting**, reporting what data has been reformatted.

It is important to stress that every step of the data preparation process should be clearly described and motivated, as to be reproducible.

The fourth step, **data modeling** consists in the application of machine learning techniques to bring to light patterns hidden within the data. The **tasks** associated with this phase are:

- **Modeling technique selection**: explicitly stating the modeling technique that we chose and its related assumptions (the limits, hypothesis on the data, etc.).
- **Generating the test design**.
- **Building the model**, setting the various parameters, doing the actual modeling, commenting and reporting on the models and the tools used.
- **Assessing the model**, performing the tests and checking for its validity, possibly revising the parameter settings.

After the data modeling process, we have the **evaluation** step, in which we **rigorously assess the results of the data mining process**, possibly **comparing different choices on a qualitative and quantitative basis**. We are also expected to **evaluate the confidence of the derived models**, **estimating the impact of wrong decisions on the business**. While the assessments of the previous phase were more related to the machine learning part, these evaluations are meant to judge the results by the business point of view. The **tasks** associated with the **evaluation** phase are:

- **Result evaluation**, assessing the data mining results with relation to the KPIs and typically approving the best performing model.
- **Process review**.
- **Determining the list of possible future actions and decisions**.

The last step of the CRISP-DM methodology is the **deployment**, in which the results of the data mining process (i.e. the models) are used "in production" to obtain the desired results and the return on investment. The **tasks** for this phase are:

- **Creating a deployment plan**.
- **Planning monitoring and maintenance**.
- **Producing a final report on the project**.
- **Reviewing the project and creating documentation on the experience gained**.

05 – Location intelligence

We start by presenting two definitions of **location intelligence (LI)**: one by Wikipedia stating that *"location intelligence (LI), or spatial intelligence, is the process of deriving meaningful insight from geospatial data relationships to solve a particular problem"* and one of our own, describing location intelligence as the **extension of traditional Business Intelligence systems with the spatial dimension** (as in, we add a map visualization to the information already provided by BI systems).

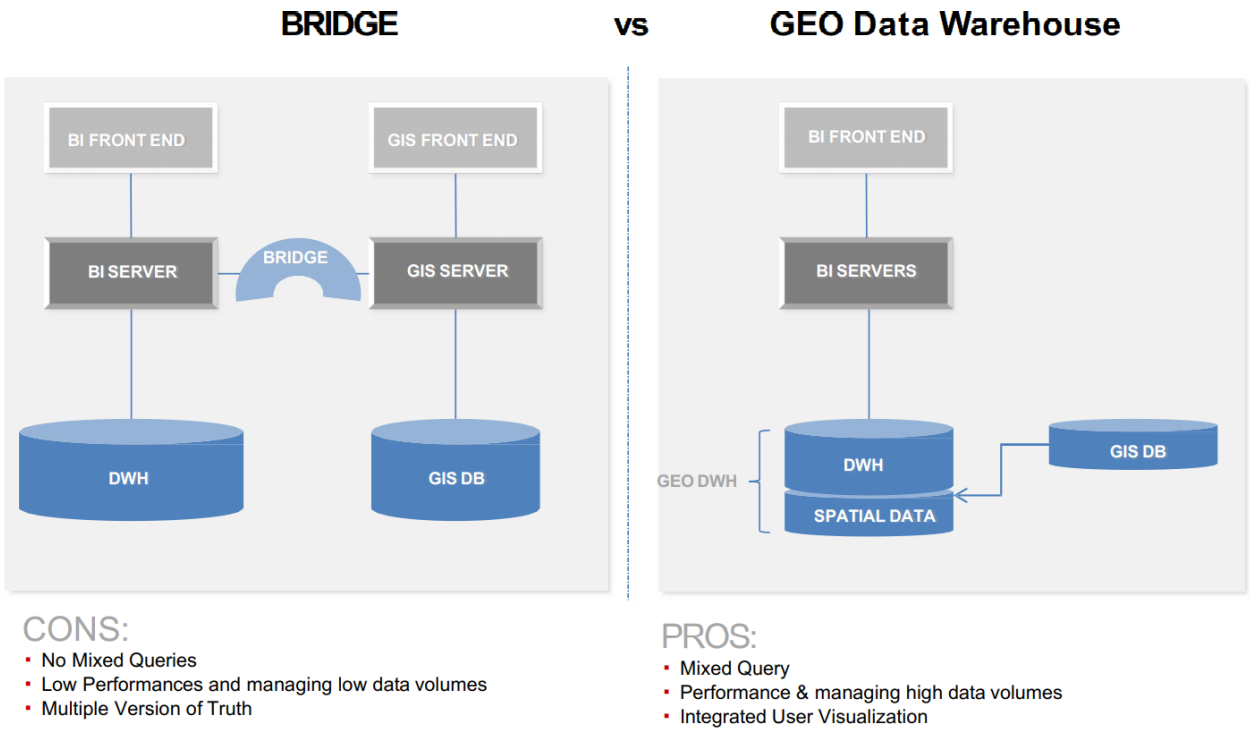
We have already seen that in business intelligence we are dealing with multiple data sources, often containing **different and ambiguous glossaries**, **non-conformed master data** and **multiple versions of "the truth"** that must be settled within the ETL (Extract, Transform, Load) process that builds the data warehouse. It is in fact said that one of the key success factors is having **conformed dimensions** (dimensions that have the same meaning across all the systems).

In the last few years, more and more data has been published as part of **open data** initiatives, often dealing with demographics, income, weather, crime, etc. We can clearly see that these datasets contain spatial components, making them part of the **linked open data** category. This is crucial in the big data era since, due to the profound differences between data sets, it is often difficult to find conformed dimensions to use in the ETL process. It is only after integrating and correlating data, in fact, that we can unlock its complete potential, both in terms of analysis and monetary value.

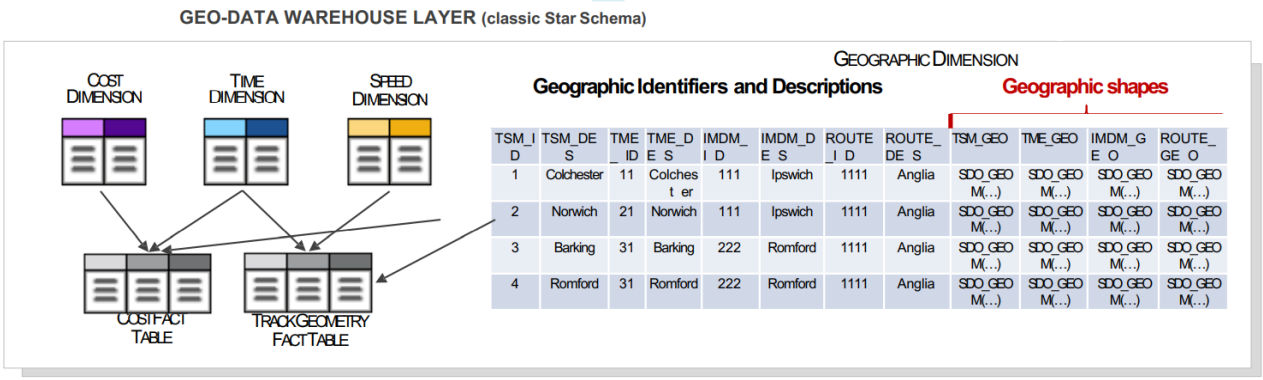
When dealing with geographical data, we need to be aware of a few aspects:

- **The same point could be described by means of different notations**, for example:
  - **Latitude and longitude** (e.g. 44.4887653 N, 11.3299231 E).
  - **Spherical Mercator Projection** (e.g. 1.685.290, 4.928.818).
  - **Street address, ZIP code and city** (e.g. Viale del Risorgimento 2, 40136 Bologna). It is important to note that a postal address can be converted into a spatial representation by means of **geocoding**.
- **The shortest path between two points may not be a line**: especially on longer distances, in fact, we need to take into account the curvature of the Earth.

To deal with location intelligence, we need an architecture that supports it. This can be done by means of a **bridge** or a **GEO Data Warehouse**, as shown in the picture below:



In the **bridge** architecture, we have two systems: one dealing with the symbolic data (the data warehouse) and a **geographical information system database (GIS DB)**, the former containing our representation of the facts of the real world and the latter containing information about the places. The two systems are then connected via a bridge that enables both frontends to have access to all the information. With this approach, though, **we cannot perform mixed queries** (we can only query one system and ask the bridge to link the result dataset with the other data), **we have low performance** and **multiple versions of truth**. On the contrary, a **GEO data warehouse** enables mixed queries and has high performances, but with a higher cost. An example of a GEO data warehouse can be found in the following picture:



A list of the differences between various solutions for location intelligence is presented in the table that follows:

	Catalogue	Bridge	Geo DWH
Effective KPI representation on a map	YES	YES	YES
Complex KPIs on a map (e.g. pie charts)	NO	NO/YES	YES
Integration with GIS maps	NO	YES	YES
GIS-like tools to interact with maps	NO	YES	YES
Map and BI report can interact	NO	YES	YES
Map and BI report can interact in an **integrated** way	NO	NO	YES
Hybrid queries	NO	NO/YES	YES
Integrated Security	NO	NO	YES
Scalability e Performance	NO	NO	YES

Once our location intelligence solution is set up and running, we will be able to **develop advanced, territory-centric marketing strategies or analyses** and **comprehend better complex phenomena by leveraging geographical information**. Examples of these new possibilities include:

- **Geographical navigation of areas of interest**, choosing them in terms of proximity and correlation.
- **Integration of internal and external data sources**.
- **Run-time mixed queries**, combining analytical (alphanumerical) and geospatial criteria.
- **Predictive analysis** and **GEO-what-if analysis**: predictive analysis (site location, site distribution, sales territory potential, …).
- **Spatial data mining**: points of interest and co-location, forecasting areas of high potential, etc.