Chapter 2. State of the Art

This chapter presents an overview on data quality evaluation in data integration systems and focus on the evaluation of two major quality factors: data freshness and data accuracy. We conclude with a discussion on open research problems.

1. Introduction

Traditionally, information quality is described or characterized via multiple attributes or factors which help to rank the data delivered to users (e.g. freshness, accuracy, completeness) or the processes that manipulate this data (e.g. response time, reliability, security). Many lists of quality factors have been proposed, some of them containing a great number of quality definitions. For example, Redman identified four dimensions of data quality: accuracy, completeness, currency and consistency [Redman 1996] while Wang and Strong analyzed the various quality attributes from the user perspective [Wang+1996].

However, there is no consensus in the definition of quality factors, which may be overlapping and contradicting. Each application domain has its specific vision of data quality as well as a battery of (generally ad hoc) solutions to solve quality problems [Berty-Equille 2004]. Furthermore, even if quality factors are frequently treated as being independents, there exist lots of relationships among them. Several works studied some relationships for specific systems, for example [Ballou+1995] [Bright+2002] [Cappiello+2002] [Theodoratos+1999] [Han+2003].

The great number of quality factors and their inter-relations cause quality evaluation to be a complex problem of many variables. To improve the quality of a system corresponds to optimize a problem of N variables, which may have a great complexity if done in a general context. As a consequence, it is difficult to consider all quality factors at a time. In order to study data quality in depth, it is necessary to study separately each quality factor as well as the properties of the environment that impact in it. Relationships should be studied thereafter.

Among the quality factors that have been proposed, we choose two of the main families: data freshness and data accuracy. The following sections describe both of them: Section 2 presents an analysis of data freshness and its various underlying metrics, describing some dimensions that impact freshness evaluation as well as the relevant works proposed for dealing with freshness in several kinds of DIS. Analogously, Section 3 analyzes data accuracy, also describing metrics, dimensions that impact its evaluation and relevant works.

The analysis of data freshness and data accuracy suggests open problems in the specification of user expectations, the acquisition of source quality measures and the formulation of cost models for the DIS. This knowledge can be used for developing techniques for data quality auditing and techniques for designing a system driven by quality expectations. An overview of such research problems is presented at the end of each section. We conclude, in Section 4, positioning our work with respect to those research problems.

2. Data freshness

Data freshness has been identified as one of the most important attributes of data quality for data consumers [Shin 2003] [Wang+1996]. Some surveys and empirical studies have proved that data freshness is linked to information system success [Wang+1996] [Mannino+2004] [Shin 2003]. Then, achieving required data freshness is a challenge for the development of a large variety of applications. Furthermore, the increasing need to access to information that is available in several data sources introduces the problem of choosing among alternative data providers and of combining data having different freshness values.

This section presents an analysis of data freshness and its various underlying metrics. We analyze some dimensions that impact the evaluation and enforcement of data freshness and we present a taxonomy that summarizes this discussion. The taxonomy is used for the analysis and classification of existing work. At the end of the section we discuss some open problems.

2.1. Freshness definitions

Intuitively, the concept of data freshness introduces the idea of how old is the data: Is it fresh enough with respect to the user expectations? Has a given data source the more recent data? Is the extracted data stale? When was data produced?

But data freshness has not a unique definition in the literature. Several freshness definitions have been proposed for different types of data integration systems. The traditional freshness definition, called *currency* [Segev+1990], is related to view consistency when materializing data and describes how *stale* is data with respect to the sources. Recent proposals incorporate another notion of freshness, called *timeliness* [Wang+1996], which describes how *old* is data. Therefore, freshness represents a family of quality factors, each one representing some freshness aspect and having its own metrics. For that reason freshness is commonly mentioned as a *quality dimension* [Jarke+1999]. Each factor best suites a particular problem or type of system. For example, in a replication system the number of refresh operations that have to be applied to a replica relation in order to reflect the changes made in a master relation is a good freshness metric, but in a data warehousing system the time passed from the update of an object to its delivery may be more relevant. In this sub-section we present an analysis of data freshness definitions.

We distinguish two quality factors for this quality dimension:

- *Currency factor* [Segev+1990]: The currency factor expresses how stale is data with respect to sources. Data is extracted from sources, processed (possibly stored for some time) and delivered to the users, but source data may have changed since data extraction and the user may receive stale data. The goal is to return the same data that is stored at sources so currency captures the gap between data extraction and data delivery. For example, currency indicates how stale is the account balance presented to the user with respect to the real balance at the bank database.
- *Timeliness factor* [Wang+1996]: The timeliness factor expresses how old is data (since its creation/update at the sources). The age of the data may be not appropriate for a given application. The goal is to return data that is valid (not too old) so timeliness captures the gap between data creation/update and data delivery no matter when data was extracted from sources. For example, when retrieving a top 10 CD ranking, timeliness indicates how old such ranking is: when was it created and stored in the CD seller source.

Figure 2.1 illustrates the gaps each factor aims to capture.

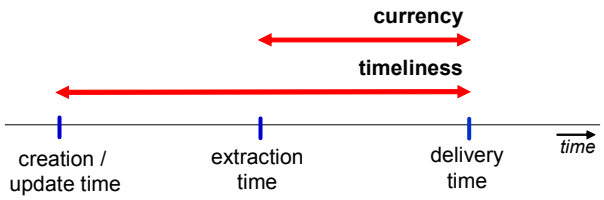

Figure 2.1 – Currency and timeliness factors

Note that both definitions are stated with respect to source databases. None of existing proposals compare with real world. The major argument is that capturing real world changes is not easy while capturing source changes is feasible (either querying, polling, subscribing…). However, for specific systems where real world changes are known both definitions can be enlarged to compare with real world data, i.e. how stale is delivered data (with respect to real world data) and how old is data (since its creation/update in real world). Examples of such systems are real-time registration of events (e.g. supermarket sales), sensor measurements and administrative documents (with specific timestamps).

2.2. Freshness measurement

A metric is a specific instrument that can be used to measure a given quality factor. There might be several metrics for the same quality factor. We describe the metrics proposed in the literature for measuring data freshness, classified by freshness factor:

- Metrics for the currency factor:
  - *Currency metric*: It measures the time elapsed since the source data changed without being reflected in the materialized view (if changes are propagated immediately then currency is 0) [Segev+1990]. In practice, as the precise change time may be difficult to obtain, currency is estimated as the difference between data extraction time and data delivery time. This estimation is used in data warehousing systems [Theodoratos+1999]. In caching systems it has been defined as *recency* or *age*, representing respectively the time elapsed since an object was cached [Bright+2002] and the time elapsed since an object became stale [Cho+2000]. In replication systems it is called *age* and measures the time since the oldest tuple of a relation has been waiting for the first refresh transaction [Gancarski+2003].
  - *Obsolescence metric*: It measures the number of updates to a source since the data extraction time. It can be measured from source logs or delta files or using change detection techniques [Hammer+1995]. Knowing the obsolescence of a source relation, the update frequency can be estimated and vice versa. Obsolescence is often called *age* in caching systems, meaning the number of times an object has been updated at the remote server since it was cached [Huang+1994]. In query processing systems, it is defined as the number of insertions, deletions and modifications since the data materialization time [Gal 1999]. In replication systems it is called *order* and measures the number of refresh transactions that have been committed in the master node but have not yet been propagated to the slave node [Gancarski+2003].
  - *Freshness-ratio metric*: It measures the percentage of extracted elements (tuples or attributes) that are up-to-date, i.e. their values equal the source relation ones. It can be estimated from the knowledge one has about data sources and from the way data is updated [Cho+2000]. It was defined as *freshness* in caching systems, meaning the number of elements of the cache that are up-to-date over the total number of elements [Cho+2000]. In [Labrinidis+2003], freshness is measured as the percentage of web pages that are fresh (not stale) in the cache but considering that the pages may have portions that are fresh and portions that are not.
- Metrics for the timeliness factor:
  - *Timeliness metric*: It measures the time elapsed since data was updated. It can be measured from source logs or delta files or using change detection techniques [Hammer+1995]. In [Braumandl 2003] it is measured from data timestamps, when they are available. Timeliness is generally estimated as the time elapsed from the last update to a source and bounded using the update frequency of the source data [Naumann+1999]. It was used in mediation systems [Naumann+1999] and web systems [Gertz+2004]. In [Ballou+1998], the metric is calculated as delivery time minus extraction time, adding the age of data at extraction time; some authors called it *currency*.

Previous data freshness metrics have been used as input for more complex indicators, for example, in [Ballou+1998] a freshness indicator is defined as[*]:

$$\max\{(1 - \text{timeliness} / \text{volatility}), 0\}^{s}$$

where volatility is measured in terms of shelf-life and the exponent *s* controls the sensibility of the ratio.

Table 2.1 presents a summary of freshness factors and their corresponding metrics.

Factor	Metric	Description
Currency	Currency	The time elapsed since data was extracted from the source (the difference between delivery time and extraction time).
	Obsolescence	The number of updates transactions / operations to a source since extraction time.
	Freshness ratio	The percentage of tuples in the view that are up-to-date (have not been updated since extraction time).
Timeliness	Timeliness	The time elapsed from the last update to a source (the difference between delivery time and last update time).

Table 2.1 – Summary of freshness factors and metrics

[*] In [Ballou+1998] the timeliness metric is called currency, and the composed indicator is called timeliness. We have changed their names for homogeneity purposes.

2.3. Dimensions for freshness analysis

In this sub-section we analyze some dimensions that impact the evaluation and enforcement of data freshness. We analyze the nature of data, the architectural techniques and synchronization policies of the underlying system. In next-subsection we present a taxonomy composed of these dimensions, which allows comparing different proposals for freshness evaluation.

Dimension 1: Nature of data

According to its change frequency, we can classify source data into three categories:

- *Stable data*: It is data that is improbable to change. Examples are scientific publications; although new publications can be added to the source, older publications remain unchanged. Other examples are person names, postal codes and country names.
- *Long-term-changing data*: It is data that has a very low change frequency. Examples are the addresses of employees, country currencies and hotel price lists in a tourist center.
- *Frequently-changing data*: It is data that has intensive change, such as real-time traffic information, temperature sensor measures and sales quantities.

The concept of "low frequency" is domain dependent; in an e-commerce application, if the stock of a product changes once a week it is considered to be low-frequency change while a cinema that changes its playbills weekly has a high-frequency change for spectators.

The nature of data is important because it is related to the notion of freshness that users are interested in. When working with frequently changing data, it is interesting to measure how long data can remain unchanged and minimize the delivery of expired data (i.e. evaluate currency). However, when working with stable or long-term changing data, these questions have no sense since data does not change very often. It is more interesting to measure how often new data is created or how old is the data (i.e. evaluate timeliness).

The changes can occur in a random way or with a defined frequency. For example restaurant menus, which are updated every morning, have a defined change frequency, but the account balances, which are updated with every account movement, have not got a defined frequency. In such cases, we can use data properties to develop specialized techniques, for example, synchronizing applications to extract data at the best moment.

Certain types of data have a lifecycle which describes explicitly its states and changing events. Some examples are the marital status of a person, the moon phases or the status of a semaphore. Sometimes, the events that make the states change are well known and can be predicted (as the semaphore). The fact that states are known in advance may allow the development of specialized techniques and treatments.

Dimension 2: Architectural techniques

The freshness of the data delivered to the user depends on *source data freshness* (the freshness of source data at extraction time) but also on the *execution delay* of the DIS processes (the amount of time from data extraction to data delivery). The DIS processes are very relevant in freshness evaluation because they can introduce significant delays. These delays may be relevant or not depending on freshness requirements. For example, in a given system, the evaluation of a query (minutes) is irrelevant compared to timeliness requirements (weeks), while in another system the aggregation processes may have the same order of magnitude of currency requirements (hours).

Specific cost models should take into account different parameters. These parameters depend on the system architectural techniques. We distinguish three main families of architectural techniques: those that calculate data when a new query is posed, those that cache the data most frequently used, and those that materialize the data needed to answer user queries. The features of these three categories of techniques are summarized below:

- *Virtual techniques*: The system does not materialize any data so all queries are calculated when they are posed. The system queries the relevant sources and merges their answers in a global answer that is delivered to the user. Examples are pure virtual mediation systems and query systems in database federations.
- *Caching techniques*: The system caches some information, typically data that is frequently accessed or the result of some frequent queries, and invalidates it when the time-to-live (TTL) has expired. If the

information required to answer a user query is stored in the cache, the system delivers it to the user; if not, the system queries the sources as in virtual systems. Examples are caching systems.

- *Materialization techniques*: The system materializes large volumes of data which is refreshed periodically. The users pose their queries and the system answers them using the materialized data. Examples are data warehousing systems and web portals that support materialization.

Virtual techniques allow to query sources and to return data immediately, so data is almost current. The processing and communication costs are the delays that influence currency. Caching techniques are conceived to return data as current as possible, estimating the TTL of each object for deciding when to invalidate it. However, materialized systems can tolerate some level of staleness. Data is stored for some time in the DIS repositories, which decreases its freshness; the refreshment frequency is an important delay.

Dimension 3: Synchronization policies

The way DIS are implemented influences the freshness of the data delivered to the users. Particularly, the synchronization among the sources, the DIS and the users has impact in data freshness because it introduces delays. For example, a DIS that synchronizes updates each end of the day may provide data which is not fresh enough with respect to the expectations of a given user.

According to the interaction among the DIS and the sources, the extraction processes can have *pull* or *push* policies. With pull policy, the DIS queries the sources to obtain data and with push policy, the source sends data to the DIS. In the latter, the notification of new available data can be sent by an active agent, for example initiated by a trigger, or can be determined by the DIS continuously polling the source. Active sources can have their own policies as sending each updated tuple, or sending sets of tuples every regular periods of time or when changes surpass a threshold. Pull policies can also be driven by temporal or non-temporal events.

According to the interaction among the DIS and the users, the query processes can also have pull or push policies. With pull policy, users directly pose queries to the DIS. With push policy users subscribe to certain queries and the DIS regularly conveys response data to the users. Pull and push policies can also be driven by temporal or non-temporal events.

Combining the previous interactions among users, DIS and data sources leads to six possible configurations which are shown in Figure 2.2. With synchronous policies, the user directly accesses source data. With asynchronous policies, the DIS answers user queries using materialized data and asynchronously, the materialized data is refreshed from source data. We name each configuration with the user-DIS policy followed by the DIS-source policy. Asynchronism is represented by a slash (/), synchronism by a dash (-):

- Pull-pull (arrow (a)): The interaction is fully synchronized. When a user poses a query (pull), it is decomposed and sent by the DIS to the sources (pull). This configuration is common in virtual mediation systems.
- Pull / pull (arrows (b) and (c)): When a user poses a query (pull) the DIS answers it using materialized data. Asynchronously, the DIS queries the sources in order to refresh materialized data (pull). It is common in data warehousing systems.
- Pull / push (arrows (b) and (e)): When a user poses a query (pull) the DIS answers it using materialized data. Asynchronously, the sources send data which refresh materializations (push). It is also used in data warehousing systems.
- Push / push (arrows (d) and (e)): When sources send data to the DIS (push), it is used to refresh the materializations. Asynchronously, the DIS conveys data to the users (push). It is used in publish/subscribe environments.
- Push / pull (arrows (d) and (c)): Materialized data is conveyed asynchronously to the users (push) and also asynchronously, the DIS queries the sources in order to refresh the materialized data (pull). It represents certain user applications (e.g. data marts) that are regularly fed from warehouse data.
- Push-push (arrow (f)): The interaction is synchronized. When sources send data to the DIS (push), the DIS conveys it to the users (push). This configuration is specific to some real time systems (alert systems) which capture events from sensors and conveys them to users but maintain also a history of these events. This configuration is not usually implemented in the three architectural techniques described previously.

Asynchronous policies introduce delays. The refresh frequency of the DIS repository is important to evaluate the freshness of retrieved data. When pushing data to the user, the push frequency is also important.

In systems where there are heterogeneous data sources with different access constraints and users with different freshness expectations, it is important to support and combine several kinds of policies.

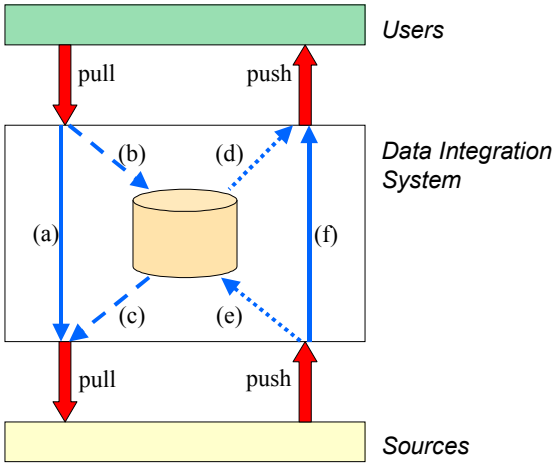

Figure 2.2 – Combination of synchronization policies

In next sub-section we present a taxonomy that relates these three dimensions summarizing their analysis.

2.4. A taxonomy for freshness measurement techniques

In this section we present a taxonomy that summarizes the discussion and allows comparing different proposals for freshness evaluation. The taxonomy is composed of the previously described dimensions: (i) nature of data, (ii) architectural techniques and (iii) synchronization policies.

Nature of data is a user-oriented dimension which qualifies the properties of source data from a user point of view. But not all the combinations of nature of data and freshness factors are interesting. On the one hand, when data changes very frequently, there is no interest in measuring timelines, which captures stable data behavior. On the other hand, there is no sense to evaluate the currency of long-term and stable data because they are almost always current as they do not change very often. In the latter case, the system can assure currency even without explicit evaluation. The other combinations need evaluation to determine the freshness level. For them, the development of evaluation tools is interesting. Table 2.2 shows the relation, indicating when data freshness can be assured without evaluation and when it is interesting to evaluate it (✓) or not (×).

Nature of data / *Freshness factor*	**Frequently changing**	**Long-term changing**	**Stable**
Timeliness	×	✓	✓
Currency	✓	assured	assured

Table 2.2 – Interesting combinations of freshness factors and nature of data

Architectural techniques, *synchronization policy* and *process complexity* are system-oriented dimensions which describe the system relation with data freshness. There is a correlation between *architectural techniques* and *synchronization policies*, since virtual techniques only support the synchronous pull-pull configuration, caching techniques are conceived for user pulls and materialization techniques support the asynchronous configurations. Table 2.3 shows the interrelations among them, indicating the valid combinations.

However, a priori, all combinations of *nature of data* and *architectural techniques* are possible, i.e. virtual, caching or materialization techniques (with their valid combinations of synchronization policies and process complexities) can be built to query different types of data.

Sync. policies / *Arch. techniques*	**pull-pull**	**pull/ pull**	**pull/ push**	**push/ pull**	**push/ push**
Materialized	×	✓	✓	✓	✓
Caching	✓	✓	✓	×	×
Virtual	✓	×	×	×	×

Table 2.3 – Valid combinations of architectural techniques and synchronization policies

In addition, system-oriented dimensions are also orthogonal to the freshness factors, i.e. users may be interested in both freshness factors, independently of the way the system is implemented. But taking into account the particularities of the systems and the availability of metadata and estimations, the different metrics are generally more related to some kinds of systems. For example, in virtual systems the main interest is the response time, so the currency metric is appropriate; in caching systems all the metrics have been identified as interesting, as different existent applications evaluate them [Bright+2002] [Huang+1994] [Cho+2000]; in materialized systems, currency and obsolescence have been used [Theodoratos+1999] [Gal 1999]. Table 2.4 shows the correlation among all the taxonomy dimensions and freshness metrics.

Nature of data / *Arch. techniques – Sync. policies*		**Frequently changing**	**Long-term changing**	**Stable**
Virtual	**Pull-pull**	Currency	Timeliness	Timeliness
Caching	**Pull-pull** **Pull/pull** **Pull/push**	Currency Obsolescence Freshness-ratio	Timeliness	Timeliness
Materialized	**Pull/pull** **Pull/push** **Push/pull** **Push/push**	Currency Obsolescence	Timeliness	Timeliness

Table 2.4 – Correlation of all the taxonomy dimensions

The technical problems to solve for each cell of the taxonomy are quite different. For example, enforcing currency in a materialized system implies developing efficient update propagation algorithms to deal with consistency problems, while evaluating timeliness in virtual systems is quite independent on the query rewriting algorithms and is dominated by source data timeliness. In Sub-section 2.6 we discuss evaluation problems.

2.5. Some systems that consider data freshness

In this sub-section we analyze several types of systems that evaluate freshness and we describe the goals and problems that they present. At the end of the sub-section, Table 2.5 summarizes the proposals in terms of the taxonomy presented before.

Data Warehousing systems

In data warehousing systems, freshness is studied through the *currency factor* in the context of view materialization.

The materialization of some views over source databases allows speeding up OLAP queries and reduces the overload of the sources. Traditional query optimization algorithms are extended to take into account the materialized views. In [Gal 1999], a cost model has been proposed for analyzing and comparing query plans, which can access to virtual and materialized data. The cost model strikes a balance between the query generation and data transmission cost on the one hand, and the *obsolescence* cost on the other hand.

Materialization introduces potential inconsistencies with the sources and warehouse data may become out-of-date [Hammer+1995]. The notion of consistency means that the DW state reflects an actual source state at some

"recent time" [Zhuge+1997]. The view maintenance problem consists in updating a materialized view in response to changes arisen at source data. Most of the work concentrates in assuring DW consistency for different types of views and refresh strategies. A classification of view maintenance algorithms is presented in [Gupta+1995]. A key problem in the last years has been the selection of a set of views to materialize in order to optimize the query evaluation and/or the maintenance cost, possibly in the presence of some constraints (commonly storage constraints). There are several works in this area [Harinarayan+1996] [Gupta 1997] [Theodoratos+1997] [Yang+1997] [Baralis+1997].

Data freshness is implicitly considered when defining the update propagation processes. Changes can be notified by active sources or can be monitored by the system accessing logs, querying the sources or comparing successive source snapshots [Widom 1995]. In most works, the update propagation processes are triggered by sources when the amount of changes is greater than a threshold or are executed periodically [Hull+1996] [Theodoratos+1997] [Baralis+1997] (pull/push and pull/pull policies). In [Theodoratos+1999], *data currency* is introduced as a quality factor in DW design. They propose an algorithm that takes as input the user expectations for data currency and determines the minimal update frequencies that allow achieving these values (pull/push policy).

Mediation systems

In classical mediation systems, freshness is also studied through the *currency factor*. In [Hull+1996], they formally define the concept of *guaranteed freshness*. A view is guaranteed fresh within a time vector, if it always corresponds to recent states of the source databases, that is, the difference between actual time and the time the view was calculated is always lower than the given time vector. Authors propose the construction of *Squirrel mediators*, which combine virtual and materialized data, and formally proof that they satisfy guaranteed freshness. Squirrel mediators combine pull-pull and pull/pull policies.

New proposals take into account the *timeliness factor*. It is used as a quality metric to compare among sources and to filter the data returned to the user. In [Naumann+1999], they introduce quality factors in a mediation system, which include *timeliness*. They study how to propagate a set of quality factors from several heterogeneous sources to the mediator. The propagation consists basically of merge functions that combine actual values of two sources through a relational operator. They propose a virtual scenario with pull-pull policy.

Caching systems

In caching systems, data is considered fresh when it is identical to data in the sources, so freshness is represented by the *currency factor*, and measured with the above mentioned metrics (currency, obsolescence, freshness-ratio).

An important problem is keeping cache data up-to-date. Traditional cache proposals manage the idea of *invalidation*. The system estimates the *time-to-live* (TTL) of an object as the time the object is supposed to be up to date, so the cache can store frequently changing data as well as long-term changing data. When the TTL has expired the object is invalidated in the cache, so the next access to the object will be directly read from the source and the cache will be refreshed (pull-pull and pull/pull policies). In some contexts the source can send invalidation information to the cache.

In [Cho+2000], they study the synchronization policies for cache refreshment and experimentally verify their behavior. They measure freshness with two metrics: currency (called age in the paper) and freshness-ratio. In [Li+2003a], they focus on the fine tuning of the caching policy of a dynamic content page cache, balancing response time and invalidation cycles for assuring data currency. In [Bright+2002], they propose the use of *latency-recency* profiles to adapt caching algorithms to user currency requirements, that is, if users demand more current data it will be extracted from the remote site paying communication times, but if currency of cached data is enough for user needs the user has an immediate response from the cache.

Newer proposals combine caching and materialization techniques. In [Labrinidis+2003], an adaptive algorithm has been proposed to combine view materialization and caching, balancing performance and data freshness. They combine a cache with server-side invalidation and a set of materialized WebViews (html fragments derived from a database) which are updated following a pull/push policy. As materialization degrades *currency*, the proposed algorithm selects which WebViews to materialize without exceeding a given currency threshold.

Replication systems

In a replication context, data at a slave node is totally fresh if it has the same value as the same data at the master node, i.e. all the refresh transactions for that data have been propagated to the slave node [Gancarski+2003]. Freshness is studied by means of the *currency factor* for frequently changing data.

A freshness model in a mono-master replication environment that supports OLTP transactions and OLAP queries has been presented in [Gancarski+2003]. Their goal is to determine the minimum set of refresh transactions needed to guarantee that a node is fresh enough with respect to the user freshness requirements for a given query. If data is not fresh enough, some refresh transactions are applied, then we can consider the mechanism as a cache invalidation method. The proposal consists in the evaluation of the freshness of slave nodes and the proposition of a load-balancing algorithm that takes freshness into account to decide when to refresh a replica. They follow pull-pull and pull/pull policies.

Works	Measurement	Nature of data	Arch. techniques	Synch. policy
Materialization for query processing [Gal 1999]	Obsolescence	Frequently changing	Virtual, materialized	Pull-pull, pull/pull
View maintenance [Gupta+1995] [Hammer+1995] [Zhuge+1997]	Currency	Frequently changing	Materialized	Pull/pull, pull/push
View maintenance policy [Theodoratos+1999]	Currency	Not specified	Materialized	Pull/pull
Selection of views to materialize [Harinarayan+1996] [Gupta 1997] [Theodoratos+1997] [Yang+1997] [Baralis+1997]	Currency	Frequently changing	Materialized	Pull/pull, pull/push
Mediation design combining virtual and materialized approaches [Hull+1996]	Currency	Not specified	Virtual, materialized	Pull-pull, pull/pull
Source selection in virtual mediation [Naumann+1999]	Timeliness	Not specified	Virtual	Pull-pull
Cache refreshment [Bright+2002] [Huang+1994] [Cho+2000] [Li+2003a]	Currency, obsolescence, freshness-ratio	Frequently changing / Long-term changing	Caching	Pull-pull, pull/pull, pull/push
Cache refreshment [Labrinidis+2003]	Freshness-ratio	Frequently changing	Caching, materialized	Pull-pull, pull/push
Replica refreshment [Gancarski+2003]	Currency, obsolescence	Frequently changing	Caching	Pull-pull, pull/pull

Table 2.5 – Summary of proposals

2.6. Research problems

Although data freshness has been studied in various ways in many papers, the analysis of the state of the art has shown that many problems remain unsolved or insufficiently treated. This sub-section summarizes these problems and mentions, when they exist, the references which have done significant contributions in each class of problem.

The freshness evaluation process needs to know about (i) the users' and source profiles, i.e. metadata about users' expectations and source properties, and (ii) the cost models used to extract source data, to maintain cached or materialized data and to evaluate query answers in different architectural configurations. Such elements should be adequately combined in order to evaluate the freshness of data conveyed to users. Freshness evaluation can be used for auditing an existing DIS or for designing a new DIS under quality constraints. In the following, we discuss these problems.

Defining users' and source profiles

Evaluating data freshness implies testing whether user's freshness expectations can be satisfied from source data freshness. One of the first problems is how and where to specify user expectations and how to define data source properties which impact data freshness.

Specification of freshness expectations

Users have freshness expectations for their applications which should be formulated according to some factors and metrics among those we have seen in Sub-sections 2.1 and 2.2. The specification of these factors and metrics pose a number of questions:

- Which language or formalism to use? Alternatives can vary from the simple specification of <property-value> pair [Bright+2002][Theodoratos+1999][Li+2003a] associated to each object type, to the definition of a specialized language if a preference order is introduced among freshness of different object types.
- At what level freshness expectations should be specified? We distinguish four levels: (i) for the whole system (for all users and data sources), (ii) for each data source (for all users), (iii) for each user or group of users, and (iv) for each user query. Each level implies different technical problems. When defining freshness expectations for the whole system or per source, individual user expectations should be reconciled. The expectations per user can be specified in a user quality profile. The definition of accurate profiles that allow users to understand the different freshness metrics and to express their expectations is an open problem. A first approach for introducing freshness in a user profile was presented in [Bright+2002]. Query languages such as *Preference SQL* [Kieβling+2002] can be extended to express freshness expectations in each user query.

Acquisition of source data freshness

The evaluation of data freshness at source level implies the selection of a freshness factor and the definition of metrics and measurement processes for it. The definition of new factors and metrics is an interesting area. Most existing works concentrate in the currency factor, but new types of DIS applications require the evaluation of other aspects of data freshness such as timeliness. Furthermore, several surveys have demonstrated the user interest in *timeliness* [Wang+1996] [Mannino+2004] [Shin 2003]. The acquisition of source data freshness poses some questions:

- Which source metadata is necessary to represent freshness in a source profile? In order to characterize source actual freshness some metadata should be obtained, for example the source update frequency [Cho+2003].
- How to acquire such metadata from sources? Some sources can provide useful information as the last update time or the update frequency, but for other sources these values must be learned or estimated from statistics elaborated during the data source exploitation. Existing techniques as comparing successive snapshots [Hammer+1995] or executing sampling queries [Jermaine 2003] can be adapted for freshness metadata acquisition. Techniques for specific metadata should be developed as in [Cho+2003].

Defining cost models

The freshness of the data delivered to the users depends on source actual freshness but also on the propagation delay from the sources to the user. Two main costs constitute this delay: the query evaluation cost and the update propagation cost. Depending on the taxonomy defined in Sub-section 2.4, these costs can be modeled differently:

Modeling of query evaluation cost: Query evaluation cost models for classical and distributed databases have been studied for a while and are well understood. Some proposals provide cost models for queries to cached or materialized data [Li+2003a] [Abiteboul+1998] [Cho+2003] and hybrid systems that combine materialization techniques in virtual or caching contexts [Labrinidis+2003] [Hull+1996]. Despite the existence of several proposals for specific systems, putting such capabilities to use in complex heterogeneous systems still requires modeling effort. Furthermore, existing cost models do not represent the cost of complex data processing involving ad hoc transformations and cleaning procedures such as in data warehousing systems. Several specialized tools, e.g. Ajax [Galhardas+2000] and Potter's Wheel [Raman+2001], require user interaction whose cost, although hard to estimate, should also be integrated.

Modeling of update propagation cost: Several cost models have been proposed to evaluate update propagation into materialized views [Chirkova+2001] [Yang+1997] [Hull+1996]. But as for query evaluation cost models, the update propagation cost models should be extended to represent complex workflow contexts with long transactions or interactive processes.

The challenge is the combination of all relevant parameters in a unified cost model in order to represent complex and hybrid architectures. Examples of such architectures are systems that extract data from heterogeneous sources with different synchronization policies and constraints. Some existing works combine several synchronization strategies [Segev+1990] [Theodoratos+1999] and temporal storage [Labrinidis+2003], but the land of hybrid systems is almost unexplored.

Several related questions are:

- Which DIS metadata is necessary to store in a DIS profile? Useful metadata will depend on the cost models used for modeling query evaluation cost and update propagation cost. For example, in [Hull+1996] several parameters are proposed for modeling the update propagation delay: announcement delay (amount of time between a source update and its announcement to the DIS), communication delay (amount of time needed for communicating with the source), update holding delay (amount of time between update announcement and start of update processing) and update processing delay (amount of time needed for processing an update). Large metadata should be studied for more complex cost models representing hybrid architectures.
- How to acquire such metadata from the DIS? For some metadata, as the query processing delay, acquisition techniques have been studied for classical and distributed databases [Abiteboul+1998] [Chirkova+2001]. Further metadata should be learned or estimated from statistics elaborated during DIS exploitation. Techniques for new types of metadata are also needed.

Auditing data freshness

Having users and source profiles and having cost models, one of the challenging problems is how to combine these elements in order to evaluate the freshness of the data conveyed to users. The main questions are: How should the different parameters of the source profile be combined for evaluating data freshness? What is the impact of update propagation cost and query cost in the freshness of data?

An additional question is how to combine several source actual values to obtain a global quality estimation. There are proposals only for specific environments. The combination of timeliness values within a virtual system is studied in [Naumann+1999]. The combination function is very simple (maximum of the input values) because they consider only simple views in a virtual context, but when dealing with materialization and complex calculation activities (which represent a bigger spectrum of DISs), other DIS properties (e.g. update propagation costs) should be considered. The combination of currency values within a materialized system is treated in [Theodoratos+1999]. They consider a DW architecture where source images are refreshed periodically and the propagation of changes to other materialized views follows push policies. Data currency is estimated adding the propagation cost of each materialized view and the refreshment period. But there are stills many combinations of freshness factors and types of systems for which there are no proposals. In particular, the combination of freshness values in hybrids systems, that manage data of different nature, different types of storage and synchronization policies is an open problem.

Freshness evaluation techniques can be used in the development of auditing tools, responsible for the evaluation of data quality. Several kinds of auditing tools can be conceived, for example:

- Prediction tools, for predicting the quality of data that can be returned in response to a query, without executing the query.
- Integrated evaluation tools, for measuring data quality during query execution and labeling delivered data with its quality levels. These tools can be integrated to the query evaluation process.
- Statistical tools, for taking samples of data quality during query execution and storing statistics. These tools can serve the first two categories.

The tools should use the query execution cost model and metadata describing the sources, the DIS and user expectations. They can be used at design time to evaluate the system, for example to test if user expectations can be achieved, or they can be used at run time for example, to predict the quality of the data delivered by alternative processes in order to choose the process that best suites user quality expectations.

Quality-driven engineering

Quality-driven engineering consists in designing a DIS under quality constraints. This kind of techniques may help in the selection among several design choices:

- Selection among alternative query plans that access alternative data sources.
- Specific optimization techniques as indexing and materialization
- Optimization of extraction and transformation algorithms.

Although several kinds of techniques have been explored in depth for specific systems, adapting their capabilities to heterogeneous systems requires addressing additional problems, as the combination of source data with different quality values, or even the comparison of data source profiles.

Obviously, quality-driven engineering techniques are based on quality auditing techniques, but the use of the latter ones may differ from evaluating an existing system and evaluating design alternatives. The challenge in quality-driven engineering is in the identification of the variables that influence data freshness and the proposition of techniques to achieve such improvements. There is few work done in this line, especially the study of synchronization policies [Cho+2000] [Li+2003a] [Segev+1990] or performance [Labrinidis+2003].

As improving freshness is not the unique quality goal for a given system, the relationship with other quality properties becomes a major challenge in DIS design. Research described in [Naumann+1999] has introduced quality factors to drive the design of virtual mediation systems; but quality factors are treated independently and only combined by means of a weighted sum.

The relationship among freshness and other quality properties has been only partially studied. There are two main lines: (i) tuning other properties in order to optimize freshness, and (ii) relaxing freshness in order to optimize other quality factors. In the former, the challenge is in the identification of related quality properties and the proposition of techniques that take advantages from these relationships in order to improve the global quality of data. Existing works in this line mainly concern the balance of freshness and performance [Labrinidis+2003] [Segev+1990] [Li+2003a]. In the latter, the expected freshness level is taken as a constraint. The main interest of existing works is performance or maintenance cost improvement [Bright+2002] [Gancarski+2003] [Theodoratos+1999] but the relation with other quality factors is still unexplored.

Discussion

Among the research problems previously mentioned, this thesis deals with data freshness auditing and freshness-driven engineering. Regarding data freshness auditing, we are interested in the combination of relevant parameters (source data freshness, cost models, user expectations) in order to evaluate the freshness of data conveyed to users. Users', sources and DIS profiles are taken as input. Contrarily to existing proposals [Naumann+1999] [Theodoratos+1999], we aim at considering the query evaluation and update propagation costs in a big spectrum of DISs, including those with data materialization and complex calculation activities. In addition, we are concerned with the tuning of DIS properties in order to enforce data freshness. Specifically, we focus in the proposal of improvement actions that can be used as building blocks for optimizing data freshness. Chapter 3 deals with data freshness evaluation and enforcement.

3. Data accuracy

Most data quality studies include accuracy as a key dimension for different types of DIS [Gertz+2004] [Shin 2003] [Vassiliadis+2000] [Mecella+2002] [Gertz+1998] [Missier+2001]. However, although the term has an intuitive appeal, there is no commonly accepted definition of what it means exactly. For example, in [Naumann+1999], accuracy is characterized as "the percentage of objects without data errors such as misspellings, out-of-range values, etc.", and in [Redman 1996], it is described as "the degree of agreement between a collection of data values and a source agreed to be correct". Then, accuracy represents a family of quality factors, or a quality dimension, with different associated metrics. Each factor best suites a particular problem or type of system.

This section presents an analysis of data accuracy and its various underlying metrics. We analyze some dimensions that impact the evaluation and enforcement of data accuracy and we present a taxonomy that summarizes this discussion. The taxonomy is used for the analysis and classification of existing work.

3.1. Accuracy definitions

Data accuracy is concerned with the correctness and precision with which real world data of interest to an application domain is represented in an information system [Gertz+2004]. Intuitively, the concept of data accuracy introduces the idea of how precise, valid and error-free is data: Is data in correspondence with real world? Is data error-free? Are data errors tolerable? Is data precise enough with respect to the user expectations? Is its level of detail adequate for the task on hand?

Data accuracy has not a unique definition in the literature. The most common definitions concern how correct, reliable and error-free is the data [Wang+1996] and how approximate is a value with respect to the real world [Redman 1996]. But there are various definitions (and variants of such definitions) concerning different concepts and metrics, which are mainly due to the different objectives of the systems where they are used. Furthermore, as data accuracy is highly related to other quality factors (e.g. data completeness, data consistency and data freshness), definitions are commonly confusing and overlapping. In this sub-section we present an analysis of data accuracy definitions.

Data accuracy comprises a family of quality factors, each one representing some accuracy aspect and having its own metrics. For that reason accuracy is commonly mentioned as a quality dimension [Jarke+1999]. We distinguish three quality factors for this quality dimension:

- *Semantic correctness factor*[*]: It concerns the correctness and validity degree of the data [Wang+1996] [Pipino+2002], describing how well data represent states of the real-world [Shanks+1999]. The concept of correctness implicitly or explicitly involves a comparison with the real world or with reference data agreed to be correct. It captures the gap (or the semantic distance) between data represented in the system and reference data. For example, the degree of concordance between the set of students that assist to a course and the list of students enrolled into this course.

  According to this definition, every set of data should represent a real world situation [Bobrowski+1998]. Different forms of incorrectness are: data without real world correspondent (mismembers), data referencing a wrong correspondent and data with erroneous attribute values [Kon+1995]. For example, data about a student may reference an inexistent person, reference an incorrect person or reference the correct person but having erroneous attributes (e.g. telephone).

- *Syntactic correctness factor*[†]: It expresses the degree to which a set of data is free of syntactic errors such as misspellings [Naumann+1999] and format discordances. Data is argued to be correct, in a syntactic way, if it satisfies syntactic rules and constraints imposed by users. Such constraints are not necessarily defined as integrity constraints of source databases, and generally cannot be added to data sources because of source autonomy; however, they correspond to the rules and constraints that DIS users expect that data verifies. Examples of constraints are: "inter-company telephone numbers have 4 digits" or "streets names must be registered in a street catalog". Typical constraints assure that data is always presented in the same format and is compatible with previous data [Wang+1996].

- *Precision factor*[‡]: It concerns the quantity of data to be stored and how precise this data must be [Redman 1996]. Data precision involves the level of detail of data representation [Bobrowski+1998] [Bouzeghoub+2002]. For example, the weight of a person may be stored in kilos (e.g. 88 Kg.) or with a greater precision (e.g. 88,25 Kg.). Analogously, a birthday may be represented by a year (e.g. 1973), by month and year (e.g. September 1973), a date (e.g. 19/9/1973) or even including time. In all cases there is a partial hierarchy inside each domain that allows deciding if a representation is more precise than another.

Note that considering the nature of the definitions, the semantic correctness and syntactic correctness factors attempt to quantify the quantity of data errors while the precision factor intends to quantify the precision of data. However, considering the type of comparison, the syntactic correctness and precision factors verify rule

[*] In the literature, this quality factor has been called *semantic correctness* [Missier+2001], *correctness* [Bobrowski+1998], *semantic accuracy* [Fugini+2002], *accuracy* [Wang+1996] [Gertz+2004] [Redman 1996] [Shanks+1999] [Wand+1996] [Naumann+2001] [Missier+2003] [Altareva 2004] [Gertz+1998], *free-of-error* [Pipino+2002], *soundness* [Motro+1998], *validity* [Müller+2003] and *data quality* [Ballou+1998]. Another alternative name is *precision* [Motro+1998].

[†] In the literature, this quality factor has been called *syntactic correctness* [Missier+2001], *correctness* [Missier+2003], *syntactic accuracy* [Fugini+2002], *accuracy* [Naumann+1999], *structural consistency* [Redman 1996], *representational consistency* [Wang+1996], *format consistency* [Missier+2003], *consistency* [Schurr+2002] and *schema conformance* [Müller+2003].

[‡] In the literature, this quality factor has been called *precision* [Bobrowski+1998] [Missier+2003] [Motro 1995], *accuracy* [Gertz+1998], *level of detail* [Redman 1996] and *ambiguity* [Wand+1996].

satisfaction focusing on syntactic aspects (data representation and constraints) while the semantic correctness factor compares to real world focusing on semantic aspects (data significance).

The following example will be used along the document:

Example 2.1. Consider an information system that handles information about students (Table 2.6) and consider real-world data about students (Table 2.7). Attributes describing students are: stid (the student identification number), name, address, telephone, interview (initial level determined by interviews; taking values 'high', 'medium' or 'low') and test (initial test result; taking values between 0 and 1). The key of the relation is {*stid*}.

Considering semantic correctness, stid 21 and 22 do not exist in real-world (mismembers), and stid 58 references the wrong student. Other students present matching errors or null values in some attributes (value inaccuracy), for example: student 43 has wrong address and telephone and student 102 has no registered address. However, note that even not written in the same way, some attribute values are good, e.g. the name of student 103 and the address of student 57. The address of student 56 is also ok; discrepancies are because the street name was changed some years ago, but postal office keeps the old name so address can be found.

Regarding syntactic correctness, note that some names and addresses do not follow standard rules, e.g. name of student 103 and addresses of students 21 and 57. The address of student 22 is not in an address catalog because of typing errors (*Coloniaa* instead of *Colonia*), and the test attribute of student 58 has an out of range value (9).

Finally, regarding precision, note that some test values are rounded (e.g. student 57) leading to lose of precision while other ones are more precise. The address of student 21 is also imprecise (*Carrasco* is the name of a neighborhood, not a detailed address). □

stid	name	address	telephone	interview	test
21	María Roca	Carrasco	6001104	low	1.0
22	Juan Pérez	Coloniaa 1280/403	9023365	medium	.5
43	Emilio Gutiérrez	Irigoitía 384	3364244	high	.8
56	Gabriel García	Propios 2145/101		low	.5
57	Laura Torres	Maldonado & Yaro	099628734	medium	.7
58	Raúl González	Rbla Rca Chile 1280/1102	4112533	high	9
101	Carlos Schnider	Copacabana 1210	094432528	high	.9701
102	Miriam Revoir		9001029	medium	.7945
103	A. Benedetti	Charrúa 1284/1	7091232	low	.9146
104	Luis López	Sixtina s/n		high	.8220

Table 2.6 – Students relation

stid	name	address	telephone	interview	Test
43	Emilio Gutiérrez	Potosi 934	6019945	high	.8000
56	Gabriel García	Batlle y Ordoñez 2145/101	5143029	low	.5130
57	Laura Torres	Maldonado 864	099628734	medium	.6965
58	Horacio Acher	Soca 2315	7079428	medium	.7600
59	Renzo Quinteros	Juan Paullier 635/001	4037690	low	.5505
101	Carlos Schnider	Copacabana 1210	094432528	high	.9701
102	Miriam Revoir	Canelones 1524	9001029	medium	.7945
103	Ana Benedetti	Charrúa 1284/1	7091232	low	.5146
104	Luis López	Sixtina s/n		high	.8218

Table 2.7 – Real world students

Other quality factors related to accuracy

Data accuracy is a quality measure for the relative amount of erroneous data accessed and manipulated by a system. In the past, accuracy was known as "data quality" and proposals devoted to evaluate and enforce data quality centered in accuracy (see for example [Ballou+1998]). For that reason, several accuracy definitions include other quality aspects as completeness or freshness, i.e. incomplete and expired data is considered inaccurate. In [Redman 1996], Redman distinguishes four quality dimensions for data values: *accuracy* (in the sense of correctness), *freshness* (in the sense of timeliness)[*], *completeness* and *consistency*, but remarks that accuracy is the most fundamental one, with freshness, completeness and consistency being special cases.

In this sub-section we discuss larger definitions of data accuracy and its relation with other quality factors.

Accuracy and Completeness

Completeness is commonly defined as the degree to which all data relevant to an application domain have been recorded in an information system [Gertz+2004]. It expresses that every fact of the real world is represented in the information system [Bobrowski+1998]. It is possible to consider two different aspects of completeness: (i) whether all required entities for an entity class are included, and (ii) whether all data values are present (not null) for required attributes. In [Redman 1996], the first aspect is called *entity completeness* and the latter *attribute completeness*, but other authors use different terms. Most works of literature define completeness as one or both of the previous definitions.

Some works consider accuracy and completeness as a same family of problems. In [Kon+1995], authors study error measurment and classify them into three categories: value innacuracy (errors or null values in some attributes), class mismembership (system data references inexistent real-world data) and class incompleteness (there is real-world data not referenced). They treat null values as a case of innacuracy and inexistent data as incompleness. Analogous definitions are presented in [Parssian+1999].

Accuracy and Consistency

Consistency expresses the degree to which a set of data satisfies a set of integrity constraints [Redman 1996]. Data is argued to be consistent if it satisfies the constraints[†]. Examples of constraints are: "the age of an employee accords its birthday year" (e.g. "Paul has 18 years" is inconsistent with "Paul was born in 1943") or "employee codes are unique". The most common constraints checks for null values, key uniqueness, duplicates and functional dependencies. In this sense, constraints state that two or more values do not conflict each other [Mecella+2002]. In particular, duplicate detection is one of the current challenges.

Semantic correctness (as previously defined) is very difficult to check, however, automated checking of constraints is feasible. In this direction, consistency checks provide a cost-effective way to identify data values that are not valid, knowing that satisfaction of constraints does not assure that data is semantically correct. Redman says that two data values are inconsistent if they cannot be correct simultaneously, so argues that defining consistency rules helps in the identification of data values that cannot be correct [Redman 1996]. The reader should note that such consistency rules may be defined for the integrated data (to be delivered to users), so, they may be not defined as constraints in the sources. While in traditional database systems all consistency rules are automatically checked by database engine and tuples not verifying them are rejected from the database, in DIS, consistency rules may represent expected rules (and may be not implemented as constraints), so the achievement of constraints can be measured as other quality factors (e.g. as a rate).

Accuracy and Freshness

As data values change with time, a lag between the time real-world data changes and the time changes are represented in the system is inherent. Data timeliness measures this lag, expressing the degree to which the recorded data are up-to-date.

[*] Timeliness is called currency in [Redman 1996]

[†] Analogously to syntactic constraints, integrity constraints are not necessarily defined in the source databases. They correspond to the constraints that DIS data is expected to verify.

Data changes have also an impact in data accuracy. As argued in [Redman 1996], a datum value is up-to-date if it is correct in spite of a possible discrepancy caused by time-related changes to the correct value; a datum is outdated at time *t* if it is incorrect at *t* but was correct at some time preceding *t*. In this sense, being out of date is simply a specific type of inaccuracy.

In next sub-section we analyze accuracy metrics for measuring accuracy quality factors.

3.2. Accuracy measurement

In this sub-section we describe accuracy metrics. We firstly introduce three types of metrics for measuring the accuracy of individual data items* and we discuss some aggregation functions for them; then, we explain the specific metrics that have been proposed in the literature.

We highlight three types of metrics:

- *Boolean metric*: It is a Boolean value (1=true, 0=false) that indicates if a data item is accurate (correct, precise) or not [Motro+1997].
- *Degree metric*: It is a degree that captures the impression or confidence of how accurate is data [Laboisse 2005]. Such degree is commonly represented in the [0-1] range.
- *Value-deviation metric*: It is a numeric value that captures the distance between a system data item and a reference one (e.g. its real-world correspondent entity) [Kon+1995]. Such distance is generally normalized to the [0-1] range.

In order to provide the user with a global accuracy measure for a whole relation (or a query result), an aggregated accuracy measure has to be synthesized. Given a set of data items S, with n elements (|S|=n)†, and a set of accuracy values $\{a_i\}$ where a_i is the accuracy value of the i-th element of S, $1 \le i \le n$, aggregation functions build a synthesized accuracy value for S. Typical aggregation functions are:

- *Ratio*: This technique calculates the percentage/ratio of accurate data of the system [Motro+1998]. This percentage is calculated as the number of accurate data items in the system divided by the total number of data items in the system. The accuracy of data items is expressed with Boolean metrics, i.e. $a_i \in \{0, 1\}$, $1 \le i \le n$. The accuracy of S is calculated as:

  $$Accuracy_{Ratio}(S) = |\{a_i / a_i = 1\}| / n$$

  A generalization can be done for the other types of metrics, considering the number of data items whose accuracy values are greater than a threshold θ, $0 \le \theta \le 1$:

  $$Accuracy_{Ratio}(S) = |\{a_i / a_i \ge \theta\}| / n$$

- *Average*: This technique calculates the average of the accuracy values of data items [Ballou+1998]. The accuracy of data items can be expressed with any type of metric. The accuracy of S is calculated as:

  $$Accuracy_{Avg}(S) = (\Sigma_i\ a_i) / n$$

  This technique is the most largely used, for the three types of metrics. Note that if accuracy of data items are Boolean values the aggregated accuracy value coincides with a ratio.

- *Average with sensibilities*: This technique uses sensibilities to give more or less importance to errors [Ballou+1998] and calculates the average of the sensitized values. Given a sensitivity factor α, $0 \le \alpha \le 1$, the accuracy of S is calculated as:

  $$Accuracy_{Sens}(S) = (\Sigma_i\ a_i^{\alpha}) / n$$

- *Weighted average*: This technique assigns weights to the data items, giving more or less importance to them [Ballou+1998]. Given a vector of weights W, where w_i corresponds to the i-th data item, $\Sigma_i\ w_i = 1$, the accuracy of S is calculated as:

  $$Accuracy_{weight}(S) = \Sigma_i\ w_i\ a_i$$

---

* Generally, the term *data item* refers to an attribute of a tuple (a cell), but in some application contexts it refers to the whole tuple.

† |A| denotes the cardinality (number of elements) of the set A.

Considering the three types of metrics and the aggregation functions, we describe the metrics proposed in the literature for measuring data accuracy, classified by accuracy factor.

- Metrics for the *semantic correctness* factor:
  - *Semantic correctness ratio metric*: It measures the percentage of semantically correct data in the system [Motro+1998]. This percentage is calculated as the number of system data items that match real world data divided by the number of system data items. This metric has been used in several works, for example [Shankaranarayan+2003] [Redman 1996] [Motro+1998].

    In practice, comparing all data against real world may be not viable, so correctness-ratio is commonly estimated via sampling. A portion of system data is taken as a sample, which is validated against real-world [Motro+1998] or a reference system considered reliable enough [Missier+2003]. For some types of data as postal addresses, telephone numbers or emails, there exist referential catalogs. When there are several available data sources, the most reliable one is usually taken as reference to perform the comparison; the appearance frequency could also be taken into account to decide whether data is correct or not [Shankaranarayan+2003] [Fugini+2002]. Other strategies consist in taking into account other quality factors to determine the most reliable data source, e.g. data consistency [Redman 1996]. Source providers or domain experts can also provide error ratio estimations based on their knowledge/experience with the data. Other specific methods are currently used to perform further validations [Laboisse 2005]; they include enquiries, automatic-generated emails or telephone calls to validate data with customers, even if such methods are very expensive and time consuming.

    Additional metrics have been defined to measure special cases of inaccuracies [Parssian+1999] [Kon+1995]:

    - *Mismembership ratio metric*: It measures the percentage of mismembers, i.e. the percentage of system data without correspondent in real-world.
    - *Value inaccuracy ratio metric*: It measures the percentage of system data containing errors in some attributes values or containing null values.
  - *Semantic correctness degree metric*: It captures the impression or confidence of how correct is data [Laboisse 2005]. The calculation can be done manually by a domain expert or it can be estimated based on historical data or statistics. Most common degrees are in [0-1] range (or translations) or any ad-hoc classification such as “good”, “medium” and “low”.

    This metric is typically used in automatic input-processing systems, e.g. optical character recognition, image recognition and address searching. For example, in optical character recognition applications, each symbol is compared to the alphabet characters; the most similar one is returned with its similarity degree. For example, for a certain application the three symbols of Figure 2.3 may be recognized as the character “C”, but the confidence of such recognition may be 80, 100 and 65 respectively.

    In order to compute the correctness degree of a set of elements, weighted average is typically used as an aggregation function, assigning different weights to the attributes according to their relative importance [Laboisse 2005].

Figure 2.3 – Character recognition examples

  - *Semantic correctness deviation metric*: It measures the semantic distance between a system datum and its real-world correspondent datum [Kon+1995] [Fugini+2002]. The calculation of such distance depends on the data type and the application, for example, for numeric data, it can be calculated as the difference between values (normalizing or not) [Shankaranarayan+2003] or for string data counting the number of characters to change (add, delete or modify) [Navarro 2001] or considering the soundness of words [USNARA 2000]. Values are generally normalized (translated to the [0-1] interval).

    In practice, if a comparison with real-world data is not possible, the comparison is done against a reference value which can be obtained from other data source or synthesized from several source values, e.g. using statistics of appearance frequency [Morey+1982] or taking an average value [Shankaranarayan+2003] [Fugini+2002].

For some attributes (belonging to a discrete domain), a similarity relationship among domain values may be defined, indicating to what extent each pair of labels resemble each other. When such relationship is defined, the similarity degree can be used as accuracy estimation. For example, Table 2.8 shows a possible similarity relationship for the interview attribute of the Students relation of Example 2.1, so the accuracy of the interview value of student 58 is 0.5. A similarity relationship can be defined by users according to its expectations or can be taken from some domain convention (e.g. color similarity ratios). Some types of fuzzy values (those of a discrete non-ordered dominion) have associated a similarity relationship (see [Galindo+2004] for a description of fuzzy values).

In order to compute the value-deviation of a set of elements, simple or weighted averages are typically used as aggregation functions, in the latter assigning different weights to the attributes according to their relative importance [Motro+1998].

	low	medium	high
low	1	0.5	0
medium	0.5	1	0.5
high	0	0.5	1

Table 2.8 – Similarity relationship among values of the interview domain

- Metrics for the *syntactic correctness* factor:
  - *Syntactic correctness ratio metric*: It measures the percentage of syntactically correct data of the system [Pipino+2002]. This percentage is calculated as the number of system data that satisfies syntactical rules divided by the number of system data. This metric is largely used in the literature; see for example [Naumann+2000] [Pipino+2002] [Naumann+1999].

    Note that when evaluating semantic correctness metrics, when the tuple is a mismember, all the attribute values are inaccurate and when the key is not accurate most of the attribute values are inaccurate (except for hazard coincidences). This is the desired effect. So, semantic correctness metrics evaluate if an attribute corresponds to the real world object represented by the key. However, when evaluating syntactic correctness, the metric semantics depends on the precise rules; for example, syntax constraints or belonging to a range can be checked independently of the key attributes.

    The most typical syntactical rules checks for illegal values (e.g. out-of-range), non-standard format or embedded values (e.g. “Paris, France” in a *city* attribute).

  - *Syntactic correctness deviation metric*: It measures the syntactic distance between a system datum and some neighbor data that is syntactically correct [Fugini+2002]. The calculation of such distance (and neighbor determination) depends on the type of constraint, the data type and the application, for example, for string domain conformity checks (data belongs to a catalog) the distance to most similar element can be used [Navarro 2001] [Gravano+2001] [USNARA 2000] or for out of range checks the distance to the nearest range can be calculated. As an example, consider the *Name* attribute of Table 2.7 as a reference catalog for students’ names. The nearest element for “A. Benedetti” is “Ana Benedetti” and the value-deviation metric can be calculated using some edit distance function.

    In order to compute the value-deviation of a set of elements, simple or weighted averages are typically used aggregation functions, assigning different weights to the attributes according to their relative importance [Laboisse 2005].

- Metrics for the precision factor:
  - *Scale metric*: For numeric values, precision is commonly associated to the measurement scale (or error-ratio of the measurement instrument). For example, if the length of a certain table is 87 ±1 cm, then imprecision is ±1 cm. A relative imprecision metric can be obtained dividing by the data value, i.e. 1/87; the precision metric is obtaining subtracting imprecision from 1, i.e. 1 – 1/87. When the error-ratio is not given, a confidence degree in the measurement process can be used, generally measured as the units of the least significant digit of a measurement, e.g. in 17,130 meters imprecision is millimeters (0,001 m) [Wikipedia 2006].
  - *Standard error metric*: In contexts where a data item is obtained taking several measures of some phenomena (e.g. temperature or traffic flow), imprecision is usually characterized in terms of the standard deviation of the measurements, called the measurement process's standard error [Wikipedia

2006]. This metric, even largely used in scientific environments, is rarely used in data integration systems because metadata about data production is not frequently available.

- *Granularity metric*: It involves the number and coverage of attributes that are used to represent a single concept [Redman 1996]. For example, an address may be represented by only a country name (e.g. Uruguay) or by a set of attributes like street name, door number, city, postal code and country (e.g. Cubo del Norte, 3840, Montevideo, 11700, Uruguay). The second set of attributes provides a more granular view of data. A simple metric consists in counting the data values (non-null values) representing a concept.

Table 2.9 summarizes accuracy factors and their corresponding metrics.

Factor	Metric	Description
Semantic Correctness	Semantic correctness ratio	The percentage of system data that match real-world data.
	Mismembership ratio	The percentage of system data without correspondent in real-world.
	Value inaccuracy ratio	The percentage of system data containing errors in some attributes representation.
	Semantic correctness degree	The confidence (degree) on the correctness of data
	Semantic correctness deviation	The semantic distance between a system datum and its correspondent real-world datum.
Syntactic Correctness	Syntactic correctness ratio	The percentage of system data that satisfies syntactical rules.
	Syntactic correctness deviation	The syntactic distance between a system datum and a reference one considered as syntactically correct.
Precision	Scale	The precision associated to the measurement scale.
	Standard error	The standard deviation of a set of measurements.
	Granularity	The number of attributes used to represent a single concept.

Table 2.9 – Summary of accuracy factors and metrics

3.3. Dimensions for accuracy analysis

In this sub-section we analyze some dimensions that impact the evaluation and enforcement of data accuracy. We analyze the granularity of the measurement, the typology of errors, the data types and the architectural techniques of the underlying system. In next-subsection we present a taxonomy composed of these dimensions, which allows comparing different proposals for accuracy evaluation.

Dimension 1: Granularity of measurement

The techniques for accuracy acquisition may vary according to the evaluation granularity. Traditionally, a global accuracy measure was used to qualify a whole relation [Kon+1995] or a view over a relation [Naumann+1999]. In addition, some measurement techniques consider the relationship among relations [Rahm+2000]. Recent proposals, especially industrial works, focus on the measurement of the accuracy of individual cells* [Laboisse 2005]. We describe these three levels of granularity:

- *Cell granularity*: In this approach, an accuracy value is associated to each cell, measured either as a Boolean value (accurate / not accurate), a deviation from the correct value or a degree of accuracy. The accuracy measure is generally obtained evaluating each cell of the relation. This is a very costly task but industrial investment in accuracy measurement and improvement [Laboisse 2005] [Amat+2005] [Graveleau 2005] indicate its importance despite the cost.

* The term *cell* refers to an attribute of a tuple.

- *Set granularity*: In this approach, a unique accuracy value is estimated for a whole relation or view, typically measured as the number of accurate cells over the number of cells (ratio metric). The accuracy measure can be obtained evaluating each cell of the relation or each cell of a sample; statistical techniques can also be used. In several cases, the experience or confidence of experts can be used as estimation.
- *Relationship granularity*: In this approach, a unique accuracy value is associated to the relationship among relations or views, typically measured as the number of accurate cells over the number of cells (ratio metric). The accuracy measure is generally obtained applying arithmetic operations to the accuracy of relations or views (e.g. multiplying such values [Naumann+1999]).

The main problem of associating an accuracy value to a whole relation is that it does not show where inaccuracies are concentrated (some attributes, some sets of tuples). As argued in [Motro+1998] information sources are rarely of uniform quality, so a unique accuracy value may be a very crude estimation of the accuracy of specific data. On the other hand, cell granularity is sometimes too detailed, as sets of attributes or tuples may have the same behavior. This causes great overhead in measurement and excessive additional storage.

An intermediate solution consists in partitioning the source relation in areas that are highly homogeneous with respect to their accuracy [Motro+1998] and assigning an accuracy value to each area (thus remaining in set granularity). Homogeneity means that any sub-area of a highly homogeneous area would maintain roughly the same accuracy as the initial area. Areas are defined as views, which may involve selection and projection. Partitioning may be done by source providers considering their knowledge about data, for example which attributes are more probable to have errors or which sets of tuples (e.g. those corresponding to foreign sales) are more probable to be inaccurate. In [Motro+1998], Motro and Rakov propose an algorithm to perform partitioning in an automatic way, testing different partitioning criteria. Even if the algorithm has a great complexity, heuristics can be considered for specific cases. Furthermore, authors argue that the partitioning could be done only once, at design time, so the cost may be tolerated. The accuracy measure for each area can be obtained in the same way as for the whole relation.

Example 2.2. Consider the *Students* relation of Example 2.1 and the measurement of semantic correctness on it. The source database administrator provides the following information:

- Addresses and telephones of old students (id < 100) is very inaccurate (accuracy=0.25) because of data obsolescence.
- Stid, name, interview and test of old students have accuracy of 0.50 due to typing errors and lack of automated checking in an old information system.
- Data about new students (id ≥ 100) is almost accurate (accuracy=0.90).

A horizontal partition with two areas can be defined, with the following predicates: (i) stid < 100, and (ii) stid ≥ 100. The first area is also vertically partitioned in two sub-areas for the {stid, name, interview, test} and {address, telephone} sets of attributes respectively. Table 2.10 illustrates the partitioning coloring areas (and sub-areas) with different colors. □

stid	Name	address	telephone	interview	test
21	María Roca	Carrasco	6001104	low	1.0
22	Juan Pérez	Coloniaa 1280/403	9023365	medium	.5
43	Emilio Gutiérrez	Irigoitía 384	3364244	high	.8
56	Gabriel García	Propios 2145/101		low	.5
57	Laura Torres	Maldonado & Yaro	099628734	medium	.7
58	Raúl González	Rbla Rca Chile 1280/1102	4112533	high	.9
101	Carlos Schnider	Copacabana 1210	094432528	high	.9701
102	Miriam Revoir		9001029	medium	.7945
103	A. Benedetti	Charrúa 1284/1	7091232	low	.9146
104	Luis López	Sixtina s/n		high	.8220

Table 2.10 – Partition of the Students relation ($\pi_{address, telephone}(\sigma_{stid < 100}(Students))$, $\pi_{stid, name, interview, test}(\sigma_{stid < 100}(Students))$, and $\sigma_{stid \geq 100}(Students)$)

Regarding storage, cell granularity needs more storage space. Accuracy values are stored for each cell, for example in an accuracy matrix [Motro+1998] where columns represent attributes, rows represent tuples and values in the table correspond to the accuracy of the tuple attribute (cell). In the proposal of [Motro+1998], authors store Boolean metrics of semantic correctness but the matrix can be used with the other types of metrics and factors.

Example 2.3. Continuing Example 2.1, Table 2.11 and Table 2.12 show two accuracy matrixes for the *Students* relation, both measuring semantic correctness, the former with a Boolean metric and the latter with a value-deviation metric. The value deviation metric is calculated as the distance between the actual value (v) and the database one (v'), normalized and subtracted from 1. For the test attribute (numeric value), the arithmetic difference is used as distance and the actual value is used for normalization:

$$\text{accuracy}(v',v) = \max\{(1 - |v - v'| / v), 0\}$$

For the name attribute (string value), the string edit distance is used, counting the number of characters that must be changed (added, deleted or modified) in order to transform the database value into the actual one; the size of the actual value is used for normalization:

$$\text{accuracy}(v',v) = \max\{(1 - \text{string\_distance}(v - v') / \text{size}(v)), 0\}$$

For the address attribute, a GIS application is used in order to determine if addresses match. A value of 1 corresponds to addresses that completely match (no matter typing errors), 0 corresponds to addresses that do not match and intermediate values correspond to addresses that partially match (e.g. same street but different door-numbers). Note that incomplete addresses can be useful (for example, letters can arrive to destination), however little errors in telephone numbers causes that the person cannot be contacted; so, for the telephone attribute, the distance is calculated by a Boolean function. For the interview attribute (enumeration with values low, medium and high), their similarity degrees are used as distance (given in Table 2.8). □

	stid	name	address	telephone	interview	test
21	0	0	0	0	0	0
22	0	0	0	0	0	0
43	1	1	0	0	1	1
56	1	1	1	0	1	1
57	1	1	1	1	1	1
58	1	0	0	0	0	0
101	1	1	1	1	1	1
102	1	1	0	1	1	1
103	1	1	1	1	1	0
104	1	1	1	1	1	1

Table 2.11 – Accuracy matrix of the Students relation (semantic correctness, Boolean metric)

	stid	name	address	telephone	interview	test
21	0	0	0	0	0	0
22	0	0	0	0	0	0
43	1	1	0	0	1	1
56	1	1	1	0	1	0.9974
57	1	1	0.5	1	1	0.9950
58	1	0	0	0	0.5	0.8444
101	1	1	1	1	1	1
102	1	1	0	1	1	1
103	1	0.8333	1	1	1	0.5627
104	1	1	1	1	1	0.9998

Table 2.12 – Accuracy matrix of the Students relation (semantic correctness, value-deviation metric)

An alternative storage way is extending the relation with additional attributes, each attribute for storing the accuracy of each data attribute. For example, Table 2.13 shows the *Students* relation with additional attributes (A_{sti}, A_{nam}, A_{add}, A_{tel}, A_{int} and A_{tes}) which store the accuracy values corresponding to the stid, name, address, telephone, interview and test attributes respectively.

For the set granularity only an accuracy value is stored for the whole relation or view. Following previous example, a semantic correctness ratio of 0.6 (ratio metric) or an average of 0.6247 (value-deviation metric) can be stored for the Students relation. In the particular case of partitions (areas are views), an accuracy value is stored for each area. For example, the semantic correctness ratios of 0.25, 0.50 and 0.9 are associated to the areas defined in Example 2.2 respectively. For the relationship granularity an accuracy value is stored for the relationship.

stid	A_{sti}	name	A_{nam}	address	A_{add}	telephone	A_{tel}	interview	A_{int}	test	A_{tes}
21	0	María Roca	0	Carrasco	0	6001104	0	low	0	1.0	0
22	0	Juan Pérez	0	Coloniaa 1280/403	0	9023365	0	medium	0	.5	0
43	1	Emilio Gutiérrez	1	Irigoitía 384	0	3364244	0	high	1	.8	1
56	1	Gabriel García	1	Propios 2145/101	1		0	low	1	.5	1
57	1	Laura Torres	1	Maldonado & Yaro	1	099628734	1	medium	1	.7	1
58	1	Raúl González	0	Rbla Rca Chile 1280/1102	0	4112533	0	high	0	.9	0
101	1	Carlos Schnider	1	Copacabana 1210	1	094432528	1	high	1	.9701	1
102	1	Miriam Revoir	1		0	9001029	1	medium	1	.7945	1
103	1	A. Benedetti	1	Charrúa 1284/1	1	7091232	1	low	1	.9146	0
104	1	Luis López	1	Sixtina s/n	1		1	high	1	.8220	1

Table 2.13 –Students relation enlarged with accuracy values (semantic correctness, Boolean metric)

Dimension 2: Typology of errors

As measuring accuracy corresponds to quantifying data errors or data imprecisions, measurement techniques are closely related to the types of errors or anomalies that arise to data. We use the term *error* in a wide sense, including not only incorrect and malformed values but also values that do not follow user expectations, for example, values that do not follow certain representation standard.

Several error classifications have been proposed in the literature, particularly in the domains of data cleaning and data mining [Rahm+2000] [Oliveira+2004] [Müller+2003] [Quass 1999] [Berti-Equille 2004]. We grouped such types of errors in 7 categories:

- *Value errors*: This category encloses syntactical errors in cells, such as out-of-range values, misspellings and other typing errors.
- *Standardization errors*: This category includes values that are syntactically correct but do not follow an expected standard, e.g. standard format or standard units.
- *Embedded values*: This category represents cell values that correspond to multiple values (e.g. an address attribute embedding street, door number, city and postal code).
- *Missing values*: This category corresponds to dummy or null values for mandatory attributes.
- *Integrity rule violations*: This category encloses violations to integrity rules among attributes and among tuples, such as functional dependencies or uniqueness constraints.
- *Duplicates*: This category corresponds to values that are duplicated in the database, either consistently or contradictorily.
- *Wrong values*: This category represents values that do not correspond to real-world entities.

Table 2.14 presents a summary of error types proposed in the literature, detailing a name for the problem*, a brief description and an example; the type column corresponds to our classification.

The classification of missing values is debatable; they can also be considered as value errors or integrity rule violations. We have chosen to classify them in a separate category because many works concentrate in null values, for example [Naumann+1999] [Redman 1996]. However, note that null values are generally related to the consistency and completeness quality factors but not to accuracy; to the former because having null values for mandatory attributes violates data integrity and to the latter because they cause incomplete descriptions of real world entities. Similarly, *integrity rule violations* and *duplicates* categories do not refer to accuracy but consistency problems, i.e. errors in the relationships among values and violations of dependencies among them. We described them here in order to be consistent with existent error classifications but we will omit them in the rest of the section.

Type	Problem	Descripcion	Example
Value errors	Illegal values	Values outside of domain range	date = 30/13/70
	Misspellings	Values incorrectly written; usually typos and phonetic errors	city=”Paaris”
	Misfielded values	Value entered in the wrong attribute	city=”France”
Standardization errors	Cryptic values	Cryptic values and abbreviations	occupation=”DB Prog.”
	Use of synonyms	Expression syntactically different but semanti-cally equivalent	occupation=”professor” occupation=”teacher”
	Word transpositions	Word transpositions, usually in a free-form field	name= “J. Smith” name=”Smith J.”
	No standard format	Values appear in different formats	date=”13/08/1974” date=”1974/08/13”
	No standard units	Values appear in different units	value=123 (euros / dollars)
Embed-ded values	Embedded values	Multiple values entered in one attribute (e.g. in a free-form field)	name=”J. Smith 12.02.70 New York”
Missing values	Missing values	Dummy or null values for mandatory attributes	phone=9999-999999
Integrity rule violations	Attribute dependency violations	Attribute dependency violation	age=22; date=12/02/70
	Uniqueness violations	Uniqueness violation	emp1=<”John Smith”,832> emp2=<”Peter Miller”,832>
	Referential integrity violations	Referential integrity violation	emp1=<”John Smith”, D127> department D127 not defined
Duplicates	Duplicated records	Same information represented twice (e.g. due to some data entry errors)	emp1=”John Smith” emp2=”J. Smith”
	Contradicting records	Same information is described by different (contradicting) values	emp1=<”John Smith”, 12/02/70> emp2=<”John Smith”, 22/02/70>
Wrong values	Incorrect values	Value does not correspond to real world situation	age=35 actual age is 37
	Wrong references	Referenced value is defined but wrong	emp=<”John Smith”, D127> actual department is D157

Table 2.14 – Typology of errors

Dimension 3: Data types

Measurement techniques are closely related to data types. Example 2.3 motivated this fact, where specific formulas were proposed in order to calculate value deviations for different data types. In the literature, most efforts are dedicated to evaluating distances among numeric values [Ballou+1998] and among string values

* When different names were proposed we take the most frequently used.

[Navarro 2001] [USNARA 2000], but some works propose specialized techniques for particular cases, such as addresses, telephones and emails [Amat+2005].

We observe data types from the user point of view, i.e. which is the type of the data that users expect. Note that data types may be different at sources, e.g. a date may be embedded in a string attribute. Furthermore, different sources may provide the same data but with different data types; for example, sex may be represented by integer numbers (0 and 1) and by strings ("male", "female", "M", "F", etc.). As a consequence, it may be possible that source data does not comply with the data types expected by users. In some cases appropriate transformation functions can be applied in order to standardize data (e.g. sex attribute of previous example), but in other cases, as in free-form strings, functions are very hard to abstract (e.g. free form address attributes). The most detailed the data types are specified, the easier data errors and anomalies may be identified and corrected.

We propose a classification of data types in some basic categories and discuss some special cases for each one:

- *Numeric types*: They constitute the easiest data type to check for imprecisions and deviations, allowing the definition of precise arithmetic functions to estimate data accuracy. As a special case, range specification allows detecting further anomalies.
- *Date types*: As dates are precisely typed, it is also easy to check for invalid values (e.g. 30/02/2006), format discrepancies (e.g. 02/20/2006 when expecting DD/MM/YYYY format) and imprecise values (e.g. February-2006 when also expecting the day). Deviations can be easily calculated with standard date-difference functions.
- *Enumeration types*: They enclose all data types representing a finite set of elements, e.g. month names, which can be mapped to naturals. Out of range checking is easy; the distance to the closer element can be used to correct errors (e.g. transform "Fevruary" into "February"). Lots of special cases can be defined, generally using term glossaries, domain ontologies or reference catalogs for comparing with valid elements. Most common examples are code lists (e.g. airport international codes) and name catalogues (e.g. street names). The increasing definition of domain ontologies allows checking for belonging of additional attributes, as technical terms (e.g. disease names) and the detection of synonyms and abbreviations. Mapping functions may also be defined in order to transform from an enumeration type (used at a source) to another one (expected by users), for example for translating technical terms from English to Spanish.
- *String types*: They constitute the most widely used data type and the most difficult to check for errors. In some special cases, attribute domains may be described by grammars, which allow the detection of value errors (e.g. personal names following the format "name initial dot surname" as "J. Smith"). Free form strings are very hard to treat and generalize; embedded values and lists of elements may appear. However, many special cases have been sufficiently treated for some specific domains, for example, customer addresses, emails and telephones in CRM* applications.

Further data types can be defined for specific applications, for example, streams, pictures, video, etc.

Dimension 4: Architectural Techniques

The accuracy of the data delivered to the user depends on *source data accuracy* (the accuracy of source data at extraction time) but also on the errors introduced or corrected by DIS processes. The *architectural techniques* dimension discussed for data freshness also have impact in data accuracy because some detection and correction techniques can be executed only in particular environments. We recall the three main families of architectural techniques discussed in Sub-section 2.3:

- *Virtual techniques*: The system does not materialize any data so all queries are calculated when they are posed. The system queries the relevant sources and merges their answers in a global answer that is delivered to the user. Examples are pure virtual mediation systems and query systems in database federations.
- *Caching techniques*: The system caches some information, typically data that is frequently accessed or the result of some frequent queries, and invalidates it when the time-to-live (TTL) has expired. If the information required to answer a user query is stored in the cache, the system delivers it to the user; if not, the system queries the sources as in virtual systems. Examples are caching systems.

* CRM = Customer Relationship Management

- *Materialization techniques*: The system materializes large volumes of data which is refreshed periodically. The users pose their queries and the system answers them using the materialized data. Examples are data warehousing systems and web portals that support materialization.

When materializing data, complex transformations can be applied to data, including routines for error correction or format standardization. Such processes can also introduce errors, for example, rounding numbers can lose precision or normalizing addresses can cause values to be not longer semantically correct. Virtual techniques must provide query answers within short delays, so such complex transformation cannot be applied, however, simple format transformations and arithmetic operations are sometimes included in such systems. So even in less degree, some errors can be introduced by virtual techniques. Caching techniques only copy data from sources, so no errors are introduced no corrected during the process.

In addition, some measurement techniques, especially those comparing to real-world, which needs greats amounts of time for executing, only can be implemented for materialized data.

In next sub-section we present a taxonomy that relates these four dimensions summarizing their analysis.

3.4. A taxonomy of accuracy measurement techniques

In this sub-section we present a taxonomy that summarizes the discussion and allows comparing different proposals for accuracy evaluation. The taxonomy is composed of the previously described dimensions: (i) granularity of measurement, (ii) typology of errors, (iii) data types, and (iv) architectural techniques.

Firstly, there is a close relation between the *typology of errors* and the accuracy factors, since factors allow quantifying data errors, i.e. inaccuracies. Some relations are quite intuitive; semantic correctness is related to wrong values (values that do not match real world situations) and syntactic correctness is related to value errors, standardization errors and embedded values (all of them representing syntactic errors). However, the relation of the precision factor with some types of errors is less evident to see. Commonly, lack of precision appears as a standardization error (e.g. 143 when expecting a value with two decimal digits). In addition, some value errors, specifically out of domain values (e.g. 1974 when expecting a date value) correspond to imprecision. However, although embedded values may contain incomplete information (e.g. "John" when expecting the whole name) they are very hard to quantify and are not considered as lacks of precision. Table 2.15 shows the relation among accuracy factors and error types, indicating the valid (✓) and invalid (×) combinations.

Accuracy factor / *Error type*	**Semantic correctness**	**Syntactic correctness**	**Precision**
Value errors	×	✓	✓
Standardization errors	×	✓	✓
Embedded values	×	✓	×
Wrong values	✓	×	×

Table 2.15 – Relation among accuracy factors and typology of errors

There is also a relation among the *data types* and the *typology of errors*, specifying the kind of errors that can appear in each data type. For example, for some special cases of string data (as addresses) embedded values are frequent. Specific techniques for error detection and correction have been developed for specific data types and specific error types. But note that the relationship between these dimensions is merely syntactical; the correspondence with real world is independent of data types, and consequently, wrong values (values that do not match real world situations) are not related to specific data types. Table 2.16 shows the interrelations among these dimensions categories, indicating the valid combinations; value and standardization errors can appear for all data types, however, embedded values are most frequent for the string type.

Architectural techniques are related to error types. Specifically, the evaluation of wrong values generally implies the comparison with real world, which is generally very costly (in time); therefore, assessment techniques only can be executed when data is materialized. The checking of embedded values also requires expensive routines. Virtual and caching techniques only can execute simple assessment functions, looking for value and standardization errors. Table 2.17 shows the interrelations among these dimensions categories, indicating the valid combinations; value and standardization errors can appear for all types of techniques, however, embedded and wrong values are treated only by materialized techniques.

Data type / *Error type*	**Numeric**	**Date**	**Enumeration**	**String**
Value errors	✓	✓	✓	✓
Standardization errors	✓	✓	✓	✓
Embedded values	×	×	×	✓
Wrong values	no matter data type			

Table 2.16 – Relation among typology of errors and data types

Architectural techniques / *Error type*	**Virtual**	**Caching**	**Materialization**
Value errors	✓	✓	✓
Standardization errors	✓	✓	✓
Embedded values	×	×	✓
Wrong values	×	×	✓

Table 2.17 – Relation among typology of errors and architectural techniques

A priori, all combinations of *granularities of measurement* and *error types* are possible, i.e. cell, set or relationship granularity can be used in the evaluation of the different types of errors (with their valid combinations of data types and architectural techniques).

In addition, *granularity of measurement* is also orthogonal to the accuracy factors, i.e. users may be interested in any accuracy definition, independently of the way accuracy measures are stored. But taking into account the particularities of the systems and the availability of metadata and estimations, the different metrics are generally more related to some levels of granularity. Concretely, semantic correctness ratio and syntactic correctness ratio metrics are not appropriate for cell granularity and scale metric (precision) have no sense for string data types. In addition, as argued in Sub-section 3.2 the standard error metric (precision) is rarely used in DIS applications. Table 2.18 shows the correlation among all taxonomy dimensions and accuracy factors (some metrics are marked in brackets when not all the metrics are appropriate).

Granularity / *Error type – Data type – Arch. techniques*			**Cell granularity**	**Set granularity**	**Relationship granularity**
Value errors	**Numeric Date Enumeration String**	**Virtual Caching Materialization**	Syntactic correctness (deviation) Precision	Syntactic correctness Precision	Syntactic correctness Precision
Standard. errors	**Numeric Date Enumeration String**	**Virtual Caching Materialization**	Syntactic correctness (deviation) Precision (scale, granularity)	Syntactic correctness Precision (scale, granularity)	Syntactic correctness Precision (scale, granularity)
Embedded values	**String**	**Materialization**	Syntactic correctness (deviation) Precision (granularity)	Syntactic correctness Precision (granularity)	Syntactic correctness Precision (granularity)
Wrong values	**No matter data type**	**Materialization**	Semantic correctness (degree, deviation)	Semantic correctness	Semantic correctness

Table 2.18 – Correlation among taxonomy dimensions

The technical problems to solve for each cell of the taxonomy are quite different. For example, measuring semantic correctness at cell granularity requires costly evaluation processes, sometimes requiring human interaction, while counting the ratio of cells do not satisfying a syntactic rule can be easily implemented and executed. In Sub-section 3.6 we discuss evaluation problems.

3.5. Some systems that consider data accuracy

In this sub-section we analyze several types of systems that evaluate accuracy and we describe the goals and problems that they present. We do not try to be exhaustive because there exist lots of propositions that take into account data accuracy; we intend to present an overview of the problems. At the end of the sub-section, Table 2.19 summarizes the discussed proposals in terms of the taxonomy presented before.

Data Warehousing systems

In data warehousing systems, accuracy is studied in the context of data cleaning techniques. The main objective is to detect and correct errors. There is a great variety of techniques for detecting specific errors, generally specialized for some application domains. The most used ones consists in format parsing [Raman+2001], outliers detection [Maletic+2000] and frequency distribution analysis [Quass+1999]. In [Oliveira+2005], authors present a taxonomy of data quality errors (that refines the categories presented in Sub-section 3.3) and some methods (based on decision trees) for deciding which types of errors may arise to data. Regarding error correction, most automatic correction techniques consist in applying user defined functions [Lee+2000] [Galhardas+2000] [Sattler+2000] or simply deleting erroneous values [Vassiliadis+2001] [Sattler+2000]. When errors cannot be automatically corrected, they are reported in a log [Vassiliadis+2001] or interactively presented to the user [Raman+2001] [Galhardas+2000] in order to be manually corrected.

Several cleaning tools have been proposed in the literature, e.g. AJAX [Galhardas+2000], FraQL [Sattler+2000], Potter's Wheel [Raman+2001], ARKTOS [Vassiliadis+2001] and IntelliClean [Lee+2000]. An overview and comparison of tools and the underlying techniques can be found in [Müller+2003] [Oliveira+2004]. The tools bring support for detecting and correcting a wide range of errors, specifically value errors (basically detecting illegal formats and replacing with some derived value), standardization errors (mapping abbreviations and formats to the standard ones), embedded values (looking for frequent patterns and breaking into separate attributes) and missing values (filling-in with derived values). Tools also provide support for detecting and correcting consistency problems, however, they do not treat wrong values (corresponding to semantic correctness problems).

But cleaning techniques do not deal with measuring accuracy levels and informing them to users. Only few works focus on reporting data quality to users. In [Moura+2004], authors build multidimensional cubes to show aggregations of data warehouse quality, some of the dimensions corresponding to indicators related with data accuracy.

In [Zhu+2002] the quality of external sources is evaluated in order to select the most appropriate sources. They study and compare different methods for building quality indicators that aggregate several quality measures; the *syntactic correctness ratio* is one of the proposed quality metrics.

Mediation systems

In mediation systems, *syntactic correctness* is used as a quality metric to compare among sources and to filter the data returned to the user. In [Naumann+1999], they introduce quality factors in a mediation system, which include *syntactic correctness*. They study how to propagate a set of quality factors from several heterogeneous sources to the mediator. The propagation consists basically of merge functions that combine actual values of two sources through a relational operator. They use the *syntactic correctness ratio* metric with a relation/view granularity; considered errors are value errors (misspellings, out-of-range values, etc.).

Cooperative systems

A Cooperative Information System (CIS) is a large scale information system that interconnects various systems of different and autonomous organizations, geographically distributed and sharing common objectives [Mecella+2002]. A service-based framework and an overall architecture for managing data quality in CIS is proposed in [Mecella+2002]. Their architecture allows each organization to export data with associated quality information and to retrieve data specifying quality requirements. Their quality model includes, among other quality factors, the semantic correctness factor, measured as a value-deviation. Quality values are associated to each cell (data is represented in XML documents; quality values follows the same structure). In [Fugini+2002], authors evaluate semantic correctness and syntactic correctness (value errors), both measured as value-deviations

and associated to cells. They use quality measures to support the exchange of trusted data, enabling organizations to assess the suitability of data before using it.

A model for quality management in CIS is presented in [Missier+2001]. They propose an extensive use and exchange of quality metadata. They measure different accuracy factors: semantic correctness, syntactic correctness (value and standardization errors) and precision (scale and granularity). They store the validation history of each cell (validating an item value means testing it, using either some external reference data or some checking algorithm) and discuss its use for data quality enforcement. They use the model in a case of study in the domain of public administration [Missier+2003]. Source data presents a number of problems, e.g. addresses become stale (and then no correspond with real-world), different address formats, same names with multiple spellings and typing errors (generally misspellings). They study assessment methods for their data, classify errors and recommend cleaning techniques for correcting major errors (other errors are lead to manual correction, but their number is substantially reduced).

Other applications

Other types of applications, specifically those managing customers' data, need precise information of the accuracy of each data item. For example, CRM applications manage customer relationship information, generally accessing to multiples sources in order to collect all relevant information and to abstract a customer profile; external sources are increasingly being incorporated. In these applications, business and advertisement information is frequently sent to customers, so having accurate contact information is a major goal. Other information, useful for classifying customers (e.g. profession, income level and preferences), should be accurate enough in order to define good targets for promotions and advertising.

Information quality and particularly information accuracy is indispensable in such applications [Laboisse 2005] [Amat+2005] [Graveleau 2005]. The major goal is achieving semantic correctness. They use a great variety of techniques for detecting and correcting errors, sometimes checking for syntactic correctness in order to find

Works	Measurement	Granularity measurem.	Typology of errors	Data type	Arch. techniques
Data cleaning [Galhardas+2000] [Sattler+2000][Lee+2000] [Raman+2001] [Vassiliadis+2001]	Syntactic correctness	Cell	Value errors, standard. errors, embed. values	All	Materialization
Reporting warehouse quality [Moura+2004]	Syntactic correctness, precision	Set	Value errors, standard. errors, embed. values	All	Materialization
Source selection [Zhu+2002]	Syntactic correctness (ratio)	Set	Value errors	Not specified	Materialization
Quality-based integration [Naumann+1999]	Syntactic correctness (ratio)	Set, relationship	Value errors	Not specified	Virtual
Retrieving quality data in cooperative systems [Mecella+2002]	Semantic correctness (deviation)	Cell	Wrong values	Not specified	Materialization
Trusted data exchange in cooperative systems [Fugini+2002]	Semantic correctness (deviation), syntactic correctness(deviation)	Cell	Value errors, wrong values	Not specified	Materialization
Exchange of quality metadata [Missier+2001] [Missier+2003]	Semantic correctness, syntactic correctness, precision (scale, granularity)	Cell	Value errors, standard. errors, wrong values	All	Materialization
Data cleaning in CRM [Laboisse 2005] [Amat+2005]	Semantic correctness, syntactic correctness, precision (scale)	Cell	All	All	Materialization

Table 2.19 – Summary of proposals

possible semantic errors (e.g. an email that is syntactically incorrect will fail to be delivered). Reference dictionaries (e.g. address dictionaries) are used to verify the existence of data values. But most of the effort corresponds to expensive manual verification tasks, as telephone calls and emails for verifying the exactitude of customer data. In order to involucrate customers in the process (motivate them to answer questions) special promotions and prix are created, further increasing the quality enforcement process. Furthermore, verification and cleaning techniques are often delegated to third party organizations specialized in quality control [Laboisse 2005] [Amat+2005].

3.6. Research problems

Data accuracy has been largely studied for various types of DIS, however, many problems remain unsolved or insufficiently treated. This sub-section summarizes these problems and mentions, when they exist, the references which have done significant contributions in each class of problem. Most of the problems coincide with those ones studied in Sub-section 2.6 for data freshness, so we give only a brief description of the problems and we remit to Sub-section 2.6 for details.

The accuracy evaluation process needs to know about the users' and source profiles, i.e. metadata about users' expectations and source properties. Such elements should be adequately combined in order to evaluate the accuracy of data conveyed to users. Accuracy evaluation can be used for auditing an existing DIS or for designing a new DIS under quality constraints. In the following, we discuss these problems.

Defining users' and source profiles

Evaluating data accuracy implies testing whether user's accuracy expectations can be satisfied from source data accuracy. One of the first problems is how and where to specify user expectations and how to define data source properties which impact data accuracy.

Specification of accuracy expectations

Users have accuracy expectations for their applications which should be formulated according to some factors and metrics among those we have seen in Sub-sections 3.1 and 3.2. The specification of these factors and metrics pose a number of questions:

- Which language or formalism to use? Alternatives can vary from the simple specification of ratios [Kon+1995] [Naumann+1999] [Li+2003a] [Redman 1996] associated to each object type, to the definition of a specialized language if a preference order is introduced among accuracy of different object types.
- At what level accuracy expectations should be specified? Analogously to data freshness, we distinguish four levels: (i) for the whole system, (ii) for each data source, (iii) for each user or group of users, and (iv) for each user query. Technical problems are similar.

Acquisition of source accuracy

The evaluation of data accuracy at source level implies the selection of an accuracy factor and the definition of metrics and measurement processes for it. The definition of new factors and metrics is an interesting area. Most existing works concentrate in the syntactic correctness factor, while industry is increasingly demanding better techniques for evaluating semantic correctness and precision. Furthermore, the current development of third party service societies specialized in the hosting and measurement of accuracy of enterprise data confirms its importance [Laboisse 2005]. The acquisition of source data accuracy poses some questions:

- Which source metadata is necessary to represent accuracy in a source profile? In order to characterize source actual accuracy some metadata should be obtained, for example error distribution. This metadata will depend on the level of granularity (cell, set or relationship) choose for expressing accuracy metrics.
- How to acquire such metadata from sources? Some sources (or domain experts) can provide useful information as the confidence degree for certain attributes, but for other sources these values must be

learned or estimated from statistics elaborated during the data source exploitation. There is a large collection of techniques and functions for measuring some types of errors, commonly known as error detection techniques. The most used ones consists in format parsing [Raman+2001], outliers detection [Maletic+2000] and frequency distribution analysis [Quass+1999]. Several cleaning tools have been also proposed, e.g. AJAX [Galhardas+2000], FraQL [Sattler+2000], Potter's Wheel [Raman+2001], ARKTOS [Vassiliadis+2001] and IntelliClean [Lee+2000]. Both techniques and tools bring support for detecting a wide range of syntactical errors (value errors, standardization errors and embedded values) but do not treat wrong values (corresponding to semantic correctness problems). Techniques for specific metadata should be also developed as in [Laboisse+2005].

Auditing data accuracy

Having users and source profiles, one of the challenging problems is how to combine these elements in order to evaluate the accuracy of the data conveyed to users. The main questions are: How should the different parameters of the source profile be combined for evaluating data accuracy? What is the impact of error distribution in the accuracy of data?

An additional question is how to combine several source actual values to obtain a global quality estimation. There is a proposal for combining accuracy ratios within SQL operators, following gross hypothesis of uniform distribution of errors [Naumann+1999]. The combination function is very simple (product of the input values). Error models that better represent source data should be analyzed. An important contribution in this line was presented in [Motro+1998], partitioning source data according to error distribution. The impact of complex operations such as data cleaning processes are not treated at all.

Accuracy evaluation techniques can be used in the development of auditing tools, responsible for the evaluation of data quality. Several kinds of auditing tools can be conceived, for example:

- Prediction tools, for predicting the quality of data that can be returned in response to a query, without executing the query.
- Integrated evaluation tools, for measuring data quality during query execution and labeling delivered data with its quality levels. These tools can be integrated to the query evaluation process.
- Statistical tools, for taking samples of data quality during query execution and storing statistics. These tools can serve the first two categories.

The tools should use metadata describing the sources, the DIS and user expectations. They can be used at design time to evaluate the system, for example to test if user expectations can be achieved, or they can be used at run time for example, to predict the quality of the data delivered by alternative processes in order to choose the process that best suites user quality expectations.

Quality-driven engineering

Quality-driven engineering consists in designing a DIS under quality constraints. This kind of techniques may help in the selection among several design choices:

- Selection among alternative query plans that access alternative data sources.
- Specific error correction techniques as duplicate elimination.
- Relaxation of selection or join conditions, replacing crisp predicate for approximate ones.

Although several kinds of techniques have been explored in depth for specific systems, adapting their capabilities to heterogeneous systems requires addressing additional problems, as the combination of source data with different quality values, or even the comparison of data source profiles.

The challenge in quality-driven engineering is in the identification of the variables that influence data accuracy and the proposition of techniques to achieve such improvements. Most existing work consists in automating error detection and correction, either applying user defined functions [Lee+2000] [Galhardas+2000] [Sattler+2000] or simply deleting erroneous values [Vassiliadis+2001] [Sattler+2000]. When errors cannot be automatically corrected, they are reported in a log [Vassiliadis+2001] or interactively presented to the user [Raman+2001] [Galhardas+2000] in order to be manually corrected. Existing cleaning tools bring support for detecting and correcting a wide range of errors, specifically value errors (basically detecting illegal formats and

replacing with some derived value), standardization errors (mapping abbreviations and formats to the standard ones) and embedded values (looking for frequent patterns and breaking into separate attributes). However, they do not treat wrong values (corresponding to semantic correctness problems).

Certain works propose the selection of pertinent sources according to their quality [Zhu+2002] [Mihaila+2000] [Naumann+1998]. In particular, data accuracy is taken into account in [Zhu+2002]. They compare different methods for building quality indicators that aggregate several quality measures, in order to rank sources according to such indicators. None of the proposals takes into account user expectations.

As improving accuracy is not the unique quality goal for a given system, the relationship with other quality properties becomes a major challenge in DIS design. The relationship among accuracy and other quality properties has been only partially studied. There are two main lines: (i) tuning other properties in order to optimize accuracy, and (ii) relaxing accuracy in order to optimize other quality factors. In the former, the challenge is in the identification of related quality properties and the proposition of techniques that take advantages from these relationships in order to improve the global quality of data. Existing works in this line mainly concern the balance of semantic correctness and timeliness [Ballou+1995], syntactic accuracy and completeness [Ballou+2003], semantic correctness, completeness and timeliness [Cappiello+2002] and semantic correctness, process timeliness and cost [Han+2003][*]. The latter is not treated in the literature.

Discussion

Among the discussed research problems, this thesis deals with data accuracy auditing and accuracy-driven engineering. Regarding data accuracy auditing, we are interested in the combination of relevant parameters (source data accuracy, error distribution, user expectations) in order to evaluate the accuracy of data conveyed to users. We fix our context to the relational model and we take into account how inaccuracies are distributed in source relations, inspired in the partitioning mechanism proposed in [Motro+1998]. Data accuracy values are used in an accuracy-driven application which goal is to answer user queries using data that verifies accuracy expectations. Contrarily to source selection proposals [Zhu+2002] [Mihaila+2000] [Naumann+1998] which select whole sources relations (or whole sources), we consider that source relations are partitioned according to data accuracy and we answer user queries using the areas of source relations that better adapt to user needs. Chapter 4 deals with data accuracy evaluation and enforcement.

4. Conclusion

In this chapter we described two of the most used quality dimensions: data freshness and data accuracy. We analyzed several factors and metrics proposed in the literature to measure them and we explored the dimensions that influence their evaluation, which were organized in taxonomies. Guided by the taxonomies, we classified some works that consider data freshness and data accuracy and we analyzed open research problems. Both analyzes, for data freshness and for data accuracy, shown that existing work concentrates for some specific types of DIS, specific characteristics (e.g. materialized data, some types of errors) or specific metrics but other configurations remain untreated.

As research problems we identified the specification of user expectations, acquisition of source data quality, formulation of cost models, data quality auditing and quality-driven engineering. Among these problems, we are interested in the two latter. Specifically, this thesis deals with the evaluation of freshness and accuracy of the data conveyed to users in DIS applications. The obtained quality values are analyzed and compared to user expectations (i.e. quality auditing) and are input for specific techniques for quality improvement (i.e. quality-driven engineering). The following chapters develop the proposal.

[*] Semantic correctness is called accuracy in [Ballou+1995], [Cappiello+2002] and [Han+2003]; syntactic accuracy (specifically the conformance to a standard) is called consistency in [Ballou+2003] and timeliness is called currency in [Cappiello+2002].