TECHNOLOGIES FOR INFORMATION SYSTEMS

Luca Gerin

POLITECNICO DI MILANO 2022/2023

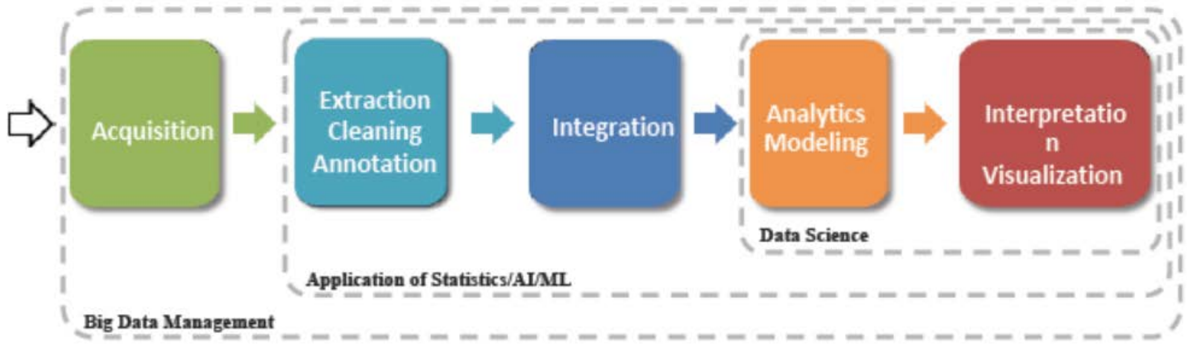
Index

1. Introduction to data integration 2
2. Structured Data Integration 5
    - Record Linkage 6
    - Data Fusion 7
    - Data integration in the multi-database 7
3. Semi-structured Data Integration 8
    - Ontologies 10
4. Data Quality 14
5. Data warehouses 17
6. Data warehouse design 19
    - DW Conceptual design 19
    - DW Logical design 22
7. Other architectures for Big Data 25
8. Frontiers in Data Integration 27
9. Ethics in Data Science 31

Questions from past Exams 32
- Data Integration questions 32
- Data Warehouses questions 36
- Data quality questions 38
- NoSQL questions 39

Multiple choice questions 40

1. Introduction to data integration


Acquisition
Extraction Cleaning Annotation
Integration
Analytics Modeling
Interpretation Visualization
Data Science
Application of Statistics/AI/ML
Big Data Management

The objective of **Data Integration** is to combine data coming from different data sources, providing the user with a unified vision for the data.

This involves detecting correspondences between similar concepts that come from different sources, and conflict solving.

The main issues in data integration come from the properties of Big data, summarized in the **four V**'s of Big data:

- Volume: each data source contains a huge quantity of data, and the number of the data sources themselves can be high and in continuous growth.
- Velocity: inside each data source, data is in continuous change, because some is added after a new collection, some deleted and some modified.
- Variety: data sources are extremely heterogeneous in terms of both schema and instance level.
- Veracity: data sources and their data are of different quality degree, with difference in completeness, validity, consistency, timeliness, accuracy and ethics dimensions and fairness.

Variety in particular can be about different things:

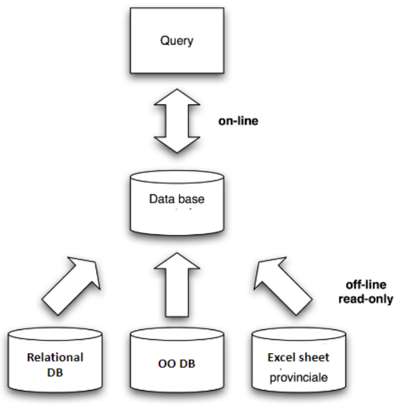
- different platform used,
- different data models,
- a different query language,
- different schemas: Schema (semantic) heterogeneity,
- different values for the same information: instance (semantic) heterogeneity.

There are three main steps in Data Integration:

1. **Schema reconciliation**: mapping the data structure
2. **Record linkage** (or Entity resolution): data matching based on the same content
3. **Data fusion**: reconciliation of non-identical content

There are 2 main ways of integrating Database systems:

- Using a **materialized database**: data from sources are extracted, transformed and loaded (ETL) in to a new database in which they are merged
- Using a **virtual** non-materialized **database**: data remains in the sources which are accessed every time the data is needed



When using a materialized database, the sources are reconciled, the data is put in the same database and then the needed queries are posed on that result database.

These large databases are **data warehouses**, and the data is inserted through ETL periodically, to build a history of the enterprise and use to for data analysis and mining.

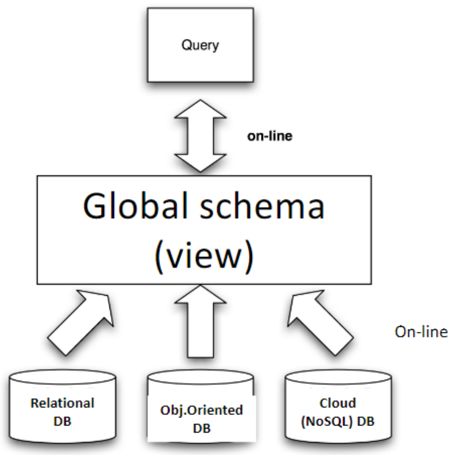
The information contained in a materialized database, for the nature of how it is collected, cannot be up-to-date data.

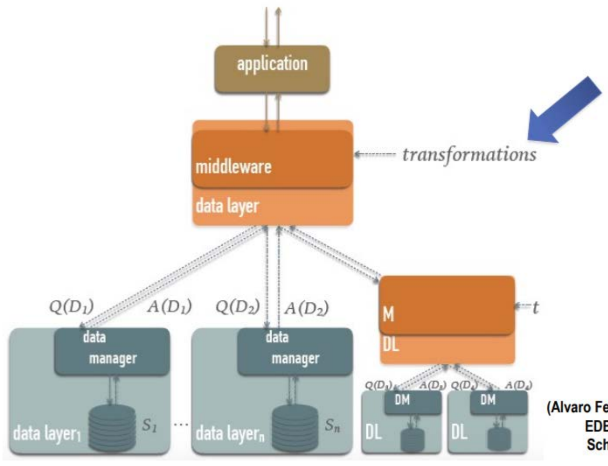
So, in case of need of an integrated data collection with always up-to-data data, a virtual database is a better choice.

When using a virtual database, a **global schema** is built, containing all attributes of interest extracted from the sources.

When querying, a query is posed on the global schema, then translated in multiple queries to send to the data sources. The results of these single queries to the data sources are then put together and integrated to produce a single response to return as result of the original query posed on the global schema.

Because the data to answer queries are always taken directly from the sources, the virtual approach will always return a fresh answer to the query.





(Alvaro Fernandez, EDBT Summ. School 2017)

In any case, to pursue the objective of querying more data sources as if they were only one, the data integration system acts like a middleware between the application posing queries and the data sources.

Note that one source for a data integration system can be another data integration system hiding its sources.

Data integration has many issues to solve, given by heterogeneity of sources and their data. This can also happen even if there is only one unique centralized DB acting as single source for the integration.

For example, a DB of a company can be detached in different areas of the company, each one asking for their part of the DB. But a lot of the data used in each part is common to their different applications, so, redundancies and different representations of the same data can appear. Moreover, when updating one instance of these duplicates, maybe the other one will remain as before, creating inconsistencies.

The solution to this problem would be the creation of different personalized views to answer the different needs of the parts of the company, as well as granting access control, but each view accesses the same integrated and unique schema of the centralized DB.

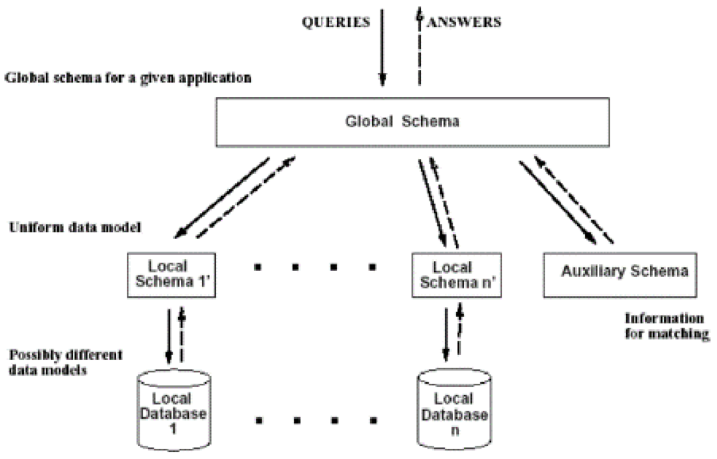
When talking about multiple sources, the virtual approach to data integration has two variants:

- Data exchange (virtual): appropriate when there are only two systems to integrate, consist of the creation of gateways between them
- Multi-database (virtual): creation of a global schema

The **Multi-database** virtual approach supports the access to different data sources, via a unifying global schema, receiving queries written in the language of the global schema and distributing rewritten queries to the sources, in order to then combine the answers to build the final answer.

The global schema provides a reconciled, integrated and virtual view of the data sources.

An auxiliary schema may be employed, containing info about the views and relationships between the global schema and source schemas. This information is useful for the middleware and utilized by it.



The process of designing data integration in the multi-database is the following:

1. Source schema identification (when present)
2. Source schema reverse engineering (data source conceptual schemata): the conceptual schemas are built from the source schema
3. Conceptual schemata integration and restructuring: conflict resolution and restructuring so to build the global schema
4. Conceptual to logical translation (of the obtained global schema)
5. Mapping between the global logical schema and the single schemata (logical view definition)
6. After integration: query-answering through data views

When coming to conflict resolution and restructuring, the steps to follow are:

a. Related concept identification: done manually or automatically, with humans intervening in complex matchings.
b. Conflict analysis and resolution
c. Conceptual Schema integration

The type of conflicts that it is possible to find are:

- Name conflicts:
    - two attributes or tables have the same name but different meaning
    - two attributes or table have different name but the same meaning
- Type conflicts: different types of data are used to represent the same information in different DBs
- Data semantics conflicts: for example, two schemas could have data referring to different conventions such as different currencies, measure systems, granularities
- Structure conflicts: if there is a difference in the structure of relationships between elements
- Dependency (or cardinality) conflicts
- Key conflicts: two elements representing the same thing have different primary keys to identify them

2. Structured Data Integration

A data integration system is a tuple (G, S, M) where:

- G is the Global schema
- S is the set of sources
- M is the set of mappings

In order for the system to work, the system must understand which real data in the data sources corresponds to the virtual data in the virtual database.

The mapping between the global logical(mediated) schema and the single source schemata happens through logical views definition.

There are two basic approaches:

- GAV: Global As View
- LAV: Local As View

Note that the same approach can be used also in case of different data models, and in that case also a model transformation is required.

The **Global As View** (GAV) approach is the one in which the global schema is expressed in terms of the data source schemata. The global schema is a big unique view on top of the data sources, formulated in terms of them. It's a good approach for stable data sources, but it is difficult to extend upon the addition of new data sources. In fact, when introducing a new source, the views present in the system need to be modified in order to account for the new source. This is a complex operation.

The most useful and used integration operators to write a relational GAV view are: Union, Outerunion, Join, Outerjoin, Generalization.

In the **Local As View** (LAV) approach, the global schema is designed independently of the data source schemata and the relationship (mapping) between sources and global schema is obtained by defining each data source as a view over the global schema. So, there is a never-modified global schema and, when a source schema is added or modified, the tables of the sources are defined as if they were views on the global schema. This way, only the views on the added or modified sources need to be written, and not the whole mapping. It's still a good approach if the global schema is stable but, contrarily to GAV, this approach favors extensibility and modularity.

However, query processing is much more complex, as the view definition utilized specifies the opposite transformation with relation to the one needed to answer queries on the global schema. So, in order to perform query processing, the system requires to do some reasoning. By reasoning, it builds a plan: a rewriting of the query as a set of queries to the sources and a prescription of how to combine their answers.

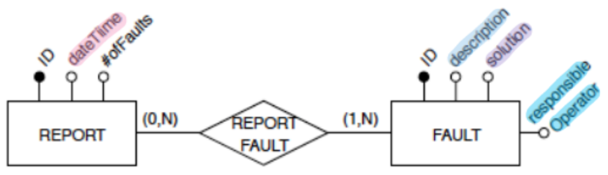
A hybrid of GAV and LAV is the **Global and Local As View** (GLAV) approach, in which the relationship (mapping) between sources and global schema is obtained by defining a set of views, some over the global schema and some over the data sources.

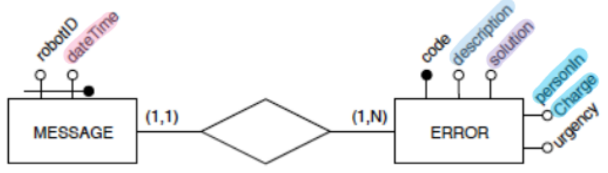
Once the schema reconciliation phase is concluded, either by using GAV or LAV, the Record linkage and successively the Data fusion phase happen.

Be there a schema or not, there may be inconsistencies in the data. Record linkage is about finding the info that refer to same real-world entities, while Data fusion is about reconcile inconsistent information, once recognized that two items refer to the same entity.

Record Linkage

During Record linkage, the objective is to recognize when two datasets contain the same information.





To perform this activity, the notion of similarity is utilized.

Problem: given two sets of strings, find all pairs from the two sets that refer to the same real-world entity. Each of these pairs is called a match.

In order to find matches, it is possible to use a <u>similarity measure</u>.

The main types of similarity measures are:

- Sequence-based
- Set-based
- Hybrid
- Phonetic

The *Edit* (or *Levenshtein*) *Distance* is based on the count of the minimal number of operations that are needed to transform a string a into a string b. Such operations are character insertion, character deletion, character replacement. Once computed the distance, the similarity is:

$$s(a,b) = 1 - d(a,b) \,/\, [max(length(a), length(b))]$$

Set-based similarity measures are based on the fact that strings are multisets of tokens (pieces of the strings)

The *Jaccard measure* is used to see how similar two sets are, and can be applied to strings seen as sets of tokens.

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|} = \frac{|A \cap B|}{|A| + |B| - |A \cap B|}$$

Phonetic similarity measures match strings based on their sound. The most used is *Soundex*, which calculates a four-character code from a word based on the pronunciation and considers two words as similar if their codes are equal. This means is strongly based on the language, and is then very good for some applications but very bad for others.

Applying a similarity measure to all pairs of data is not efficient: it is quadratic in the size of datasets. Many solutions have been proposed to choose the pairs of words that most probably match.

Record matching is about finding out when two tuples match, so when they refer to the same entity.

There are three types of record matching:

- Rule-based matching: manually written rules that specify when two tuples match, it requires a lot of time and effort.
- Learning-based matching
  - Supervised: rules for matching are learned from training data (a lot of training data is required)
  - Unsupervised: matching is typically performed via clustering, based on clustering similar values
- Probabilistic: the matching domain is modeled using a probability distribution, and the matching decisions are made based on some reasoning on the distribution. This approach can naturally incorporate a variety of domain knowledge and provide a frame of reference, but is computationally expensive and often hard to understand.

Data Fusion

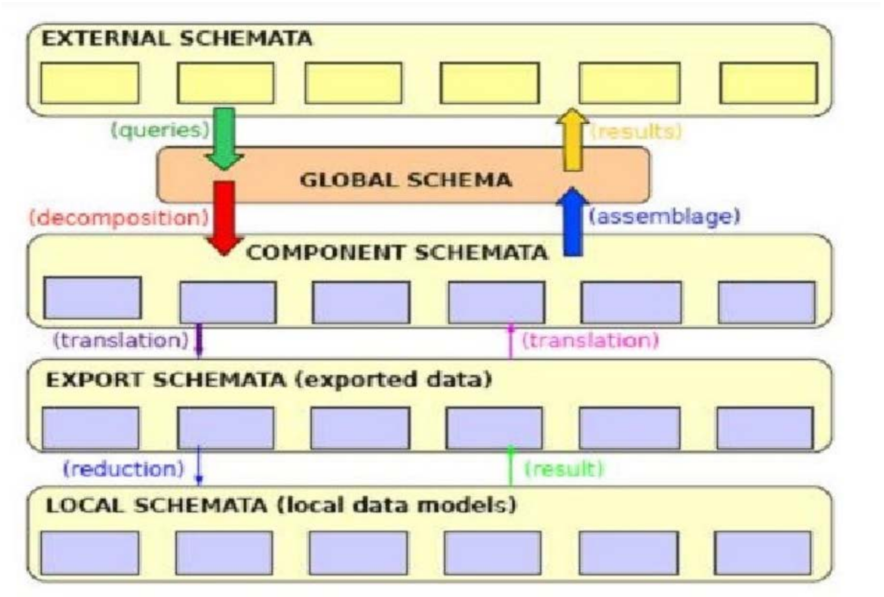
Once understood that some data represent the same entity (Record Linkage), the data needs to be put together.

There may be inconsistencies, that may depend on different reasons: One source is incorrect, or in each database there is only a partial view of the concepts to represent.

The way to integrate these data is to use a **resolution function**, that is a function of the values found for an attribute in two different schemas. The decision is on how to put data together according to intended use and domain.

Data integration in the multi-database

Recalling that sources can be heterogeneous in schemata, data models and systems, the data integration multi-database has the following structure:



The external schemata contain personalized views of the DB.

From here, queries are formulated on the Global schema.

These queries are decomposed in more queries, and translated for the export schemata (representation of sources), in order to be sent to single data sources.

The local data models are interrogated and produce a result.

This result goes up the chain: it is first translated into the language desired for the multi-database, and then assembled into one result, as if it was produced by the global schema.

Then it is delivered to the application making the query.

The translation between the language of the data integration system and the one of the sources is a task done by **wrappers**. They convert queries into queries/commands which are understandable for the specific data source they are associated to. Then, they convert query results from the source format to a format which is understandable for the application.

A wrapper is easy to produce if the data in the source is structured, but more difficult is the problem with semi-structured data.

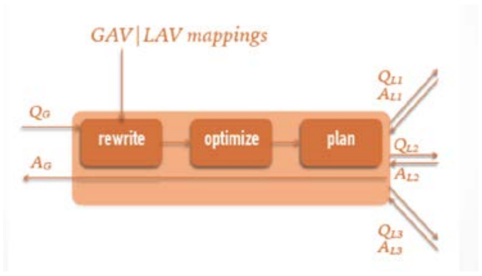
Design steps:

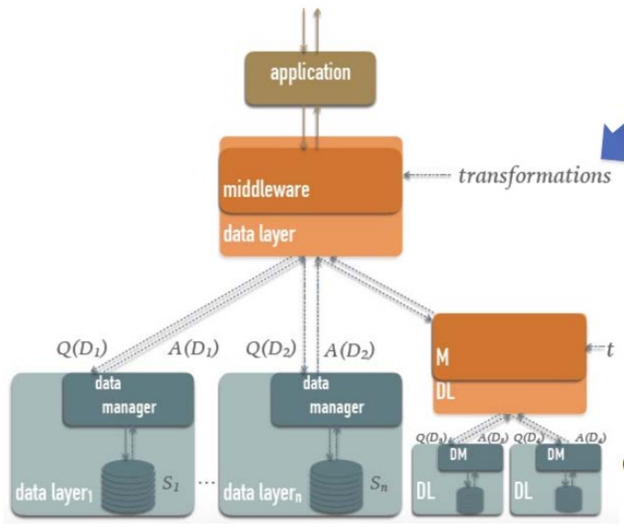
1. Reverse engineering (i.e. production of the conceptual schema)
2. Conceptual schemata integration
3. Choice of the target logical data model and translation of the global conceptual schema
4. Definition of the language translation (wrapping)
5. Definition of the data views (as usual)

3. Semi-structured Data Integration

Semi-structured data has some form of structure. But it is not as prescriptive, regular or complete as in traditional data base management systems. Examples might be web data, xml data, but also data derived from the integration of heterogeneous data sources.

The framework for data integration is the same as the one for structured data, case in which the middleware integration system works as follows:





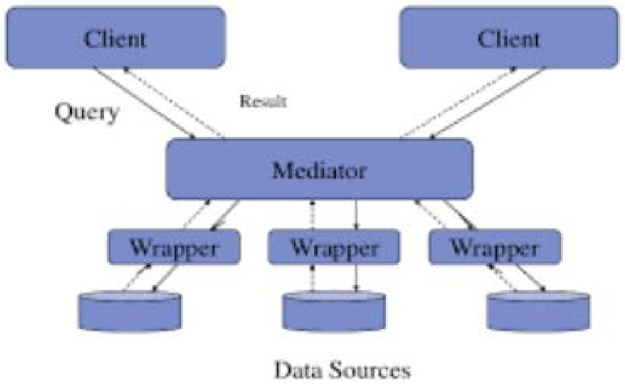
(Alvaro Fernandez, EDBT Summ. School 2017)

A query entering the middleware is re-written using the mappings, optimized and then a plan phase takes place, before the generated queries to the sources are sent and the returned results are combined and given to the application. Of course, in order to re-write a query using GAV or LAV mappings, it's necessary that data has a structure and is well-structured. Therefore, GAV and LAV are no more sufficient for semi-structured data.

Semi-structured data can have different form and degree of structure. It can be a website organized with no apparent criteria, or a website with pages generated starting from an underlying database, with a regular data structure from which data are extracted to be put in the page (usually for data-intensive websites), or an e-commerce site. The point is that semi-structured data can be of many different kinds and quality.

There are different semi-structured data models, which do not lend themselves to easy integration. They can be based on text, trees or graphs.

In order to integrate and then query semi-structured data, it is needed to progressively build an overall data representation for them, as new information sources are explored.

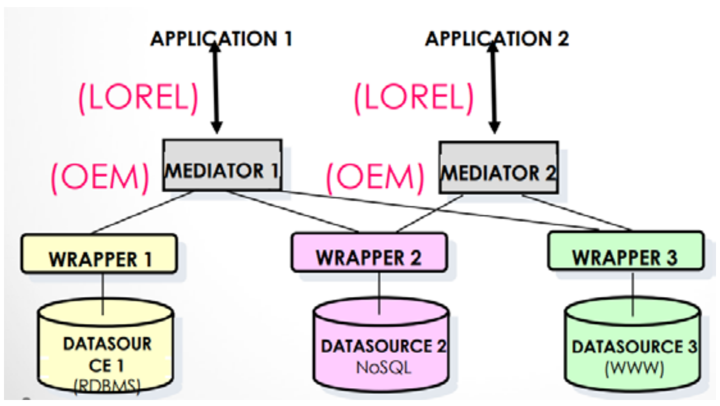


The tool to integrate semi-structured data is a **Mediator**, which is built by the designers of the integration system and must do the same as integration systems, but with more complex processes (starting from substituting GAV and LAV with other techniques).

A mediator must know what is inside and what language is utilized by the sources, in order to make the interfaces work. It must build and maintain the knowledge structures that drive the transformations needed to transform data to information. At last, it must include any intermediate storage that might be needed (Wiederhold).

These requirements make clear that each different domain needs a mediator appropriately designed to "understand" its semantics. Mediators are built ad-hoc, to contain some domain knowledge.

The mediator stores a unique internal data model.

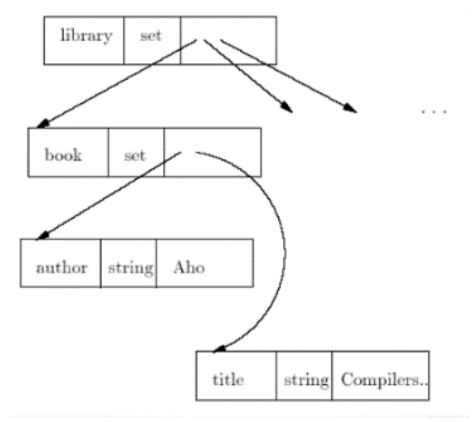


Object Exchange Model (**OEM**) is a language that can be used for this purpose. It is used by the mediator to manage an internal graph-based representation of data. It does not represent the schema of data, but directly represents the data: it is self-descriptive.

The object structure in OEM is set and is a nested structure composed by:

$$< (Object - id), label, type, value >$$

With OEM is possible to build complex data structures.



Being specialized into a certain domain, each mediator must know domain metadata, which convey the data semantics. It must be able to perform on-line duplicate recognition, reconciliation and removal.

An application poses queries to the mediator using a different language that the mediator must understand. One of these languages is **LOREL** (**L**ightweight **O**bject **Re**pository **L**anguage), which is an object based language, similar to oriented query languages, but with some modifications appropriate for semi-structured data.

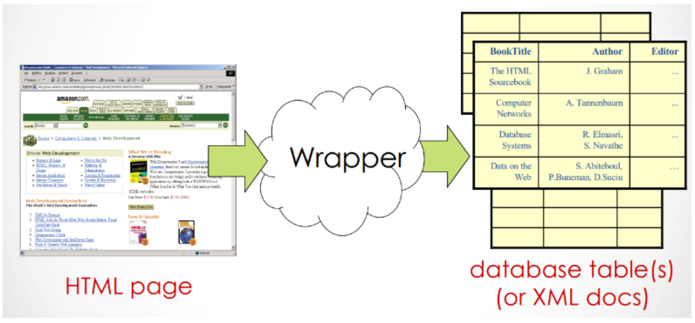
```
select library.book.title
where library.book.author = “Aho”
from library
```

In order to write a query in LOREL, a structure of data must be known. This structure is provided by OEM that is in the mediator between the application posing queries in LOREL and the sources with their wrappers.

The schema to use for the query is built progressively a-posteriori by the mediator while exploring the data sources. Such schema is called **Dataguide**.

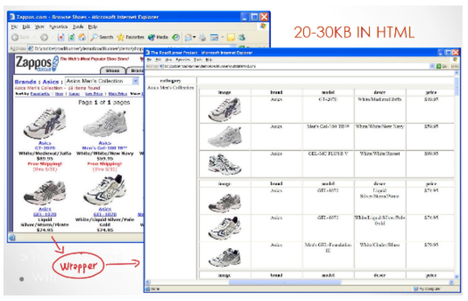
Between the mediator and the data sources, there are wrappers, which convert queries into queries/commands which are understandable for the specific data source and query results from the source’s format back to a format which is understandable for the application.

A wrapper must be able to convert data from irregular sources, like a web page, into something that is more regular.



BookTitle	Author	Editor
The HTML Sourcebook	J. Graham	...
Computer Networks	A. Tannenbaum	...
Database Systems	R. Elmasri, S. Navathe	...
Data on the Web	S. Abiteboul, P.Buneman, D.Suciu	...

To extract information from HTML docs, from a source format composed of plain text with HTML tags to a target format more structured like a relational table or XML, extraction rules that exploit the marking tags are employed by the wrapper.



If the data source changes, even a little, the wrapper of that source needs to be modified too. A web site layout change may affect the extraction rules.

Unfortunately, human-based maintenance of an ad-hoc wrapper is very expensive.

A possible, but partial, solution for this issue is <u>automatic wrapper generation</u>. This strategy can be used only when pages are regular to some extent, for example if many pages share the same structure or pages are dynamically generated from a DB (suitable for data intensive websites).

Ontologies

Ontologies are a way to solve the problem of automatic semantic matching.

An **ontology** encompasses a representation, formal naming, and definition of the categories, properties, and relations between the concepts, data, and entities that substantiate one, many, or all domains. An ontology is also a formal and shared definition of a vocabulary of terms and their inter-relationships (synonymy, homonymy, hyponymy, …), is a controlled vocabulary that describes objects and the relationships between them in a formal way.

An Entity-Relationship diagram, a class diagram, any conceptual schema is a kind of ontology.

An ontology is a formal specification of a conceptualization of a shared knowledge domain. It is a controlled vocabulary with its grammar that is used to express queries and assertions.

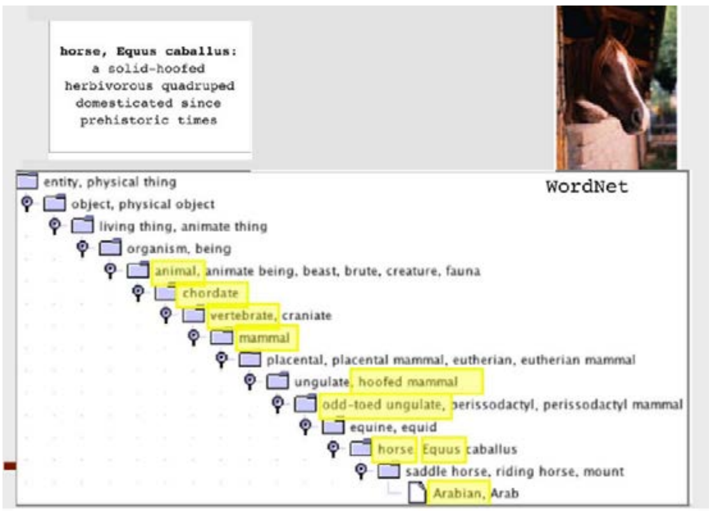
The aims of ontologies:

- A formal specification allows for use of a common vocabulary for automatic knowledge sharing
- Formally specifying a conceptualization means giving a unique meaning to the terms that define the knowledge about a given domain
- Shared: an ontology captures knowledge which is common, thus over which there is a consensus (objectivity is not an issue here)

There are two types of ontologies:

- **Taxonomic ontologies**: to provide a reference vocabulary, are the definition of concepts through terms, their hierarchical organization, and additional pre-defined relationships.
- **Descriptive ontologies** are the definition of concepts through data structures and their interrelationships. They provide information for "aligning" existing data structures or to design new, specialized ontologies (domain ontologies).

A taxonomic ontology is WordNet. An entity is defined via an "is-a" hierarchy representing specialization, while for each levels different synonyms are present.



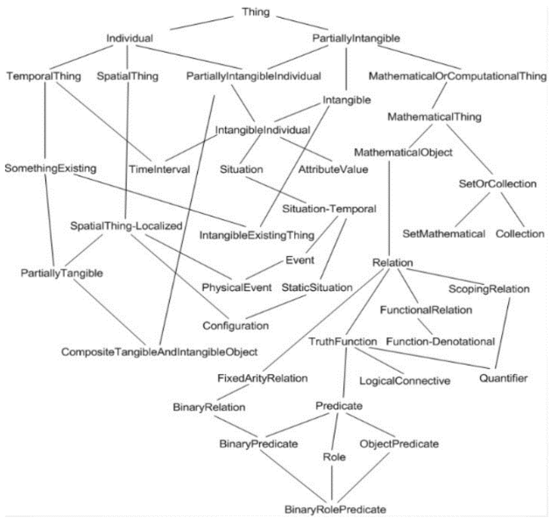
```
horse, Equus caballus:
a solid-hoofed
herbivorous quadruped
domesticated since
prehistoric times

WordNet

entity, physical thing
  object, physical object
    living thing, animate thing
      organism, being
        animal, animate being, beast, brute, creature, fauna
          chordate
            vertebrate, craniate
              mammal
                placental, placental mammal, eutherian, eutherian mammal
                  ungulate, hoofed mammal
                    odd-toed ungulate, perissodactyl, perissodactyl mammal
                      equine, equid
                        horse, Equus caballus
                          saddle horse, riding horse, mount
                            Arabian, Arab
```

An ontology is a conceptual representation of knowledge, consisting of:

- Concepts
  - Generic concepts, to express general world categories
  - Specific concepts, to describe a particular application domain (domain ontologies)
- Concept definition
  - Via a formal language
  - In natural language
- Relationships between concepts
  - Taxonomies (IS_A),
  - Metonymies (PART_OF),
  - Synonymies, homonymies, ...
  - User-defined associations

An ontology is (part of) a knowledge base, composed by:

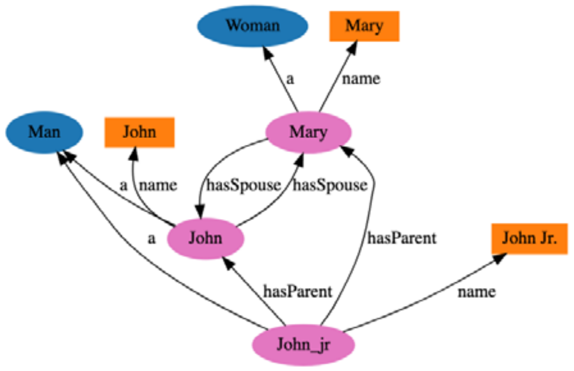
- a **T-Box**: contains all the concept and role definitions, and also contains all the axioms of our logical theory (e.g. “A father is a Man with a Child”).
- an **A-box**: contains all the basic assertions (also known as ground facts) of the logical theory. It describes the instances.
  (e.g. “Tom is a father” is represented as Father(Tom)).



OpenCyc is an ontology containing hundreds of thousands of terms, along with millions of assertions relating the terms to each other, forming an ontology whose domain aims to be all of human knowledge.

The **Semantic web** is a paradigm for the future, with the goal of making Internet data machine-readable, in order to allow machines to automatically process and integrate information available on the Web.

To enable the encoding of semantics with the data, technologies such as Resource Description Framework (RDF) and Web Ontology Language (OWL) are used to formally represent metadata.

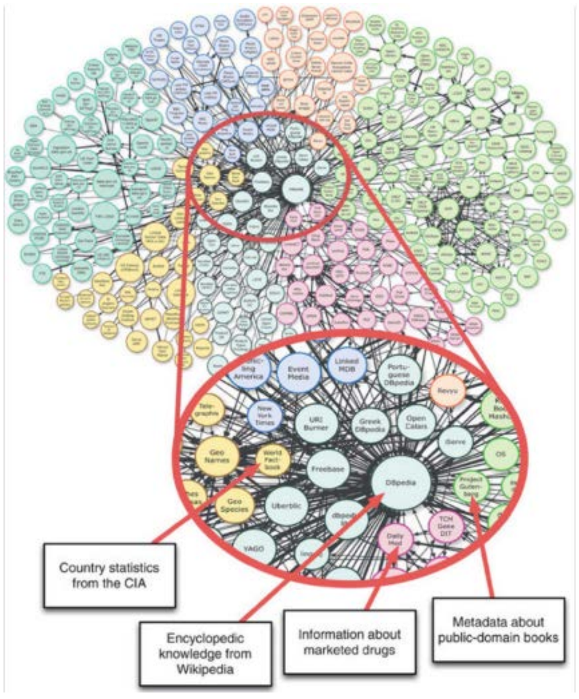


The **Resource Description Framework** (**RDF**) is a World Wide Web Consortium (W3C) standard originally designed as a data model for metadata, built on XML. It is based on the notion of a triple $(subject, predicate, object)$. The subject is a resource, represented as a node in the graph, while the predicate is an edge or a relationship leading to the object, which is another node or a literal value.

The first level above RDF is the **Web Ontology Language** (**OWL**), an ontology language, built on top of a family of knowledge representation languages such as RDF, that can formally describe the meaning of terminology used in Web documents.

OWL is designed for use by applications that need to process the content of information instead of just presenting information to humans. It has the objective of facilitatating greater machine interpretability.

OWL has three increasingly-expressive sublanguages: OWL Lite, OWL DL, and OWL Full.


Country statistics from the CIA
Encyclopedic knowledge from Wikipedia
Information about marketed drugs
Metadata about public-domain books
DBpedia
Freebase
Geo Names
YAGO
Uberblic

Linked data is a movement with the objective of connecting data sets across the web, in order to share information in a way that can be read automatically by computers. To achieve it, linked data describes a method of publishing structured data so it can be interlinked and become more useful. Linked Open data is the idea of publishing open data (data that is accessible to all) as RDF on the web and then to set RDF links among them.

From bottom to top:

- XML: provides a surface syntax for structured documents, but imposes no semantic constraints,
- XML Schema: a language for restricting the structure of XML,
- RDF: is a data model for objects and relations between them,
- RDF Schema: is a vocabulary for describing properties and classes of RDF resource,
- OWL: adds more vocabulary for describing properties and classes.

A main feature of the Semantic Web is to support reasoning services over a given set of data and knowledge. These are divided in:

- Services for the T-box: (Theory)
    - Subsumption: verifies if a concept C subsumes (is a sub-concept of) another concept D
    - Consistency: verifies that there exists at least one interpretation I which satisfies the given T-box
    - Local Satisfiability: verifies, for a given concept C, that there exists at least one interpretation in which C is true.
- Services for the A-box: (Assertions)
    - Consistency: verifies that an A-box is consistent with respect to a given T-box
    - Instance Checking: verifies if a given individual x belongs to a particular concept C
    - Instance Retrieval: returns the extension of a given concept C, that is, the set of individuals belonging to C.

The main difference between a model like the E-R model and ontologies is that E-R cannot be filled with instances, while ontology systems allow to store the definitions and also the data.

Descriptive ontologies require rich models to enable representations close to human perception.

	Ontology	DB
Complex data structures	No, cannot build concepts from other concepts	Yes
Generalization/specialization hierarchies	Yes	Yes
Defined concepts	Yes	No, only partially possible with views

In order to improve the database conceptual models to fulfill ontology requirements, they need to:

- Support defined concepts and the necessary reasoning mechanisms
- Be able to Manage missing and incomplete information.
  In a database, according to the Closed World Assumption, if something is not known, as it is not present in the database, then it surely is false (e.g. "has customer X paid? If there is no tuple, then no"), while in an ontology the Open World Assumption leaves the possibility of that something to be anything.
- Represent uncertain data

Ontologies can be used to discover equivalent concepts, to perform mapping in data integration problems. The notion of equivalence between concepts, is based on the notion of similarity also for what concerns ontologies.

Ontology matching is the process of finding pairs of resources coming from different ontologies which can be considered equal in meaning.

The similarity measure for ontologies takes into account semantics, and not only the structure of the words as similarity measures used for structured data.

Similarity is strictly related to distance. The distance between two identical object is zero, while the distance between different objects have a non-zero positive value.

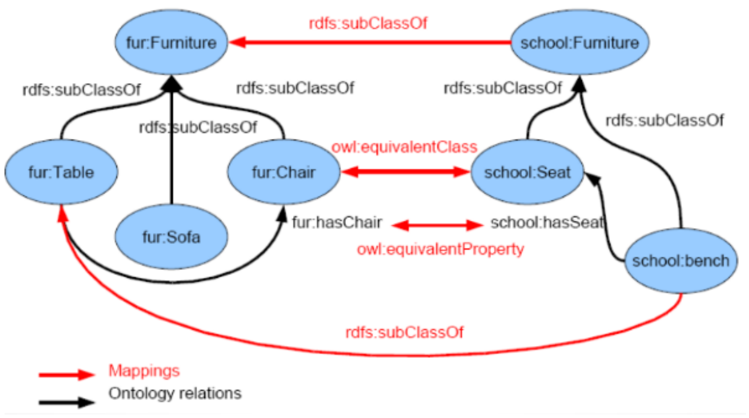
The triangle inequality says that the distance between y and z does not exceed the sum of the distance between y and x and the distance between x and z.

$$d(y, z) \leq d(y, x) + d(x, z)$$

While dealing with distance-based similarity measures, examples have been constructed where every distance axiom is clearly violated by dissimilarity measures and particularly the triangle inequality, consequently the corresponding similarity measure disobeys transitivity. For these cases a different attitude has been taken and more general concepts of distance have been proposed.

Ontology mapping is the process of relating similar concepts or relations of two or more information sources using equivalence relations or order relations.

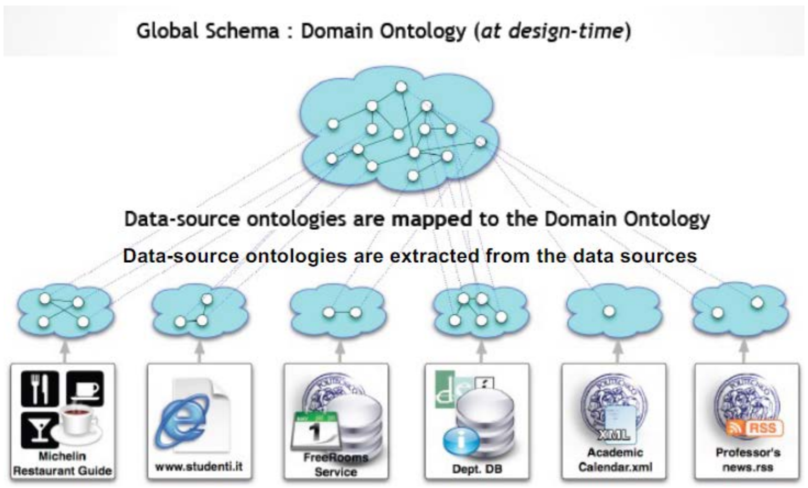
There might be ontology mismatches, given by the fact that ontologies are written in different languages. The solution is to normalize by translating to the same language or paradigm.



An ontology can be used as a schema integration support tool:

- to represent the semantics of schema elements
- to guide conflict resolution thanks to similarities between source ontologies

An ontology can also be used instead of a global schema, so to have a schema-level representation only in terms of ontologies and perform ontology mapping instead of schema integration, and then use the integrated ontology as a schema for querying.

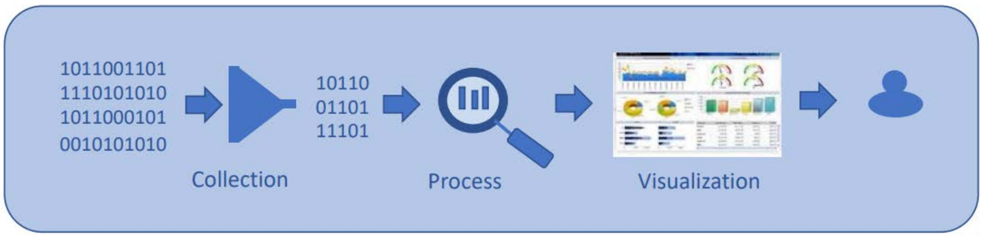

Global Schema : Domain Ontology (at design-time)
Data-source ontologies are mapped to the Domain Ontology
Data-source ontologies are extracted from the data sources
Michelin Restaurant Guide
www.studenti.it
FreeRooms Service
Dept. DB
XML
Academic Calendar.xml
RSS
Professor's news.rss

An ontology can function as a support tool for content interpretation and wrapping, and as a mediation support tool for content inconsistency detection and resolution (record linkage and data fusion).

Ontologies require query languages as well, for schema exploration, to reason on the schema, for instance querying. An example of ontology query language is *SPARQL* (W3C).

4. Data Quality

Data-driven Management is characterized by the practice of collecting data, analyzing it, and basing decisions on insights derived from the information.



The success of data-driven decision making depends on the quality of data collected and on the methods used to analyze data. Of course, if the data collected is of poor quality, then also the output decision is bad (GIGO: Garbage In – Garbage Out). Even a small error in data may sometimes cause almost catastrophic effects in the outcome of a decision process.

An adequate architecture for analyzing data is needed. So, when data is collected, also the Data preparation happens:

- Assessment
- Cleaning
- Transformation
- Entity resolution
- De-duplication
- Imputation
- Natural language processing
- Quality management

Real-word data is often incomplete, inconsistent, and contain many errors, so data preparation is really important.

Data preparation, cleaning, and transformation comprises the majority of the work in a data mining application (90%).

Data quality is traditionally defined as the ability of a data collection to meet user requirements.

From an information system perspective, good quality data can be used to represent the real world without contradictions.

The data quality management includes:

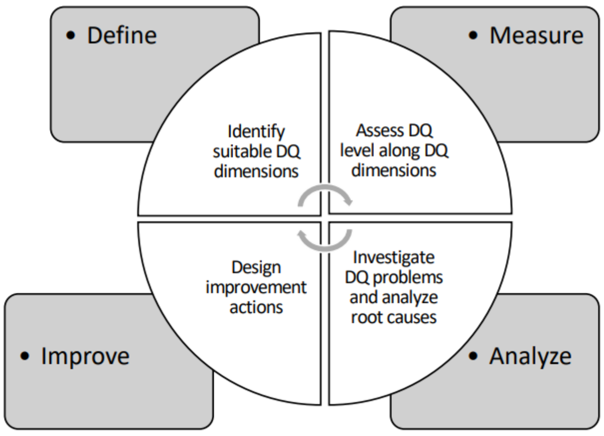
1. Quality dimensions definition
2. Quality dimensions assessment
3. Quality issues analysis
4. Quality improvement

The data quality methodology is composed of these four cyclic steps.

First quality dimensions are defined and identified, then they are measured, assessing the data quality level along the defined dimensions.

The found issues are analyzed, so to investigate the problems and analyze the root causes.

Then, action is taken to improve the overall data quality.


• Define
• Measure
• Improve
• Analyze
Identify suitable DQ dimensions
Assess DQ level along DQ dimensions
Design improvement actions
Investigate DQ problems and analyze root causes

Data Quality issues to consider in data preparation activities are mainly related to:

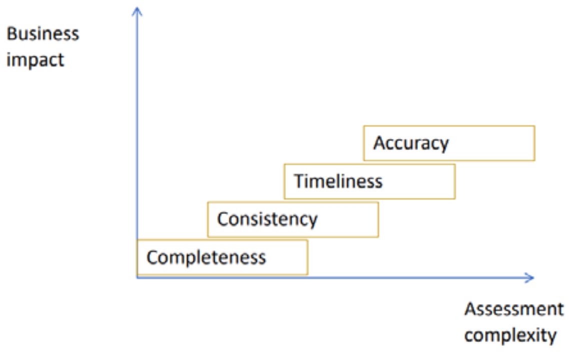
- Missing values
- Duplicate data
- Inconsistent data (e.g. two tuples with same primary key ID but different names)
- Outliers (e.g. values that are far from the average range of that attribute)
- Noise (e.g. “peter” vs “petr”)
- Out-of-date data

Data quality is measurable by data quality dimensions and metrics.

The most used objective dimensions are:

- Accuracy: the extent to which data are correct, reliable and certified
- Completeness: the degree to which a given data collection includes the data to describe the domain
- Consistency: the satisfaction of the semantic rules defined
- Timeliness: the extent to which data are sufficiently up-to-date for a task

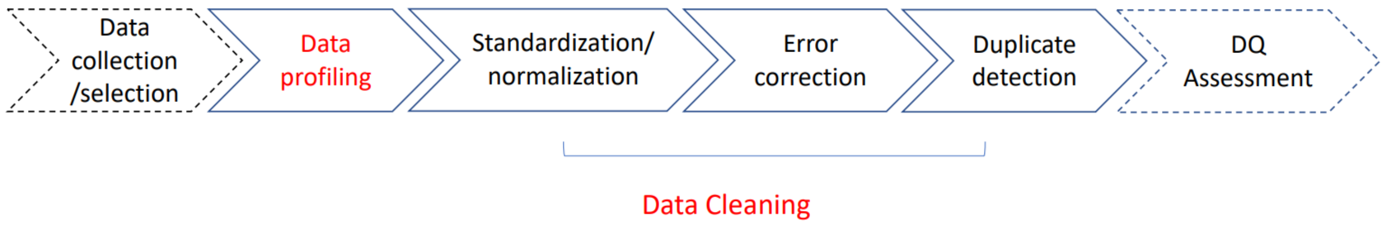
In most cases, the objective dimensions that are more complex to assess are also the ones with bigger business impact.



The process of data improvement has different strategies, divided in:

- Data-based approaches: they focus on data values and aim to identify and correct errors without considering the process and context in which they will be used
- Process-based actions: They are activated when an error occurs and aim to discover and eliminate the root cause of the error

One important data-based approach is **data cleaning**, consisting in the process of identifying and eliminating inconsistencies, discrepancies and errors in data in order to improve quality.



The **Data profiling**, which happens before the data cleaning, consists of the analysis of content and structure of attributes (Data type, domain, uniqueness, format, …), the analysis of dependencies between attributes of a single relation, the analysis of overlapping attributes from different relations, of the number of missing values or wrong values, and of duplicates.

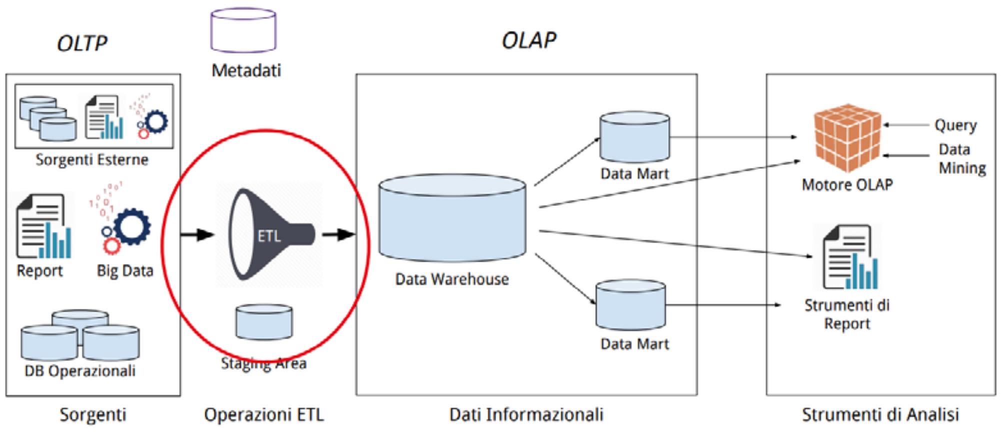
The tasks of data cleaning are:

- Normalization/standardization (Datatype conversion, discretization)
- Missing values detection and imputing
- Outliers detection
- Duplicate detection

In particular, Duplicate detection (or entity reconciliation) is the discovery of multiple representations of the same real-world object. To perform it, a good similarity measure is needed, and it would be appropriate to minimize the number of comparisons as much as possible.

In case of multiple sources, data integration is also needed.

Data Quality improvement methods are also used in Data Warehouse, during the ETL phase.



Big Data analysis allows to understand customer needs, improve service quality, and predict and prevent risks. High quality data are the precondition for guaranteeing the quality of the results of Big Data analysis, for which the four Vs (Volume, velocity, variety, veracity) pose relevant challenges.

Big Data tried to overcome Data Quality issues with Data Quantity. But quality is still an issue.

To summarize, the most common Data quality issues in big data are:

- Not integrated data
- Incomplete data
- Incorrect data
- Data cleaning have to be frequently performed
- Inconsistent sources and issues in data integration
- Source reliability
- Data variety
- Human resources: find the right competencies
- Data provenance and lineage information should be available

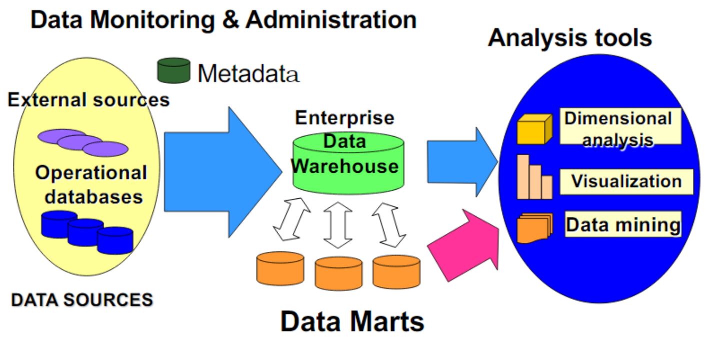
5. Data warehouses

A **data warehouse** is a single, complete and consistent store of data obtained from a variety of different sources, made available to end users so that they can use it in a business context. It is also a process for transforming data into information and for making it available to users in a timely enough manner to make a difference.

Data should be integrated into the data warehouse with the main objective to support the enterprise in its operations and organizational decision making.

Summary data provide real value to the organization, and historical data are used to understand data over time.

Data warehouses allow to perform analysis inside the company and of the company in relation to the market, also using data for forecasts and simulations (what-if).



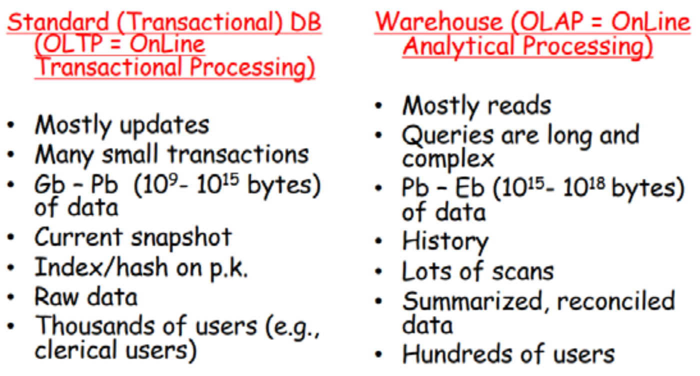
A data warehouse is:

- Subject-oriented: to use in relation to a subject of interest
- Integrated
- Time-varying: keeps all the variations of data in time
- Non-volatile: data is only added, never deleted or updated (unless found to be wrong)

A **data mart** is a part of the Data warehouse dedicated to one specific subject of interest.

A data warehouse is built as the result of the materialized data integration of the sources of interest available to the enterprise.

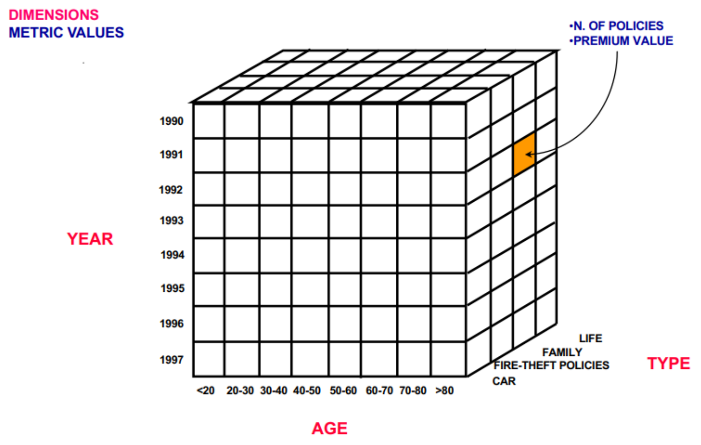
A data warehouse is stored in a relational DBMS, but it is a specialized DB, with a different kind of operations with relation to a transactional one.



Standard (Transactional) DB (OLTP = OnLine Transactional Processing)	Warehouse (OLAP = OnLine Analytical Processing)
• Mostly updates	• Mostly reads
• Many small transactions	• Queries are long and complex
• Gb - Pb (10^9- 10^{15} bytes) of data	• Pb - Eb (10^{15}- 10^{18} bytes) of data
• Current snapshot	• History
• Index/hash on p.k.	• Lots of scans
• Raw data	• Summarized, reconciled data
• Thousands of users (e.g., clerical users)	• Hundreds of users

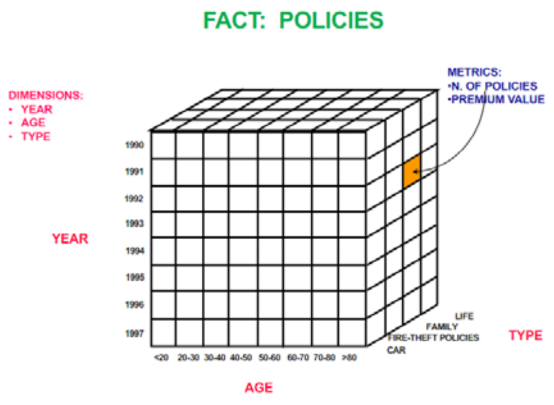
The logical data model for OLAP is the **data cube**, that allows analysis over different dimensions and hierarchies.

Dimensions of the cube are used as search keys, while the metric values are found in the cube cells.



The **Dimensional Fact Model** (DFM) allows to describe a set of **fact schema**, the components of which are:

- Facts
- Measures
- Dimensions
- Dimension Hierarchy



A <u>fact</u> is a concept that is relevant for the decisional process, typically it models a set of events of the organization.

A <u>measure</u> is a numerical property of a fact.

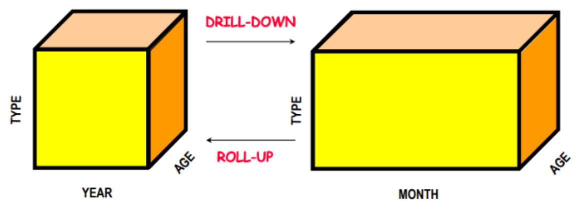
A <u>dimension</u> is a fact property defined w.r.t. a finite domain; it describes an analysis coordinate for the fact.

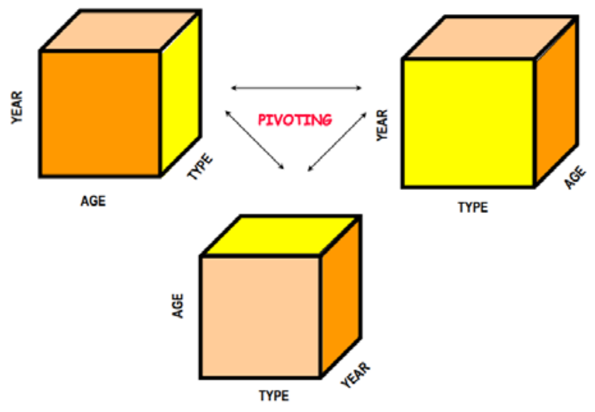
Time is always a dimension.

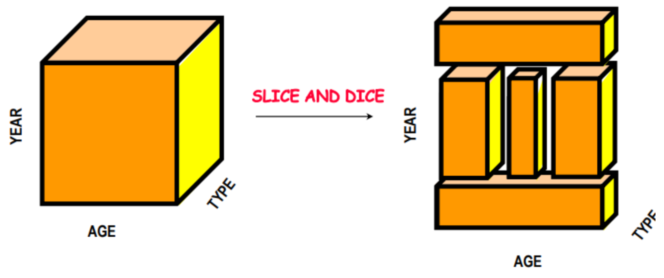
Dimensions can be organized in hierarchies.

The OLAP operations that can be performed on the Dimensional Fact Model, logically represented by the data cube, are:

- *Roll-up*: Aggregates data at a higher level
- *Drill-down*: De-aggregates data at the lower level
- *Slice-and-dice*: Applies selections and projections, which reduce data dimensionality
- *Pivoting*: Selects two dimensions to re-aggregate data (cube re-orientation)
- *Ranking*: Sorts data according to predefined criteria
- *Traditional operations* such as select, project, join, derived attributes, etc.







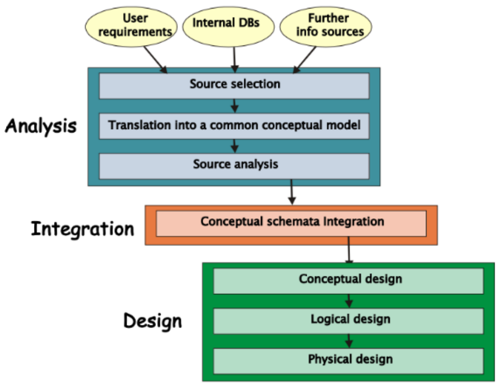
OLAP logical models can be of two kinds, according to how they store data:

- <u>MOLAP</u>: (Multidimensional On-Line Analytical Processing) stores data by using a multidimensional data structure (a “physical” data cube)
- <u>ROLAP</u>: (Relational On-Line Analytical Processing) uses the relational data model to represent multidimensional data. It’s by far the most adopted solution.

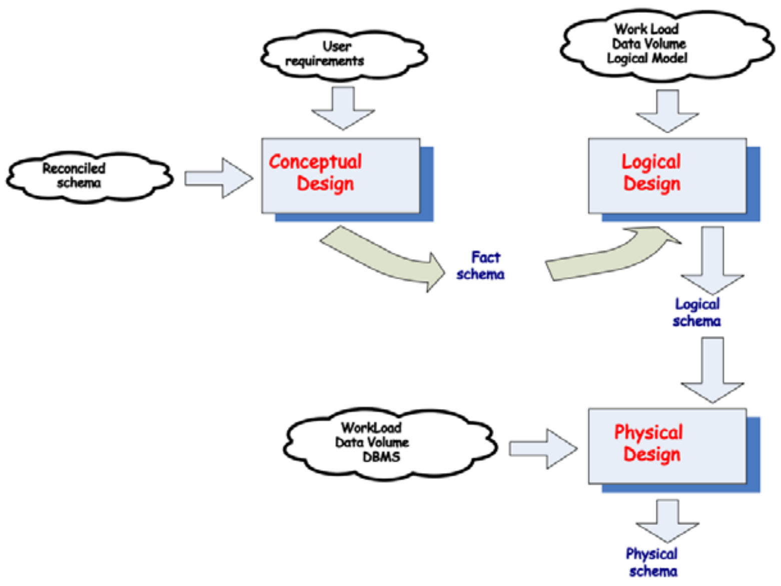
6. Data warehouse design

DW Conceptual design

Data Warehouses are based on the multidimensional model. At a conceptual level, the Entity/Relationship model cannot be used in the DW conceptual design.



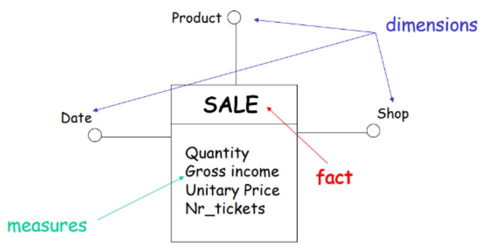
The conceptual design is that phase that determines what facts are of interest for the data warehouse and produces the fact schema. To decide what facts to include and how to model them, the reconciled schemas from the sources and the user requirements are both taken into account.



The **fact schema** is composed by facts, their measures and dimensions. A fact describes an $N:M$ relationship among its dimensions, with which there must be a functional dependency.

A dimensional attribute must assume discrete values, so that it can contribute to represent a dimension.

Dimensional attributes can be organized into hierarchies, represented as directional trees whose nodes are the attributes and edges describe $1:n$ associations between them, starting from the root dimension.



The root of a hierarchy corresponds to the primary event, and represents the finest aggregation granularity.

A **primary event** is an occurrence of a fact, and it is represented by means of a tuple of values.

Example: On 10/10/2001, ten 'Brillo' detergent packets were sold at the BigShop for a total amount of 25 euros

A **secondary event** aggregates some of the corresponding primary events.

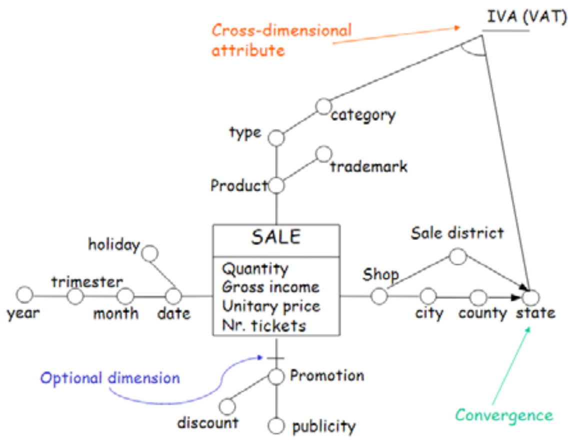
Example: In October 2001, 230 'Brillo' detergent packets were sold at the BigShop for a total amount of 575 euros

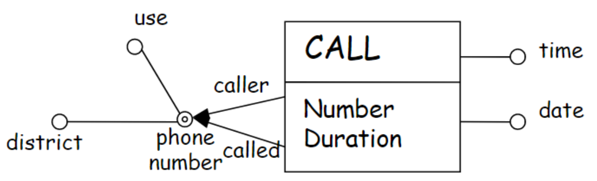
A descriptive attribute contains additional information about a dimensional attribute and is uniquely determined by the corresponding dimensional attribute.

Some edges of a fact schema could be optional.

A cross-dimensional attribute is a dimensional or a descriptive attribute whose value is obtained by combining values of some dimensional attributes.

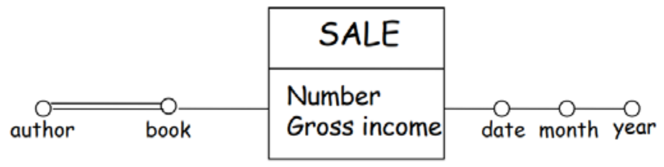
Convergence: two dimensional attributes may be connected by more than two distinct directed edges.



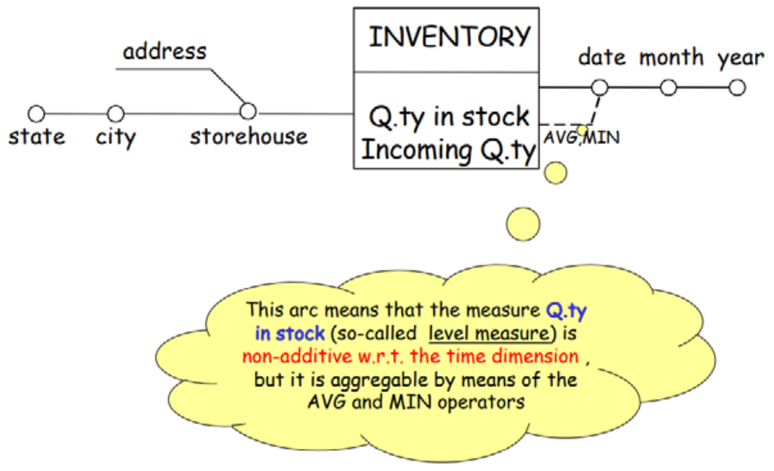


In a fact schema, some portions of a hierarchy might be duplicated, or hierarchy sharing could be allowed. For it to work, the functional dependency must hold on both branches.

Some attributes, or some dimensions, may be related by a many-to-many relationship, and are denoted on the fact schema by multiple edges. Then, at logical design time, they are dealt with in a special way.



Measure aggregation requires to specify an operator to combine values related to primary events into a unique value related to a secondary event. A measure is additive w.r.t. a given dimension if and only if the SUM operator is freely applicable to that measure along that dimension.



INVENTORY
Q.ty in stock
Incoming Q.ty
address
state
city
storehouse
date
month
year
AVG,MIN
This arc means that the measure Q.ty in stock (so-called level measure) is non-additive w.r.t. the time dimension, but it is aggregable by means of the AVG and MIN operators

A fact schema is *empty* if there are no measures. In this case, the default measure is the count.

It's possible to identify three different measure categories:

- Flow measures: related to a time period and are evaluated in a cumulative way at the end of that period. (e.g. amount of sales per day)
- Level measures: related to and evaluated in a particular time instant (e.g. amount of products in stock in a particular instant)
- Unitary measures: evaluated in particular time instants but are relative measures (e.g. money change rate in a particular instant)

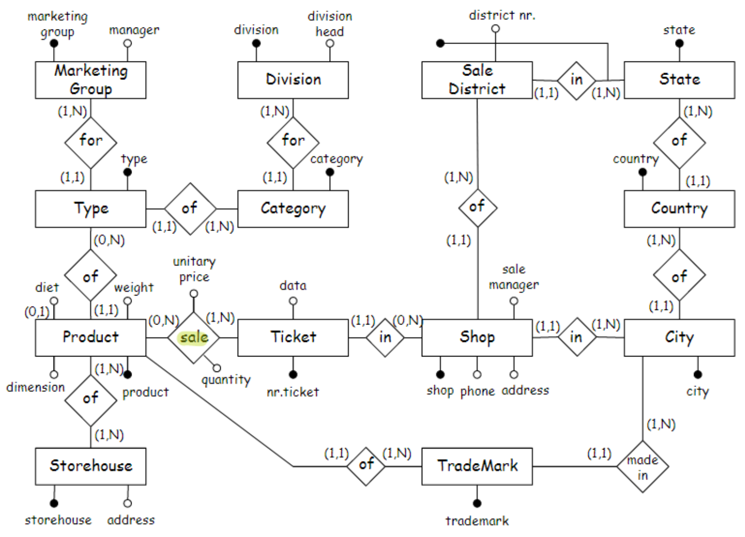
The **conceptual design** takes into account the documentation related to the integrated database as a starting point. The *top-down* methodology for conceptual design is the following:

1. Fact choice (subject oriented): decide what are the facts of interest
2. For each fact:
    a. Design of the attribute tree
    b. Attribute-tree editing
    c. Dimension definition
    d. Measure definition
    e. Fact schema creation

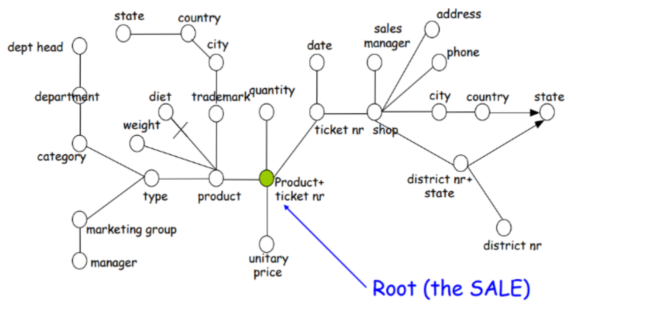
Facts correspond to events that dynamically happen in the organization; In an E/R schema, a fact can correspond to an entity or to a relationship, while in a relational schema, often a fact corresponds to a relation (table).

Good fact candidates are entities or relationships representing frequently updated archives, but not static archives, as a DW must be used to describe the dynamics of events over time.

When a fact is identified to be studied, it becomes the root of the fact schema.

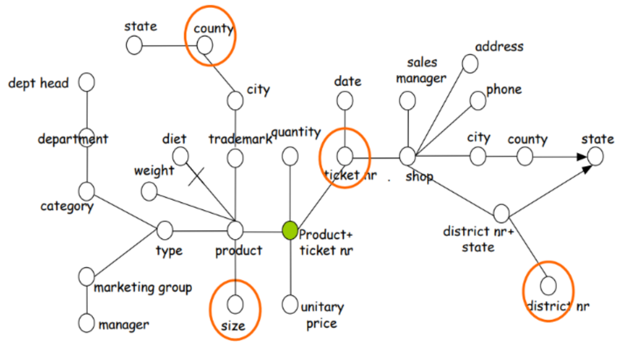


Once identified the fact, the *attribute tree* must be defined, composed by nodes corresponding to the attributes of the fact schema, and the primary key of the fact as root. The attribute tree can be obtained from the ER schema by using a semi-automatic procedure.



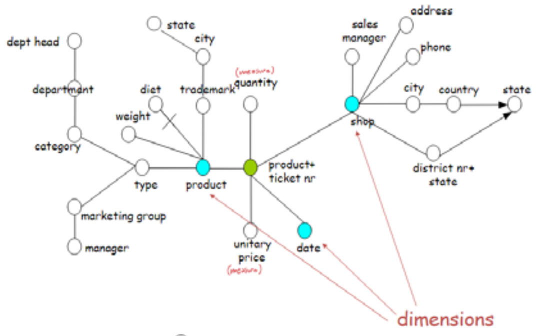
The *editing* phase for the attribute tree allows to remove some attributes which are irrelevant for the data mart:

- Pruning of a node v: the subtree rooted in v is deleted
- Grafting of a node v: the children of v are directly connected to the father of v



Cyclic relationships must be broken after a certain number of iterations, so Cycles in the schema must be broken, possibly choosing to keep the most convenient link.

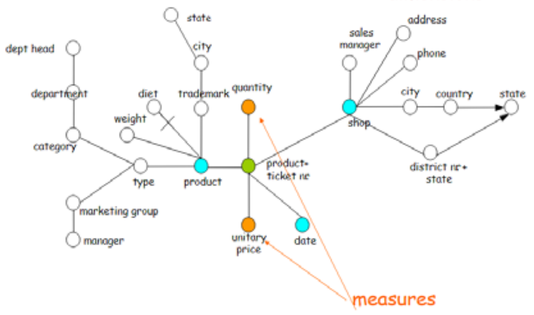
ISA hierarchies are treated like 0-1 (optional) relationships.



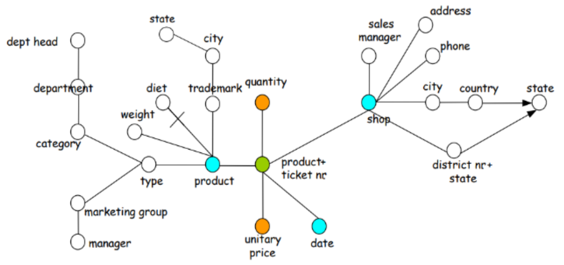
Then, *dimensions* must be defined, and can be chosen among the attributes that are children of the root of the tree. Time should always be a dimension.

To define the *measures*: if the set of attributes that act as fact identifier is included in the set of dimensions, then numerical attributes of the root (fact) are measures (e.g., unitary price). More measures are later defined by applying aggregate functions to other attributes of the tree.

It is possible that a fact has no measures (empty). If the granularity of a fact is different w.r.t. the granularity of the source schema, it can be useful to define suitable measures in order to aggregate the same attribute by using various operators.



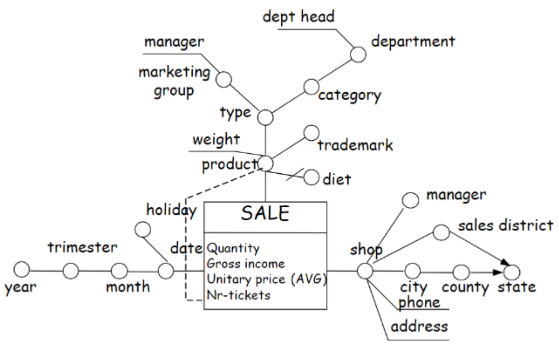
In the glossary, an expression is associated with each measure. The expression describes how we obtain the measure at the different levels of aggregation starting from the attributes of the source schema.



Quantity = SUM(Sale.quantity)
Gross income=SUM(Sale.quantity*Sale.unitaryprice)
Unitary price=AVG(Sale.unitaryprice)
Nr-tickets=COUNT(*)

At the end, the attribute tree is translated into a fact schema including dimensions and measures:

- Dimension hierarchies correspond to subtrees having as roots the different dimensions (with the least granularity)
- The fact name corresponds to the name of the selected entity



DW Logical design

The MOLAP (Multidimensional OLAP) logical model is the more traditional way of OLAP analysis. Data is natively stored in a multidimensional cube in proprietary formats. It offers an excellent performance in data retrieval and operations like slicing and dicing, it can easily perform complex calculations that are pre-generated when the cube is created.

However, there is a limit to the amount of data this model can handle, as it is not possible to include a lot when the calculations are generated when the cube is built. Indeed, data in the cube can be derived from a large amount of data, but only summary-level information will be included. This model also requires additional investment, as the cube technology are often proprietary.

The ROLAP (Relational OLAP) logical model is the current de-facto standard. It relies on manipulating the data stored in the relational database to give the appearance of traditional OLAP's slicing and dicing functionality. Given its underlying structure, it can leverage also functionalities inherent in the relational database, and can handle large amount of data.

On the other hand, performance may be slow, because every report is essentially a heavy query. Moreover, as ROLAP technology mainly relies on generating SQL statements to query the relational database, and SQL statements do not fit all needs, it is traditionally limited to what SQL can do. ROLAP vendors have mitigated this risk by building into the tool out-of-the-box complex functions.

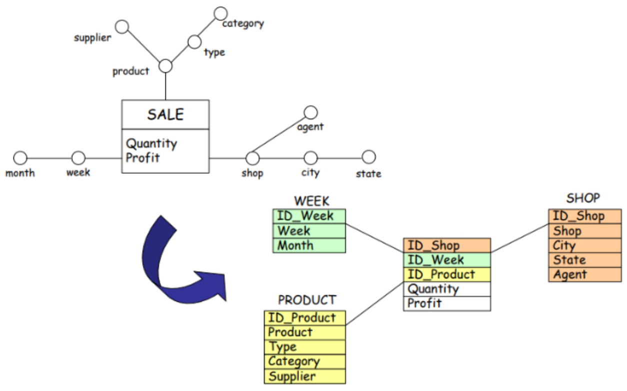
A third option is HOLAP (Hybrid OLAP), which attempts to combine the advantages of MOLAP and ROLAP. For summary-type information, HOLAP leverages cube technology for faster performance. When detail information is needed, HOLAP can "drill through" from the cube into the underlying relational data.

We will focus on Data Warehouse Logical Design in ROLAP.

ROLAP is usually based on the **Star schema**.

This kind of schema mimics the fact schema itself, and is composed of:

- A fact table, that imports the primary keys of dimension tables and has an attribute for each measure
- A set dimension tables, each corresponding to a dimension. Each dimension table is characterized by a primary key and a set of attributes



It is possible to define different variants of the star schema to manage aggregate data.

The fact table contains information expressed at different aggregation levels.

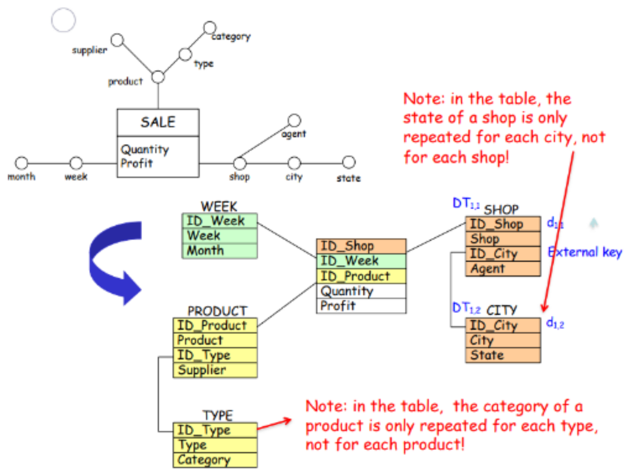
Dimension tables are de-normalized. This introduces redundancy, but comports that there are fewer joins to do.

The **snowflake schema** reduces the de-normalization of the dimensional tables.

Dimensions tables of a snowflake schema are composed by a primary key and a subset of attributes that depend on the primary key, plus zero or more external keys that allow to obtain the entire information.

The keys of the primary dimension tables are imported in the central fact table, then there are secondary dimension tables connected to the primary dimensions or to other secondary dimension tables.

This schema is more normalized and thus more space efficient. As well as the reduction of needed memory space, there are advantages in the execution of queries related to attributes contained into fact and primary dimension tables.



In data warehouses, aggregation allows to consider concise and summarized information, but aggregation computation is very expensive. A solution is in the use of pre-computation, to store queries to reduce time, but needing more space.

A **view** denotes a fact table containing aggregate data.

A **materialized view** corresponds to a query and is kept updated after every operation.

A view can be characterized by its aggregation level (pattern), in fact there are:

- Primary views: correspond to the primary aggregation levels, to queries on the first level of the hierarchy of attributes
- Secondary views: correspond to secondary aggregation levels (secondary events)

Sometimes it is useful to introduce new measures in order to manage aggregations correctly. These derived measures are obtained by applying mathematical operators to two or more values of the same tuple.

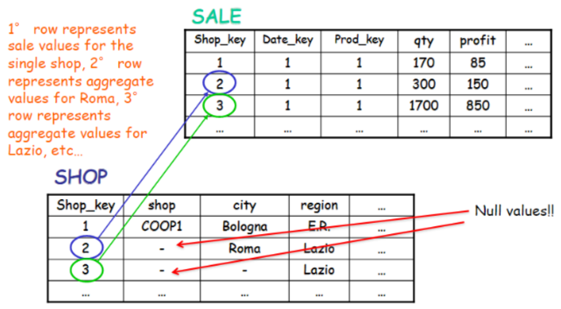
There are three kinds of aggregate operators:

- Distributive operator: allows to aggregate data starting from partially aggregated data (e.g. sum, max, min)
- Algebraic operator: requires further information to aggregate data (e.g. avg)
- Holistic operator: it is not possible to obtain aggregate data starting from partially aggregate data (e.g. mode, median)

Programs to compute information in form of aggregate data, called aggregate navigators, are currently included in the commercial DW system. They allow to formulate OLAP queries in the "best" view and manage aggregates only by means of distributive operators.

It is possible to define different variants of the star schema in order to manage aggregate data.

A first solution is to store data of primary and secondary views in the same fact table, using NULL values for attributes having aggregation levels finer than the current one.


1° row represents sale values for the single shop, 2° row represents aggregate values for Roma, 3° row represents aggregate values for Lazio, etc…
SALE
SHOP
Null values!!

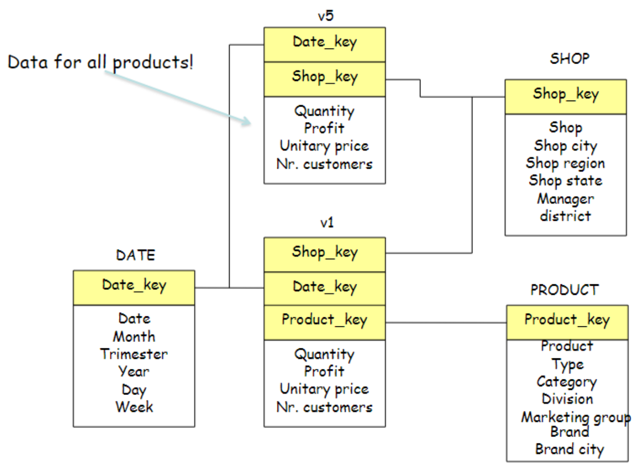
Shop_key	Date_key	Prod_key	qty	profit	...
1	1	1	170	85	...
2	1	1	300	150	...
3	1	1	1700	850	...
...	...	...	...	...	...

Shop_key	shop	city	region	...
1	COOP1	Bologna	E.R.	...
2	-	Roma	Lazio	...
3	-	-	Lazio	...
...	...	...	...	...

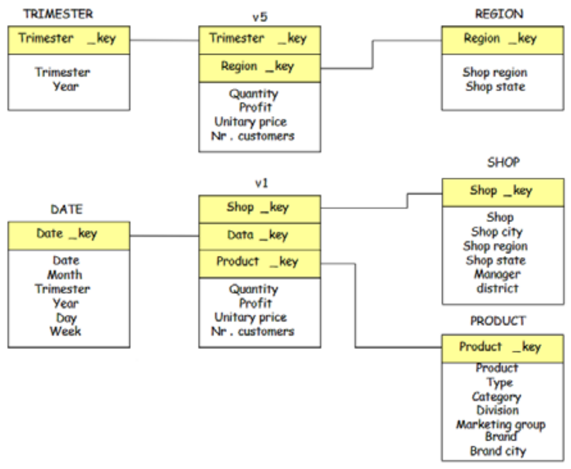
Another solution it to store distinct aggregation patterns in distinct fact tables, creating a **constellation schema**.

The constellation schema can offer the maximum optimization level with separate fact tables, and also replicated dimension tables for different (the most used) aggregation levels.

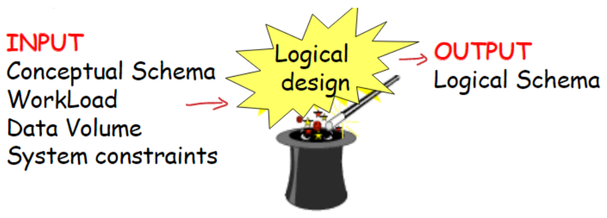
Constellation schema:



Alternative solution:



During the **logical modeling** of a data warehouse, the objective is to obtain the logical schema for a specific data mart, starting from the conceptual schema.



In OLAP systems, the workload is dynamic in nature and intrinsically extemporaneous, because users' interests may change during time, or the number of queries may grow when users gain confidence in the system. In any case, OLAP should be able to answer any (unexpected) request. It's desirable to try to deduce the workload for a system from interviews with users and standard reports. Later on, at system run-time, workload can be deduced from the system log.

The data volume depends on the number of distinct values for each attribute, the attributes' size, the number of events, and determines the table and index dimension, along with access time.

The steps of ROLAP logical modelling are:

1. Choice of the logical schema (star/snowflake schema)
2. Conceptual schema translation
3. Choice of the materialized views
4. Optimization

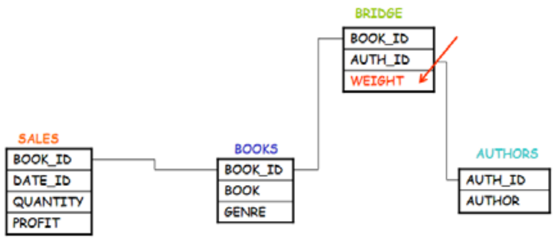
So, first the star or snowflake schema is created.

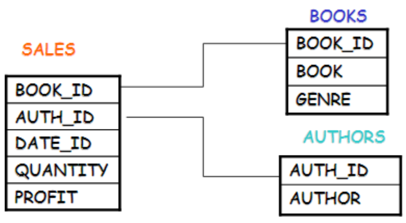
Note that the dimension table of a shared hierarchy should not be duplicated, whether the two hierarchies contain the same attributes, but with different meanings, or the two hierarchies contain the same attributes for part of the hierarchy trees.

When there are multiple edges, representing $N:N$ relationships, they are modeled by *bridge tables*. The key of the bridge table is composed by the combination of attributes connected to the multiple edge.

Weighed queries take into account the weight of the edge.

If it's desired to keep the star model, an attribute of interest can be added to the fact schema to model the relationship.





The choice about views that have to be materialized takes into account contrasting requirements: cost functions' minimization, system constraints and users constraints. The objective is to reduce the workload and to keep the view maintenance as low as possible.

It is useful to materialize a view when it directly solves a frequent query or it reduces the costs of some queries. It's instead not useful its aggregation pattern is the same as another materialized view or its materialization does not reduce any cost.

7. Other architectures for Big Data

OLAP (*On Line Analytical Processing*) is meant to support the processing of decision operations, through the use of complex and random operations involving a lot of historical and aggregate data. In OLAP the "ACID" properties (Atomicity, Correctness, Isolation, Durability) are not relevant, because the operations are read-only. The performance metrics for these systems are the query throughput and their response time.

OLTP (*On Line Transaction Processing*) are systems meant to support the processing of transactions, which implement the operational processes of the company. The operations involved are often predefined and relatively simple, involving few data, but these data must be always up-to-data and detailed. The "ACID" properties of transactions are essential. The main performance metric is the transaction throughput.

"ACID" properties are:

- *Atomicity*: a transaction is an indivisible unit of execution; it is executed fully or not at all.
- *Consistency*: the execution of a transaction must not violate the integrity constraints (foreign key, functional dependencies, facts, …)
- *Isolation*: the execution of a transaction is not affected by the execution of other concurrent transactions (concurrency control)
- *Durability* (persistence): the effects of a successful transaction must be permanent

The classical DBMSs (also distributed) are **transactional systems**. They ensure the ACID properties are respected and base their flow on the execution of transactions as elementary unit of work.

Recently, new DBMS have been proposed that are not transactional systems.

These non-transactional DBMS are commonly called **NoSQL DBMS**.

The name could be misleading, because SQL could actually be used in a NoSQL system, because the fact that a system is relational and thus uses SQL has nothing to do with the fact that it has or not transactional characteristics.

It's nowadays possible to store Big Data on **Data Clouds**, which are on demand storage services, reliable and offered on the internet with easy access to a virtually infinite number of storage resources, computing and network.

In general, the possible methods for storage can be classified as:

- **Centralized** or **Distributed**, transactional, based on a traditional model like the relational one
- **Federated** and **multi-databases**, for companies and organizations that are associated and share their data on the internet
- **Cloud databases**, to support Big Data by means of load sharing and data partitioning (usually NoSQL)

NoSQL databases provide flexible schemas, where the updates are performed asynchronously (It's never granted that un update is executed immediately on all the copies of the data). Potential inconsistencies in the data must be solved directly by users, as there are no constraints on data.

NoSQL databases are object-oriented friendly, as data is accessed via keys, that could correspond to IDs of objects, so it is easier to interface the database with an object oriented system.

Caching is easier and often embedded.

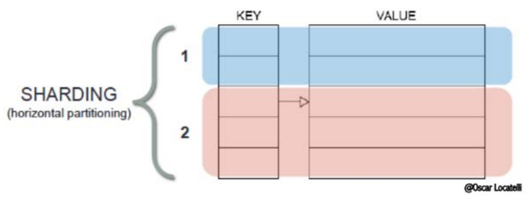
Not all "ACID" properties are supported.

The data model for NoSQL systems is of 4 main categories:

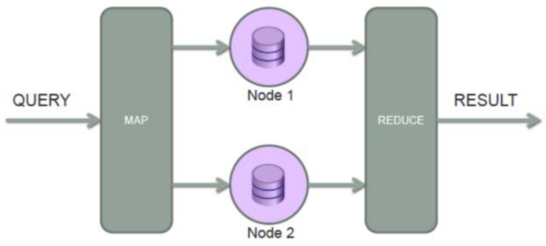
- Key-value
- Document-based
- Column-family
- Graph-based

With the **key-value** data model every entity is associated to a key, that is used for its retrieval and the value of the object is opaque (not related to the key in any way). There is no schema and there are no relationships in the database between data, and entities are decoupled and de-normalized.

This allows to store entities on different machines without having to worry about "neighborhood". The technique is called sharding and consists of the horizontal partitioning of the key-value. Every piece of data (table) is divided into shards and put on different nodes of the network.

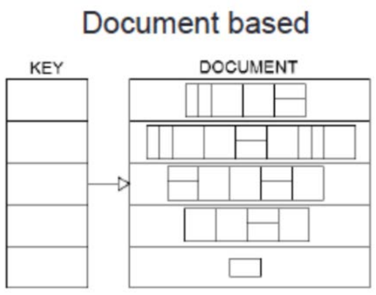


When a query comes, it is mapped to a number of nodes of the cloud, each of which contains what is necessary to perform part of the "big" operation- The results returned by each node are put together (reduced) to produce the result of the query. This process is called Map Reduce.



The key-value model is evolved into the document based and the column oriented model.

In the **document-based** model, each key is associated to a collection of documents whose structure is not fixed, each one has its own, and sometimes documents are nested. The ability to query is very rich as it is possible to filter on the internal fields of documents.



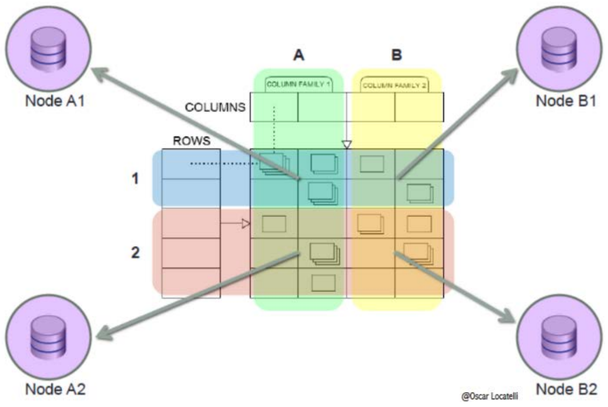
In the **Column-family** model the key is composed by a triple: $(row, column, timestap)$

The data is partitioned horizontally (sharding) and vertically by the different columns stored in different machines.

This model offers the maximum scalability, and is so oriented to Big Data.

The queries allow to filter only on rows and column keys and the sorting is automatic.

Columns in the same column family will be "close": to be determined at design time.



For NoSQL systems, holds the **CAP theorem**:

A data management system shared over the network (cloud, networked shared-data system) can guarantee at most two of the following properties:

- Consistency(C): all nodes see the same data at the same time, so all copies of data are synchronized
- Availability(A): a guarantee that every request receives a response about whether it was successful or failed; therefore, if an answer is received from a query, then surely the work has been done
- (tolerance to) Partitions(P): the system continues to operate despite arbitrary message loss or failure of part of the system

However, all these 3 properties are more "fuzzy" than binary, and the choice between C and A can occur many times within the same system at very fine granularity.

Note that in ACID the C means that a transaction preserves all the database constraints, while the C in CAP refers only to copy consistency, a strict subset of ACID consistency.

In more traditional applications, such as those banking or accounting, or bookings etc. these systems can be catastrophic. NoSQL is not suited for critical applications.

Some applications of NoSQL can be: data collected from sensors and datasets which are seldom updated.

8. Frontiers in Data Integration

Databases are assumed to represent certain data: a tuple in the database is true, any tuple NOT in the database is false (Closed World Assumption). But real life is not as certain, and to model it there are **Uncertain databases** of various kinds (e.g. based on probability), which attempt to model uncertain data and to answer queries in an uncertain world.

One way to cope with uncertainty could also be ranking.

Uncertainty in data integration can have different reasons:

- Data itself may be uncertain (e.g. extracted from an unreliable source)
- Mappings might be approximate (e.g. created by relying on automatic ontology matching)
- Reconciliation is approximate
- Approximate mediated schema
- Imprecise queries, such as keyword-search queries, are approximate

The two semantics of uncertainty are:

- Probabilistic: one value is either one thing or another, each with a probability (e.g. 60% means that with 60% probability something is in a way, with 40% in another, but is either one or the other)
- Fuzzy: something is for a percentage true and for a percentage false, with percentage meaning part of one (e.g. 60% means that 60 out of 100 tuples are that way)

Whatever the semantics of uncertainty (e.g. probabilistic...) an uncertain database describes a set of possible worlds.

Researchers are still looking for solutions for detecting duplicates in data. A common approach is the use of record-matching rules (rule-based ER), which are constraints that state “if any two records are similar on properties $P1, \dots, Pn$, then they refer to the same entity". The problem is that it’s difficult to generate record-matching rules and it’s challenging to define the bounds of similarity for approximate-match conditions. Moreover, efficient support for approximate-match conditions is non-trivial and one-to-one record matching too expensive.

Data provenance is the problem of knowing where the data have come from and how they were produced, which may be critical. The database community models provenance in terms of how the datum was derived from the original source databases, the rest is left to the application.

Provenance can be seen as a series of Annotations on data describing how each data item was produced, or can be seen as a graph of data relationships modeling derivations of tuples with edges. These two views are equivalent, and it is possible to convert one into the other as convenient.

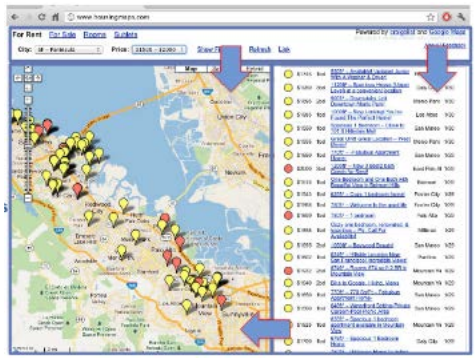
Provenance can be used to provide explanations about data results, or to assess the data quality of a source, or to understand the influence of sources on one another.

Data usage, user feedback and data quality info can be exploited to assess uncertainty and automate cleaning.

In order to build Large-Scale Structured Web Databases, methodologies for designing a fully integrated database coming from heterogeneous data sources are developed.

Sometimes, there might be the need to integrate data from multiple sources to answer a question asked once or twice. In this case, the integration needs to be done quickly and by people without technical expertise. **Lightweight integration** can be performed in this case, ideally using machine learning and other techniques to amplify the effects of human input.

A **mashup** is an application that integrates two or more mashup components at any of the application layers (data, application logic, presentation layer) possibly putting them into communication with each other.



The key elements of a mashup are:

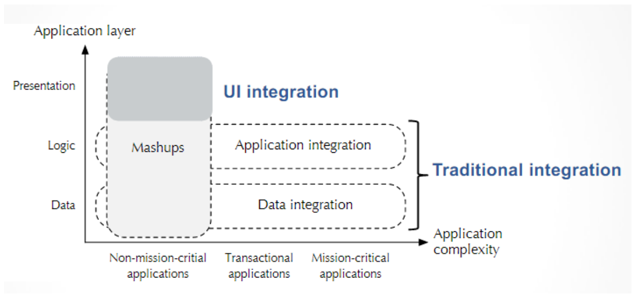
- Mashup component: is any piece of data, application logic and/or user interface that can be reused and that is accessible
- Mashup logic: is the internal logic of operation of a mashup; it specifies the invocation of components, the control flow, the data flow, the data transformations, and the UI of the mashup

A mashup can be of different types:

- Data mashups: Fetch data from different resources, process them, and return an integrated result set
- Logic mashups: Integrate functionality published by logic or data components
- User Interface (UI) mashups: Combine the component's native UIs into an integrated UI
- Hybrid mashups: Span multiple layers of the application stack, bringing together different types of components inside one

Integration happens at more than one layer.

Data mashups are a Web-based, lightweight form of data integration, meant to solve different problems. They offer the possibility to produce very specific reports or ad-hoc data analyses, or simple, ad-hoc data integrations providing "situational data" that meet short term needs. In any case, they can answer to non-mission-critical integration requests.

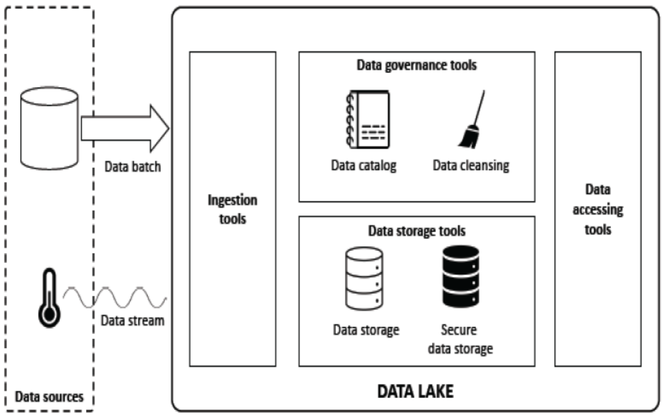


In contrast to the other kinds of data integration systems, **pay-as-you-go** systems try to avoid the need for the initial setup phase that includes creating the mediated schema and source descriptions. The goal is to offer a system that provides useful services on a collection of heterogeneous data with very little up-front effort.

A possible current solution are also the **dataspaces**, meant to be used instead of data warehouses.

They are based on two basic principles: to offer keyword search over a collection of data coupled with effective data visualization, and to improve the metadata in the system to the end of supporting and validating schema mappings, instance and reference reconciliation, or improve the results of information extraction.

Data lakes are dataspaces meant to manage big-data volume and variety, to provide data analysts with a self-service environment in which advanced analytics can be applied.



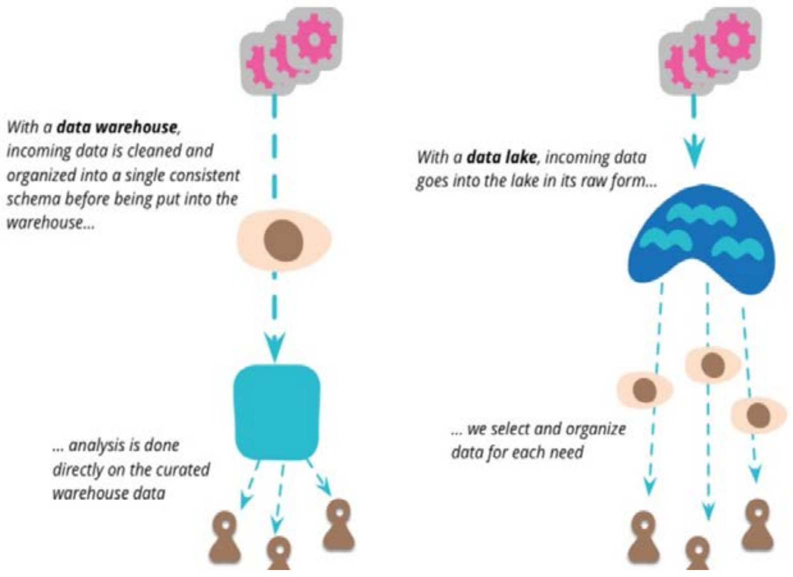
When a query or an update is submitted to a traditional integration system, this has to perform some reformulation in terms of a set of queries over the single datasets, determining which datasets are needed to answer the query, evaluating predicates in the single datasets or over all the involved ones when combined.

In a data lake these operations are performed in a pay-as-you-go fashion, avoiding the creation of mediated schemas or the permanent establishment of relationships between the elements of different sources (entity resolution and data fusion), and relying instead on temporary links based on metadata.

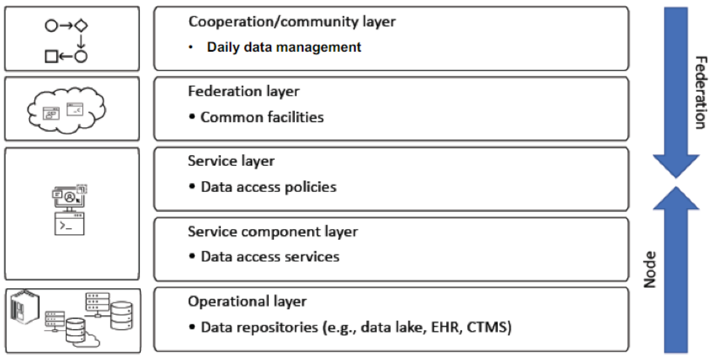
So, keyword search is utilized over a collection of data, and the metadata are kept improved as much as possible.

The **data catalog** supports the organization and description of data in order to ease data access, exploration and discovery. They are based on metadata and offer dataset annotations and interfaces supporting data search.

Metadata can be: descriptive, structural, administrative, or semantic. A more coarse-grained classifications consider technical metadata (related to the profile of the dataset), and business metadata (which provides a domain-aware description).


With a ***data warehouse***, incoming data is cleaned and organized into a single consistent schema before being put into the warehouse...
... analysis is done directly on the curated warehouse data
With a ***data lake***, incoming data goes into the lake in its raw form...
... we select and organize data for each need

Data Warehouse	Dataspace
Usually Relational DBMS	Often HDFS/Hadoop
Data of recognized high value	Candidate data of potential value
Aggregated data	Detailed source data
Data conforms to enterprise standards	Fidelity to original format and condition (transformed/cleaned later, on demand, when needed)
Known entities	Raw material (metadata) to discover entities
Data integration up front	Data integration on demand
Schema on write	Schema on read (when doing analytics)



A data lake is the collection of all the raw, unfiltered data of an organization.

Data lake federation is the software process of collecting all the data across the organization in a single repository, making the data search easier for the end-users.

From bottom to top:

- Operational layer: organizations that have implemented their own data repositories
- Service component layer: it supports the implementation of the data-access services, in charge to expose the selected datasets
- Service layer (middle layer): it implements the policies that regulate the access to the exposed data sets through the defined services
- Federation layer: it offers both infrastructural and application services, to create an ecosystem that enables data sharing
- Cooperation/community layer: it offers the tools to define and manage multi-centric operations

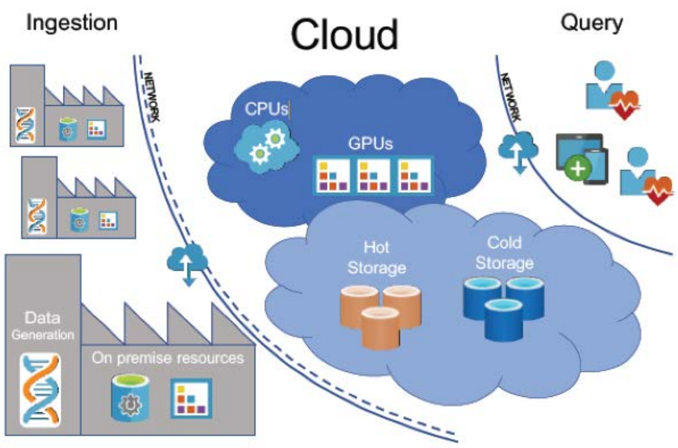
Some Descriptive and Administrative(Technical) metadata should exist also for Services. Metadata about the computation must also be added to support the peculiarities of the various kinds of data analysis. The challenge is to identify a common, minimal and sufficient set of computation-related metadata to be used to decide if the analysis should be executed locally or remotely, and with which resources.

Depending on the types of data at stake, computational resources may require CPU or GPU support.

Storage should be organized into a *Hot Storage*, containing data requiring quick access, and a cheaper and slower *Cold Storage* for less frequently used data.

Data analysis can be performed both on-premises and on the cloud.

The network is used for ingestion and for querying.



Very often, many people are trying to access the same data sources and perform similar tasks. These patterns of behavior could be used to identify correlation between sets of data or suggest better ways to optimize them based on the most frequent uses. Cloud-based tools for data management might provide such services since they log the activities of many users, but these strategies are limited by privacy issues.

Visualizing integrated data could allow to visualize important patterns in the data instead of an infinite number of rows, in order to immediately see discrepancies between the data in the sources, or to visually show the search results of a query, or visualize information about data provenance.

Integrating social media is an interesting perspective for the future. Social media data are often noisy, have transient nature and often lack of context. These data often arrive as high-speed streams that require very fast processing.

Most query engines, schema matchers, storage systems, query optimizers, have been developed for operation on a single server or a few servers. Most algorithms (schema matching, entity resolution, data cleaning, indexing, etc.) are based on assumptions of limited scale (sometimes main memory): they need to be tackled in a much more parallelizable and scalable way and need to be redesigned to exploit the power of large clusters.

9. Ethics in Data Science

Thanks to the unprecedented data collection capabilities and the enormous computational power available, Big Data has unlocked many potentialities.

But, as data have an impact on almost every aspect of our lives, it is more and more important to understand the nature of this effect.

With statistics used everywhere, it may happen that very critical decisions be taken without taking their ethical consequences into account.

Technology must be accompanied by ethical and legal considerations, because data may contain bias and algorithms often produce opaque processes that cannot be explained.

A scientist must identify what datasets to use, understand their content and chose a knowledge extraction technique to deploy in order to obtain a fair result. This sequence of choices may strongly influence the process, and biased results might be obtained.

Ethical dimensions as data protection, fairness, diversity, transparency should become fundamental issues that can drive new ideas, and should be enforced in all processes.

Data protection is something to consider while designing knowledge extraction systems, that hardly pay specific attention to ethically sensitive aspects of knowledge extraction processes and their outcomes. Such aspects are now becoming prominent, especially with regard to the protection of human rights and their consequences in normative ethics.

GDPR unifies data protection laws across all European Union members, defining a comprehensive set of rights for EU citizens, describing the requirements for companies and organizations for collecting, storing, processing and managing personal data.

Fairness of data is defined as the lack of bias. A system not properly trained would produce results that are not fair.

Transparency is the ability to interpret the information extraction process in order to verify which aspects of the data determine its results. Transparency metrics can use the notions of data provenance and explanation of how a result is obtained. However, transparency may conflict with Data Protection.

Diversity is the degree to which different kinds of objects are represented in a dataset. Ensuring diversity at the beginning of the information extraction process may be useful for enforcing fairness at the end.

The diversity dimension may conflict with data quality needs, that prioritize the use of few, high-reputation sources.

In addition to these 4 dimensions, Data quality is a typical ethical requirement: we could never trust a piece of information if it did not have the typical data quality properties. Yet, we can also assert the opposite: that data should conform to a high ethical standard, for it to be considered of good quality. Hence, the satisfaction of the ethical requirements is actually necessary to assert the quality of a result.

It is the responsibility of the system designer, and of the person/company that ordered the job, to ensure that the (necessary) ethical properties are satisfied.

Questions from past Exams

Data Integration questions

What is data integration? Which type of data integration (materialized or virtual) is used in the case of Data Warehousing? Describe the main differences between the two approaches and justify the answer in your own words, possibly using a small example.

Data integration is the process aiming at combining data coming from different data sources to provide the user with a unified vision of these data.

The main issues in this process come for the 4 Vs: volume of data, velocity of data changes, variety of data in the sources, veracity as the difference in levels of quality of data.

Talking about variety, the sources may be heterogeneous in many ways: use different platforms, different data models, different query languages, schemas, or have different values for the same information.

The process of data integration is composed of the steps: schema reconciliation, record linkage and data fusion.

Materialized and virtual data integration are two ways of integrating Database systems.

When materialized integration is performed, data from the data sources is periodically extracted, transformed and loaded (ETL process) into a single database where they are merged. The sources are reconciled and the queries are posed on the database that is result of the process.

The information contained in materialized databases is meant to contain the history of the enterprise, and because of the way it is collected, cannot be up-to-date data in every moment.

Instead, virtual integration allows to have a single collection of up-to-date data. This is possible because in a virtual database the data is never extracted from the sources, but left there, where it is constantly updated.

To query on a virtual database, an application interfaces with the built global schema, which contains all the attributes of interest of the sources combined, like if they were in a single database. When a query is posed on the global schema, it is translated in smaller queries to the single data sources needed and then the results of these queries are merged to produce the answer to the initial query.

When only two data sources are involved, virtual integration can happen in the form of virtual data exchange, consisting in the creation of gateways between the two sources.

Data warehousing makes use of the materialized approach, as it doesn't need dynamic data but only historic and aggregated data. This also allows for a better performance for complex queries that the decision making process of a data warehouse implies.

Describe the conflict analysis step in data integration; describe the most important classes of conflicts and provide a set of examples.

Once the identification of the sources and the reverse engineering of their schema have been completed, the next step in the data integration process is conflict resolution and reconstructing, to build the global schema.

When coming to it, the steps to follow are: first, related concept identification, done manually or automatically with humans intervening when needed, followed by conflict analysis and resolution and at last conceptual schemata integration.

The main classes of conflicts are the followings:

- Name conflicts: when two attributes have the same name but different meaning (e.g. "number" can be a phone number or refer to an amount of something), or when attributes have different name but the same meaning (e.g. "cost" and "price")
- Type conflicts: the same attribute can be represented by different types, for instance M/F or 0/1. Two different entities might represent the same real concept with different types of attributes.
- Data Semantics conflicts: two schemas could have data referring to different conventions such as different currencies, measure systems, granularities (e.g. meters vs feet or vs Kilometers to represent attribute "distance")
- Structure: if there is a difference in the structure of relationships between elements, for example if the same concept is represented with an attribute in one source and with an entity in another.
- Cardinality: same relation between entities but with different cardinalities.
- Key: the same entity in different sources has different keys.

Describe the GAV (global as view) and LAV (local as view) approaches used to define the mapping between the global logical schema and the single source schemata in the data integration context. Discuss the main differences between the two approaches and describe under which conditions GAV is more appropriate than LAV, and vice versa.

A data integration system must understand which real data in the sources correspond to the data in the virtual database, and to do so it must map the global logical schema to the single sources schema.

To do so, two main approaches are possible: GAV and LAV.

With the Global As View approach (GAV), the global schema is expressed in terms of views on top of the data sources. This approach leaves few work to do when the system needs to answer a query and is good for static data sources, because, if the sources were dynamic, then upon the addition of a new source, or the modification of one, the views performing the mapping would need to be modified, and this can be a complex operation.

With the Local As View approach (LAV), the sources are the ones to be expressed with views in terms of the global schema. So, the global schema can be defined a priori and never be modified, while adding a new source only implies the need to define the associated view in terms of the global schema. LAV favors extensibility and modularity.

However, in LAV the query processing is much complex, because the needed transformation to answer a query is the opposite of the one used to do the mapping. Therefore, some "reasoning" is needed by the system every time, in order to build a "plan", that is a rewriting of the queries to the sources.

A third approach is Global and Local As View (GLAV) that is a hybrid between the two.

Define Wrappers and Mediators, explain in which circumstances their use is advised in Data Integration and the way they work. Discuss the various types of Wrappers and Mediators that have been introduced during the course.

Often the Data Integration process must deal with multiple, distributed, and heterogeneous data sources which can include semi-structured or unstructured data. For the former, data have some form of structure, but is not prescriptive, regular, or complete as in traditional DBMS. The final aim is to integrate, query and compare data from multiple sources and with different structure just as if they were all structured.

To this aim, mediators and wrappers are employed.

Wrappers are modules that are associated to a single source, for which they are built ad-hoc. Their job is to perform translation between the language of the data integration system and the one of the source. A wrapper converts queries or commands going into the source and results coming out of them. For semi-structured data sources, a wrapper must be able to convert irregular data into something more regular.

The wrapper is very specific, so a change in the source generally needs the wrapper to be maintained to adapt to it. Wrappers' maintenance is generally expensive, so some techniques to partially automate it exists.

Mediators is the tool connected to all the wrappers on one side and to the applications needing the data integration system on the other, and are the ones to perform the integration when it comes to semi-structured data. The mediator must build and maintain knowledge structures useful for the transformation of data from sources into information. Therefore, mediators are domain specific. Mediators can also include an intermediate storage.

(During the course) TSIMMIS is the first system based on the mediator/wrapper paradigm. In TSIMMIS, queries are posed to the mediators in a specific object-oriented language called LOREL and the data model adopted in mediators is OEM (Object Exchange Model), a graph-based and self-descriptive model which directly represents data with no schema at all. The data model is completely managed by the mediator that knows the semantics of the application domain and receives the queries, while wrappers are used for the model-to-model translations.

<u>Define ontologies and their possible use in data integration.</u>

An ontology is a way of representing a given domain via a controlled vocabulary that describes objects and the relationships between them. It's a way of giving a unique meaning to the terms that define the knowledge about a given domain to allow for use of a common vocabulary for automatic knowledge sharing.

An ontology is composed of:

- A T-box, containing the concepts and role definitions, the axioms of the logical theory
- An A-box, containing the instances and the basic assertions

There are two types of ontologies:

- Taxonomic ontologies, that provide a reference vocabulary and define concepts through terms and their relationships
- Descriptive ontologies, that focuses on user-defined relationships between concepts to design new specialized ontologies.

From a data integration point of view, ontologies are useful in many ways, especially when dealing with semi structured and unstructured data integration:

- An otology can be used to play the role of a global schema, fed by the ontologies representing the sources schemas
- Ontology matching can be used to perform mapping and in record linkage and data fusion
- They are a support tool for content interpretation and wrapping
- An ontology can be a mediation support tool for content inconsistency detection and resolution
- An ontology functions as schema integration support tool
    - ❖ To represent semantics of schema elements
    - ❖ To guide conflict resolution

<u>Describe lightweight approaches to data integration using mashups as an example. Explain when it's appropriate to use and its advantages/disadvantages.</u>

When it's necessary to integrate data to answer a not-frequent question in a quick way and for it to be done by people with no technical expertise, lightweight integration is a way prioritizes speed over stability.

However, the result of lightweight data integration is limited to very specific reports or ad-hoc data analyses.

The advantages of using a lightweight data integration are the ease of development and the fast prototyping, but the approach cannot be used to answer to mission-critical integration requests, as they it can only provide short term "situational data".

Typical problems of lightweight data integration:

- locating relevant data sources
- assessing source quality
- helping the user understand the semantics
- supporting the process of integration.

Ideally, machine learning and other techniques can be used to amplify the effects of human input, through semi-supervised learning, where small amounts of human data classification, plus large amounts of additional raw ("unlabeled") data, are used to train the system.

Mashups are a paradigm for lightweight integration that work by integrating two or more mashup components at any of the application layer, potentially putting them in communication with each other, to respect the mashup logic of operations.

Describe concisely the concept of mashup and its utility in data integration, highlighting its distinctive features w.r.t. using other integration techniques.

A Mashup is an application for light-weight integration of two or more mashup components in a new way which can provide additional information, functions or visualization, according to the mashup logic of operations.

Components may be in communication with each other.

Two are the key elements of this application:

- A mashup component is any piece of data, logic or user interface which can be reused locally or even remotely.
- The mashup logic is the logic which specifies the invocation of components, the control flow, the data flow, the data transformations and the Ul of the mashup.

A mashup can be of different types:

- Data mashups: Fetch data from different resources, process them, and return an integrated result set
- Logic mashups: Integrate functionality published by logic or data components
- User Interface (UI) mashups: Combine the component's native UIs into an integrated UI
- Hybrid mashups: Span multiple layers of the application stack, bringing together different types of components inside one

Mashups introduce integration at the presentation layer, application layer and data layer.

A mashup generally differs from the other integration practices as it also lies on the logic and presentation layer as it applies integration also with respect on how the user perceive the contribute of different sources. They also typically focus on non-mission-critical applications and ad-hoc data analysis.

Data Warehouses questions

Define what is a Data Warehouse and describe the dimensional fact model used in the data warehouse context and define its main elements. Provide a small example.

A data warehouse is a single, complete and consistent store of data obtained from a variety of different sources, made available to end users so that they can use it in a business context.

It is also a process for transforming data into information and for making it available to users in a timely enough manner to make a difference.

Data present in data warehouses is historical data, and in form of aggregated and summary data, and is used to perform analysis inside the enterprise and to produce data forecasts.

A data warehouse is:

- Subject-oriented: to use in relation to a subject of interest
- Integrated
- Time-varying: keeps all the variations of data in time
- Non-volatile: data is only added, never deleted or updated (unless found to be wrong)

The logical data model in data warehouses is the dimensional fact model, which allows to describe a set of fact schemas. Its components are:

- Facts: concepts that are relevant for the decisional process, typically modelling a set of events of the organization.
- Measures: numerical properties of a fact
- Dimensions: fact properties defined w.r.t. a finite domain; they describe an analysis coordinates for the fact.
- Dimension hierarchies: to organize dimensions referring to facts (e.g. day -> month -> year)

A fact could be the sales of a store chain. The measures for the sales fact could be the sold quantity and the income, while the dimensions could be the time, the kind of products sold, the place etc.

Describe flow, level, and unitary measures in the data warehouse context. Provide a set of examples.

It's possible to identify three different measure categories:

- Flow measures: related to a time period and are evaluated in a cumulative way at the end of that period. (e.g. amount of sales per day, total income in a month, number of birthdays in a year)
  To compute flow measure, the used operators are SUM, AVG, MIN, MAX.
- Level measures: related to and evaluated in a particular time instant (e.g. amount of products in stock in a particular instant, number of people in a concert room)
- Unitary measures: evaluated in particular time instants but are relative measures (e.g. money change rate in a particular instant). Unitary measures cannot be aggregated.

Define the typical operations necessary in the multidimensional data model that is at the basis of data warehouses.

The multidimensional data model defines, for each combination of the dimensions of a fact, a value for its measures. The following OLAP operations are made available to the final users, which can query the DW with different purposes:

- Roll-up: Aggregates data at a higher level
- Drill-down: De-aggregates data at the lower level
- Slice-and-dice: Applies selections and projections, which reduce data dimensionality
- Pivoting: Selects two dimensions to re-aggregate data (cube re-orientation)
- Ranking: Sorts data according to predefined criteria
- Traditional operations such as select, project, join, derived attributes, etc.

<u>Define what is a Data Mart in a Data Warehouse and clearly summarize the methodological steps that lead from a collection of datasets to the specification of the logical schemas of one or more Data Marts.</u>

A data mart is a part of the Data warehouse dedicated to one specific subject of interest.

A Data Mart is a structure/access pattern specific to data warehouse environments, used to retrieve client-facing data.

There are three possible data models for Data Marts:

- MOLAP (Multidimensional OLAP) is the more traditional model, in which data is natively stored in a multidimensional cube in proprietary formats. MOLAP offers great performance in data retrieval and operations of the multidimensional data model. It can easily perform complex calculations that are pre-generated when the cube is created.
  However, there is a limit to the amount of data this model can handle, as it is not possible to include a lot when the calculations are generated when the cube is built. Indeed, data in the cube can be derived from a large amount of data, but only summary-level information will be included. This model also requires additional investment, as the cube technology are often proprietary.
- ROLAP (Relational OLAP) is the de-facto standard nowadays. The data are stores in a relational database but give the appearance of a traditional OLAP with dice and slice functionalities. Given its structure, it can accommodate as much data as the underlying database can, and it allows also operations that are proper of the relational model. On the other hand, some complex operations are not suited to be performed on a relational model, so tools to implement these functions need to be added, and, as every report is essentially a heavy query, performance may be slow.
- HOLAP (Hybrid OLAP) attempts to combine the two logical models. For summary-type information, HOLAP leverages cube technology for faster performance. When detail information is needed, HOLAP can "drill through" from the cube into the underlying relational data.

The logical modelling process includes a sequence of steps that, starting from the conceptual schema, workload, data volume and system constraints, allows to obtain the logical schema for a specific data mart.

The steps of ROLAP logical modelling are:

1. Choice of the logical schema (star/snowflake schema)
2. Conceptual schema translation
3. Choice of the materialized views
4. Optimization

The choice about views that have to be materialized takes into account contrasting requirements: cost functions' minimization, system constraints and users constraints. The objective is to reduce the workload and to keep the view maintenance as low as possible.

It is useful to materialize a view when it directly solves a frequent query or it reduces the costs of some queries. It's instead not useful its aggregation pattern is the same as another materialized view or its materialization does not reduce any cost.

<u>Describe and compare the Star Schema and the Snowflake Schema used in the data warehouse context.</u>

Star Schema and Snowflake schema are both used in the logical design phase of a Data Warehouse.

The Star Schema mimics the fact schema and consists of one or more fact tables referencing any number of dimension tables. Each dimension table is characterized by a primary key di and by a set of attributes describing the analysis dimensions with different aggregation levels. It is a simple schema, which leads to fast and simple queries. Tables are de-normalized which implies redundancy but fewer joins to perform and it is possible to define different variants of the star schema to manage aggregate data.

The Snowflake Schema reduces the de-normalization of the dimensional tables of a star schema, removing some transitive dependencies. All the attributes of a dimension table directly depend on the primary key, and there are zero or more external keys that allow to obtain the entire information following them to other dimensional tables. This schema reduces the memory space, removing data redundancy. It simplifies data update but queries are made more complex. To mitigate this problem, views can come in handy.

Data quality questions

List and describe the main dimensions of Data Quality

Data quality is the ability of a data collection to meet user requirements. To measure how much these requirements are satisfied we can make use of several dimensions, some more objective than others.

The main dimensions used in data quality assessment are:

- Accuracy: the extent to which data are correct, reliable and certified
- Completeness: the degree to which a given data collection includes the data describing the corresponding set of real-world objects
- Consistency: the satisfaction of semantic rules defined over a set of data items
- Timeliness: the extent to which data are sufficiently up-to-date for a task
- Ethics and fairness are also data quality dimensions

In most cases, the objective dimensions that are more complex to assess are also the ones with bigger business impact.

Explain the Data Cleaning process in the context of Data Quality.

Data Cleaning is a data-oriented improvement method used in the Data improvement phase.

It is the process of identifying and eliminating inconsistencies, discrepancies and errors in data to improve its quality.

The tasks of data cleaning are:

- Normalization/standardization (Datatype conversion, discretization)
- Missing values detection and imputing
- Outliers detection
- Duplicate detection

In particular, Duplicate detection (or entity reconciliation) is the discovery of multiple representations of the same real-world object. To perform it, a good similarity measure is needed, and it would be appropriate to minimize the number of comparisons as much as possible.

NoSQL questions

Define what is a NoSQL database. What does the CAP theorem state?

A NoSQL database is a non-transactional system, suited for non-critical applications, providing flexible schemas where updates are performed asynchronously and there are few constraints on data (ACID properties cannot hold). NoSQL systems are object-oriented friendly, because data is accesses via keys. They also make caching easier.

There are 4 main categories of NoSQL databases:

- Key-value
- Document-based
- Column-family
- Graph-based

The CAP theorem states that a data management system shared over a network can guarantee at most 2 of the following 3 properties at the same time:

- Consistency: all copies of data are synchronized
- Availability: every request receives a response about whether it was successful or it failed
- Partitions: the system is resistant to message loss or failures over part of it

However, these three properties are more "fuzzy" than binary and in some cases the choice between them can occur many times within the same system.

Some applications of NoSQL can be: data collected from sensors and datasets which are seldom updated.

Describe, in max 10 lines, the Column-family data model in NoSQL databases, highlighting the similarities and differences w.r.t. the basic Key-Value data model.

In the Column-family model the key is composed by a triple: (row, column, timestamp)

The data is partitioned horizontally (sharding) and vertically by the different columns stored in different machines.

This model offers the maximum scalability, and is so oriented to Big Data.

The queries allow to filter only on rows and column keys and the sorting is automatic.

Columns in the same column family will be "close": to be determined at design time.

With respect to the key-value model, both data models have "opaque" values and query on keys and share the GET, PUT, DELETE standard APIs. Key-value models can partition only horizontally and has limited scalability because of joins.

Multiple choice questions

Collection of some right answers to multiple choice questions.

<u>A data warehouse</u>:

- Is a data collection built with the aim of supporting decision-making processes
- Contains the history of data
- Is subject-oriented

<u>The design of a data warehouse</u>:

is driven by the queries that users will have to formulate

<u>The correct order for designing a data warehouse?</u>

Operational DB schema reverse engineering → Identify facts → Attribute tree → Fact schema → Glossary → Logical design

<u>Pruning consists in</u>:

deleting a leaf node or a leaf subtree of the attribute tree

<u>Grafting consists in</u>:

eliminating an internal node of the attribute tree, and attaching its children to the parent of the internal node removed

<u>An aggregation operator is</u>:

- distributive if it allows to compute aggregates starting from partial aggregates without needing further information
- algebraic if to compute aggregates from partial aggregates requires additional measures, which are named support measures

<u>In a data warehouse, the glossary</u>:

defines how to compute the measures starting from the operational database

<u>When implementing a data warehouse by means of the relational model, comparing the star schema to the snowflake schema</u>:

the star schema is more efficient in answering queries but requires more disk space

<u>What are the differences between the GAV (global as view) and LAV (local as view) approaches used to define the mapping between the global logical schema and the single source schemata in the data integration context</u>?

- LAV is when the sources are defined in terms of the global schema, while GAV is when the global schema is defined in terms of the sources.
- In GAV the global database (schema) is expressed as a virtual view of the local databases (schemas). In LAV each local schema, being a physical database, is a materialized view over the global schema.

<u>The correct order for doing Data Integration</u>:

Source schema reverse engineering → Related concept identification + Conflict analysis and resolution → Global conceptual schema design → Conceptual to logical translation of the global schema → Mapping definition

<u>Two data sources show heterogeneity of schema if</u>:

they adopt a different logical representation of the reality of interest

A data warehouse:
- contains the history of the data
- is subject-oriented

NoSQL databases:
- provide key-based access
- provide flexible schemas

Which of the following couples is correct?
- Name conflict → synonymies and homonymies
- Data semantics conflict → different currencies or units of measure

Which of these answers describes most accurately the difference between classification and clustering?
Clustering is the task of grouping a set of objects in such a way that objects in the same cluster are more similar to each other (w.r.t. to a given criterion) than to those in other clusters, while classification is the problem of identifying which of a set of categories an object belongs to

The KeyGen function:
- can be used to solve key conflicts
- is likely to be used when the primary keys for corresponding tables in two different sources are integer numbers auto-incremented

How can ontologies be useful in data integration:
- To support automatic understanding of the semantics of the instances for automatic entity resolution and data fusion
- Using an ontology as global schema instead of a relational one
- To support automatic understanding of the semantics of the schemas for schema integration

Choose, among the following ones, which are correct definitions of the corresponding data quality dimension.
- Timeliness – the fact that the data are sufficiently up-to-date for the objective they are used for
- Completeness – how closely the given dataset reflects the corresponding set of real-world objects
- Accuracy – how much data are reliable, correct, and certified

Which of the following definitions of the WITH QUBE and WITH ROLL UP operators are correct?
- WITH CUBE generates a result set that shows aggregates for all combinations of values in the selected columns
- WITH ROLLUP generates a result set that shows aggregates for a hierarchy of values in the selected columns

The virtual integration of data sources:
provides the possibility to query the data present in the data sources through a global schema

NoSQL database:
- provide key-based access
- provide flexible schemas