A Framework for Analysis of Data Freshness

Mokrane Bouzeghoub
Laboratoire PRISM, Université de Versailles
45, avenue des Etats-Unis
78000. Versailles, FRANCE
+33-1-39254051
Mokrane.Bouzeghoub@prism.uvsq.fr

Verónika Peralta
Laboratoire PRISM, Université de Versailles
45, avenue des Etats-Unis
78000. Versailles, FRANCE
+33-1-39254083
Veronika.Peralta@prism.uvsq.fr

ABSTRACT

Data freshness has been identified as one of the most important data quality attributes in information systems. This importance increases particularly in the context of distributed systems, composed of a large set of autonomous data sources, where integrating data having different freshness may lead to semantic problems. There are various definitions of data freshness in the literature, depending on the applications where they are used, as well as different metrics to measure them. This paper presents an analysis of these definitions and metrics and proposes a taxonomy based upon the nature of the data, the type of application and the synchronization policies underlying the multi-source information system. We analyze, in terms of the taxonomy, the way freshness is defined and used in several types of systems and we present some open research problems in the field of data freshness evaluation.

Keywords

Freshness, Quality evaluation, Multi-source information systems.

1. INTRODUCTION

Data freshness has been identified as one of the most important attributes of data quality for data consumers [31]. Some surveys and empirical studies have proved that data freshness is linked to information system success [34][25][31]. 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 between alternative data providers and of combining data having different freshness values. This paper presents an analysis of data freshness and its various underlying metrics within the context of a data integration system.

A Data Integration System (DIS) is an information system that integrates data of different independent data sources and provides the users with a uniform access to the data by the mean of a global model, as sketched in Figure 1. User queries are posed to the system over the global model.

Examples of DIS are Mediation systems, where data is extracted from several sources, integrated and presented to the user; wrapper-mediator architecture [36] is commonly used to perform these tasks. Data Warehouse (DW) systems [19] also extract, transform and integrate information from various, possibly heterogeneous, sources and make it available for strategic analysis to the decision makers. Other examples of DIS are federations of databases [30] where a key characteristic is the preservation of data source autonomy and Web Portals which provide access to subject-oriented information acquired and synthesized from Web sources, generally caching important amounts of data [5].

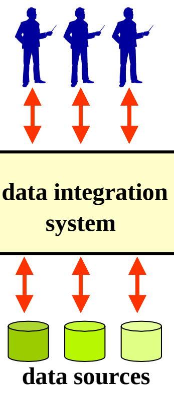

Figure 1 - Data Integration Systems

Several freshness definitions have been proposed for different types of DIS. The traditional freshness definition is related to view consistency when materializing source data at the integration level or the user level. It is called *currency* [29] and describes how *stale* is data with respect to the sources. Recent proposals incorporate another notion of freshness, called *timeliness* [34], which describes how *old* is data. Then, freshness 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.

In this paper we analyze this quality dimension and we present a taxonomy of its factors and metrics based upon the nature of the data, upon the type of application and upon the synchronization policies of the underlying management system. We analyze, in terms of the taxonomy, the way freshness is defined and used in several types of systems and we discuss some open research related to data freshness evaluation in a DIS.

The contribution of this paper is twofold. Firstly, we propose a framework for the analysis of data freshness factors and metrics within a taxonomy. Secondly, we discuss some research problems related to data freshness evaluation.

The rest of the document is organized as follows: Section 2 discusses the different definitions and metrics of the freshness dimension. Section 3 presents the taxonomy which is used in section 4 to describe some systems that evaluate data freshness. Section 5 presents some open research problems related to data freshness evaluation. Finally, section 6 concludes with our general remarks.

2. DATA FRESHNESS

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?

Data freshness has then not a unique definition in the literature. There are various definitions concerning different concepts and metrics, which are mainly due to the different objectives of the systems where they are used. For example, in a replication system the number of refresh operations that have to be applied to a slave relation in order to reflect the changes in a master relation is a good freshness metric, but in a data warehousing system the time passed from the update of an object can be more relevant. In this section we present an analysis of data freshness definitions and metrics.

2.1 Definitions

Data freshness comprises 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* [20].

From a user point of view, we distinguish two sub-dimensions of this quality dimension:

- ❑ *Currency factor* [29]: It captures the gap between the extraction of data from the sources and its delivery to the users. For example, currency indicates how stale is the account balance presented to the user with respect to the real balance at the bank.
- ❑ *Timeliness factor* [34]: It captures how often data changes or how often new data is created in a source. For example, timeliness indicates how often the product prices change in a store or how often new books are added to a library.

Analogously, other factors can be defined. For example, consider a source that is frequently updated but continues having old data that is never updated (sometimes not representing real-world phenomena). The oldest data may introduce some noise on query evaluation. A freshness factor that captures how old is the oldest data in a source can be useful for this example.

In the next section we analyze freshness metrics.

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 sub-dimension:

- ❑ 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) [29]. In practice, as the precise change time can 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 [33]. In caching systems it has been defined as *recency* or *age*, representing respectively the time elapsed since an object was cached [5] and the time elapsed since an object became stale [7]. 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 [11].
  - ▪ *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 [15]. 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 [17]. In query processing systems, it is defined as the number of insertions, deletions and modifications since the data materialization time [9]. 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 [11].
  - ▪ *Freshness-rate 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 [7]. 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 [7]. In [22], freshness is measured as the percentage of web pages that are fresh (not stale) in the cache but considering that the pages can have portions that are fresh and proportions that are not.
- ❑ Metrics for the timeliness factor:
  - ▪ *Timeliness metric*: It measures the extent to which the age of the data is appropriate for the task on hand [34]. It is generally estimated as time elapsed from the last update to a source and bounded using the update frequency of the source data [26]. It is also related to data volatility that identifies the time interval in which data remain valid [2]. It was defined as *timeliness* in mediation systems [26] and web systems [12].

Table 1 summarizes freshness factors and corresponding metrics.

3. DIMENSIONS FOR FRESHNESS ANALYSIS

In this section we analyze some dimensions that impact the analysis and enforcement of data freshness. We analyze the nature of data, the type of applications and the synchronization policies of the system. Finally, we present a taxonomy that summarizes this discussion and allows comparing different proposals for freshness evaluation.

Table 1 – Summary of freshness factors and metrics

Factor	Metric	Definition
Currency	Currency	The time elapsed since data was extracted from the source (The difference between query time and extraction time).
	Obsolescence	The number of updates transactions/ operations to a source since the data extraction time.
	Freshness rate	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 query time and last update time).

3.1 Nature of Data

According to its change frequency, we can classify source data in 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. 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.
- *Frequently-changing data*: It is data that has intensive change, such as real-time traffic information, temperature sensor measures and sales quantities. The changes can occur with a defined frequency or they can be random. 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.

When working with frequently changing data, it is interesting to measure how long data can remain unchanged and minimize the delivery of expired data. 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.

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.

Analogously, when data has a defined change frequency, applications can be synchronized to extract data at the best moment.

3.2 Application Types

The freshness of the data returned to the user depends on the freshness of extracted data but also on the processes that extract, integrate and deliver this data. These processes are very important because they can introduce additional delays. We distinguish three main families of applications: those that calculate data when a new query is posted, those that cache the data most frequently used, and those that materialize the data needed to answer user queries. When using materialization, data is stored for some time in the data integration system repositories, which decreases its freshness. The features of these three categories of DIS are summarized below:

- *Virtual systems*: The system does not materialize any data so all queries are calculated when they are posed. The users send their queries to the system and wait the response. The system re-writes user queries in terms of the sources, sends the correspondent queries to 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 systems*: The system caches some information, typically some source relations that are frequently accessed or the result of some frequent queries. The system estimates the time during which the data will be up to date, and invalidates it when passed this time. The users pose their queries to the system and if the information required to answer the queries is stored in the cache, the system delivers it to the user. If the information is not stored in the cache or it is invalidated, the system queries the sources as in the virtual systems and possibly refreshes cache data. Examples are caching and replication systems.
- *Materialized systems*: The system materializes large volumes of data which is refreshed with respect to a certain scenario. The users pose their queries and the system answers them almost using the materialized data. Examples are data warehousing systems and web portals that support materialization.

Virtual systems are conceived to retrieve data as current as possible, returning a current state of the source data. In caching systems, some level of staleness is allowed but the gap between source states and the integration system state should be relatively small. In materialized systems that gap can be greater, but its magnitude depends on the concrete applications.

3.3 Synchronization Policies

The way DIS are implemented influences the freshness of the data delivered to the users. Particularly, the synchronization between 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 between 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. The notification of new available data can come from an active source, 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. Push policies can also be driven by temporal or non-temporal events.

According to the interaction between 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. Push policies can also be driven by temporal or non-temporal events.

Combining the previous interactions between users, DIS and data sources leads to six possible configurations which are shown in Figure 2. We name each configuration with the user-DIS policy followed by the DIS-source policy. Asynchronism is represented by a slash (/), synchronism by (-):

- Pull-pull: The interaction is fully synchronized. When a user poses a query (pull), it is decomposed and sent by the DIS to the sources (pull). The configuration is represented by arrow (a) of Figure 2. It is common in virtual mediation systems.
- Pull / pull: When a user poses a query (pull) the DIS answers it using materialized data. Asynchronously, the DIS queries the sources to refresh materialized data (pull). It is represented by arrows (b) and (c) of Figure 2. It is common in data warehousing systems.
- Pull / push: When a user poses a query (pull) the DIS answers it using materialized data. Asynchronously, the sources send data to refresh materializations (push). It is represented by arrows (b) and (e) of Figure 2. It is also used in data warehousing systems.
- Push / push: 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 represented by arrows (d) and (e) of Figure 2. It is used in publish/subscribe environments.
- Push / pull: Materialized data is conveyed asynchronously to the users (push) and also asynchronously, the DIS queries the sources to refresh the materialized data (pull). It is represented by arrows (d) and (c) of Figure 2. It represents certain user applications (e.g. data marts) that are regularly fed from warehouse data.
- Push-push: The interaction is synchronized. When sources send data to the DIS (push), the DIS conveys it to the users (push). It is represented by arrow (f) of Figure 2. 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 policy is not usually implemented in the three application types described in the previous subsection.

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.

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.

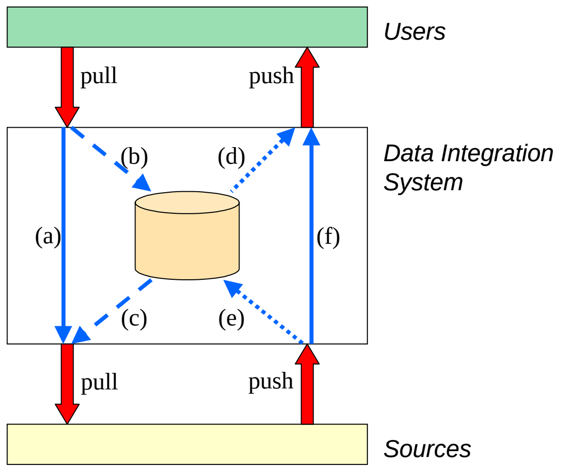

Figure 2 – Combination of synchronization policies

3.4 A Taxonomy of Freshness Works

The proposed taxonomy is composed of the previously described dimensions: (i) nature of data, (ii) application types 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 shows the relation, indicating when data freshness can be assured without evaluation and when it is interesting to evaluate it (✓) or not (✗).

Table 2 – Interesting combinations of freshness factors and nature of data

	Frequently changing	Long-term changing	Stable
Timeliness	✗	✓	✓
Currency	✓	assured	assured

Application type and *synchronization policy* are system-oriented dimensions which describe the system relation with data freshness. Not all the combinations between application types and synchronization policies are valid. Virtual systems only support the pull-pull configuration. Materialized systems support the configurations having an internal repository to store materialized data. Caching systems support the configurations that pull source data (synchronous and asynchronous). Table 3 shows the interrelations between them, indicating the valid combinations.

Table 3 – Valid combinations of application types and synchronization policies

	pull-pull	pull/pull	pull/push	push/pull	push/push
Materialized	✗	✓	✓	✓	✓
Caching	✓	✓	✗	✓	✗
Virtual	✓	✗	✗	✗	✗

The user-oriented and the system-oriented dimensions are orthogonal. Virtual, caching or materialized systems (with their valid combinations of synchronization policies) can be built to query different types of data.

The system-oriented dimensions are also orthogonal to the freshness factors, because user interest in freshness is independent to the way the system is implemented. However, the metrics for the currency factor are related to the system implementation. 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 [5][17][7]. In materialized systems, currency and obsolescence have been used [33][9]. Table 4 shows the co-relation of all the taxonomy dimensions:

Table 4 – Co-relation of all the taxonomy dimensions

	Frequently changing	Long-term changing	Stable
Virtual **Pull-pull**	Currency	Timeliness	Timeliness
Caching **Pull-pull** **Pull/pull** **Pull/push**	Currency Obsolescence Freshness-rate	Timeliness	Timeliness
Materialized **Pull/pull** **Pull/push** **Push/pull** **Push/push**	Currency Obsolescence	Timeliness	Timeliness

The technical problems to solve for each combination 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 section 5 we discuss the freshness evaluation problems.

4. SOME SYSTEMS THAT CONSIDER FRESHNESS

In this section we analyze several types of systems that evaluate freshness and we describe the goals and problems that they present. Table 5 summarizes the proposals in terms of the taxonomy presented before.

4.1 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 [9], 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 [15]. The notion of consistency means that the DW state reflects an actual source state at some “recent time” [38]. 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 [13]. 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 [16][14][32][37][24][3].

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 [35]. 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 [18][32][3] (pull/push and pull/pull policies). In [33], *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).

4.2 Mediation Systems

In classical mediation systems, freshness is also studied through the *currency factor*. In [18], 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 [26], they introduce quality factors in a mediation system, which include *timelines*. 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.

Table 5 – Summary of proposals

Works	Measurement	Nature of data	Application type	Synchronization policy
Materialization for query processing [9]	Obsolescence	Frequently changing	Virtual, materialized	Pull-pull, pull/pull
View maintenance [13][18][15][38]	Currency	Frequently changing	Materialized	Pull/pull, pull/push
View maintenance policy [33]	Currency	Not specified	Materialized	Pull/pull
Selection of views to materialize [16][14][32][37][24][3]	Currency	Frequently changing	Materialized	Pull/pull, pull/push
Mediation design combining virtual and materialized approaches [18]	Currency	Not specified	Virtual, materialized	Pull-pull, pull/pull
Source selection in virtual mediation [26]	Timeliness	Not specified	Virtual	Pull-pull
Cache refreshment [5][17][7][23]	Currency, obsolescence, freshness-rate	Frequently changing / Long-term changing	Caching	Pull-pull, pull/pull, Pull/push
Cache refreshment [22]	Freshness-rate	Frequently changing	Caching, materialized	Pull-pull, pull/push
Replica refreshment [11]	Currency, obsolescence	Frequently changing	Caching	Pull-pull, pull/pull

4.3 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 metrics (currency, obsolescence, freshness-rate).

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 [7], 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-rate. In [23], they focus in the fine tuning of the caching policy of a dynamic content page cache, balancing response time and invalidation cycles for assuring data currency. In [5], 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 [22], 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.

4.4 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 [11]. 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 [11]. Their goal is determining 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.

5. RESEARCH PROBLEMS

With the development of DIS which provide alternative data sources and alternative answers to user queries, data freshness is becoming a *first class quality dimension* whose factors and metrics are more and more required by end-users.

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 section summarizes these problems and mentions, when they exist, the references which have done significant contributions in each class of problems.

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.

The freshness evaluation process can be used at two different times: (i) at exploitation time to predict or report the data freshness provided by an existing system or (ii) at design time to engineer a new system under data freshness constraints.

5.1 Defining Users' and Source Profiles

Evaluating data freshness implies testing whether user's 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.

5.1.1 Specification of Freshness Expectations

Users have freshness expectations for their applications which should be formulated according to some freshness factors and metrics among those we have seen in section 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 [5][33][23] associated to each object type, to the definition of a specialized language if a preference order is introduced between 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 [5]. Query languages such as *Preference SQL* [21] can be extended to express freshness expectations in each user query.

5.1.2 Acquisition of Source 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 the *age* of data [34][25][31].

- A first question is 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.
- Another important related question is 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 or executing sampling queries can be adapted for freshness metadata acquisition. Techniques for specific metadata should be developed as in [8].

5.2 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 section 3.4, these costs can be modeled differently.

5.2.1 Modeling Query Evaluation Cost

Query evaluation cost models for classical and distributed databases have been studied for a while and are well understood. Some proposals also model queries to cached or materialized data [23][1][9] and hybrid systems that combine materialization techniques in virtual or caching contexts [22][18]. 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 [10] and Potter's Wheel [27], require user interaction whose cost, although hard to estimate, should also be integrated.

5.2.2 Modeling the Update Propagation Cost

Several cost models have been proposed to evaluate update propagation into materialized views [6][37][18]. But as for query evaluation cost models, the update propagation cost models should be extended to represent complex workflow contexts with long transaction 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 [29][33] and temporal storage [22], but the land of hybrid systems is almost unexplored.

5.3 Auditing Freshness of Existing Systems

Having users and source profiles and having cost models, one of the challenging problems is to provide support tools to evaluate data freshness. Among the tools we distinguish those only concerned by the evaluation of data freshness (called quality auditing tools) and those concerned by the design of a system under freshness constraints (called quality-driven engineering tools). This section deals with the former ones while the next section deals with the latter ones.

Several kinds of auditing tools can be conceived, for example:

- Prediction tools, for predicting the freshness of data that can be returned in response to a query, without executing the query.
- Integrated evaluation tools, for measuring data freshness during query execution and labeling delivered data with its freshness levels. These tools can be integrated to the query evaluation process.

- Statistical tools, for taking samples of data freshness 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 integration system 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 freshness of the data delivered by alternative processes in order to choose the process that best suites the user freshness expectations.

Auditing techniques should answer to the following questions: How should the different parameters of the source profile be combined in order to evaluate 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 freshness estimation. There are proposals for combining currency values within a materialized system periodically refreshed [18][33] and for combining timeliness values within a virtual system [26]. 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.

5.4 Quality-driven Engineering

Quality-driven engineering tools support the design of a DIS under freshness constraints. These tools may help in the selection among several design choices:

- Selection between 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 freshness values, or even the comparison of data source profiles.

Obviously, quality-driven engineering tools are based on quality auditing tools, 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 freshness and the proposition of techniques to achieve such improvements. There is a few work done in this line, mostly concerning the study of synchronization policies [7][23][29] or performance [22].

5.5 Link With Other Quality Factors

As improving freshness is not the unique quality goal for a given system, the relationship with other quality properties becomes a major challenge in information system design.

Research described in [26] 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 between 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 [22][29][23]. In the latter, the expected freshness level is taken as a constraint. The main interest of existing works is performance or maintenance cost improvement [5][11][33] but the relation with other quality factors is still unexplored.

6. CONCLUSION

Data freshness represents a family of quality factors and metrics. In this paper we have analyzed these factors and metrics and the features that influence the data freshness evaluation, namely the type of application, the synchronization policy and the nature of data.

The analysis of existing works in terms of a taxonomy suggested open problems in the specification of user expectations, the acquisition of source freshness measures and the formulation of cost models for the query evaluation and update propagation processes of heterogeneous systems. This knowledge can be used both for developing auditing tools that estimate the freshness of an existing system and for designing a system driven by freshness expectations.

Data freshness is a *first class* quality dimension which is more and more required by end-users. Solving the problems listed in this paper opens a door to consider data production as any other item production.

7. REFERENCES

[1] Abiteboul, S.; Duschka, O.: “*Complexity of answering queries using materialized views*”. In Proc. of the 1998 ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS’98), USA, 1998.

[2] Ballow, D.; Wang, R.; Pazer, H.; Tayi, G.: “Modelling Information Manufacturing Systems to Determine Information Product Quality”. Management Science, Vol. 44 (4), April 1998.

[3] Baralis, E.; Paraboschi, S.; Teniente, E.: “*Materialized view selection in a multidimensional database*”. In Proc. of the 23rd Int. Conf. on Very Large Data Bases (VLDB’97), Greece, 1997.

[4] Bouzeghoub, M.; Fabret, F.; Matulovic-Broqué, M.: “*Modeling Data Warehouse Refreshment Process as a Workflow Application*”. In Proc. of the Int. Workshop on Design and Management of Data Warehouses (DMDW’99), Germany, 1999.

[5] Bright, L.; Raschid, L.: "*Using Latency-Recency Profiles for Data Delivery on the Web*". In Proc. of the 28th Int. Conf. on Very Large Databases (VLDB'02), China, 2002.

[6] Chirkova, R.; Halevy, A.; Suciu, D.: "*A formal perspective on the view selection problem*". In Proc. of 27th Int. Conf. on Very Large Databases (VLDB'01), Italy, 2001.

[7] Cho, J.; Garcia-Molina, H.: "*Synchronizing a database to improve freshness*". In Proc. of the 2000 ACM Int. Conf. on Management of Data (SIGMOD'00), USA, 2000.

[8] Cho, J.; Garcia-Molina, H.: "*Estimating frequency of change*". ACM Trans. on Internet Technology (TOIT), Vol. 3 (3): 256-290, 2003.

[9] Gal, A.: "*Obsolescent materialized views in query processing of enterprise information systems*". In Proc. of the 1999 ACM Int. Conf. on Information and Knowledge Management (CIKM'99), USA, 1999.

[10] Galardas, H.; Florescu, D.; Shasha, D.: Simon, E.: "AJAX: An Extensible Data Cleaning Tool". In Proc. of the 2000 ACM Int. Conf. on Management of Data (SIGMOD'00), USA, 2000.

[11] Gancarski, S.; Le Pape, C.; Valduriez, P.: "*Relaxing Freshness to Improve Load Balancing in a Cluster of Autonomous Replicated Databases*". In Proc. of the 5th Workshop on Distributed Data and Structures (WDAS), Greece, 2003.

[12] Gertz, M.; Tamer Ozsu, M.; Saake, G.; Sattler, K.: "*Report on the Dagstuhl Seminar: Data Quality on the Web*". SIGMOD Record Vol. 33(1), March 2004.

[13] Gupta, A.; Mumick, I.: "*Maintenance of Materialized Views: Problems, Techniques, and Applications*". Data Engineering Bulletin, Vol. 18 (2): 3-18, June 1995.

[14] Gupta, H.: "*Selection of Views to Materialize in a Data Warehouse*". In Proc. of the 6th Int. Conf. on Database Theory (ICDT'97), Greece, 1997.

[15] Hammer, J.; Garcia-Molina, H.; Widom, J.; Labio, W.; Zhuge, Y.: "*The Stanford Data Warehousing Project*". IEEE Data Engineering Bulletin, Vol. 18 (2): 41-48, June 1995.

[16] Harinarayan, V.; Rajaraman, A.; Ullman, J.: "*Implementing Data Cubes Efficiently*". In Proc. of the ACM SIGMOD Int. Conf. on Management of Data (SIGMOD'96), Canada, 1996.

[17] Huang, Y.; Sloan, R.; Wolfson, O.: "*Divergence caching in client-server architectures*". In Proc. of the 3rd Int. Conf. on Parallel and Distributed Information Systems (PDIS 94), USA, 1994.

[18] Hull, R.; Zhou, G.: "*A Framework for Supporting Data Integration Using the Materialized and Virtual Approaches*". In Proc. of the 1996 ACM SIGMOD Int. Conf. on Management of Data (SIGMOD'96), Canada, 1996.

[19] Inmon, W.: "*Building the Data Warehouse*". John Wiley & Sons Inc., 1996.

[20] Jarke, M.; Jeusfeld, M.; Quix, C.; Vassiliadis, P.: "*Architecture and Quality in Data Warehouses: An Extended Repository Approach*". Info Systems, Vol. 24(3): 229-253, 1999.

[21] Kieβling, W.; Kôstler, G.: "*Preference SQL - Design, Implementation, Experiences*". In Proc. of the 28th Int. Conf. on Very Large Databases (VLDB'02), China, 2002.

[22] Labrinidis, A.; Roussopoulos, N.: "*Balancing Performance and Data Freshness in Web Database Servers*". In Proc. of the 29th Int. Conf. on Very Large Data Bases (VLDB'03), Germany, 2003.

[23] Li, W.S.; Po, O.; Hsiung, W.P.; Selçuk Candan, K.; Agrawal, D.: "*Freshness-driven adaptive caching for dynamic content Web sites*". Data & Knowledge Engineering (DKE), Vol. 47(2):269-296, 2003.

[24] Ligoudistianos, S.; Sellis, T.; Theodoratos, D.; Vassiliou, Y.: "*Heuristic Algorithms for Designing a Data Warehouse with SPJ Views*". In Proc. of 1st Int. Conf. on Data Warehousing and Knowledge Discovery (DaWaK '99), Italy, 1999.

[25] Mannino, M.; Walter, Z.: "*A Framework for Data Warehouse Refresh Policies*". Technical report CSIS-2004-001, University of Colorado at Denver, 2004.

[26] Naumann, F.; Leser, U.: "*Quality-driven Integration of Heterogeneous Information Systems*". In Proc. of the 25th Int. Conf. on Very Large Databases (VLDB'99), Scotland, 1999.

[27] Raman, V.; Hellerstein, J.: "*Potter's Wheel: An Interactive Data Cleaning System*". In Proc. of the 27th Int. Conf. on Very Large Data Bases (VLDB'01), Italy, 2001.

[28] Redman, T.: "*Data Quality for the Information Age*". Artech House, 1996.

[29] Segev, A.; Weiping, F.: "*Currency-Based Updates to Distributed Materialized Views*". In Proc. of the 6th Int. Conf. on Data Engineering (ICDE'90), USA, 1990.

[30] Sheth, A.; Larson, J.: "*Federated Database Systems for Managing Distributed, Heterogeneous and Autonomous Databases*". ACM Computing Surveys, Vol. 22(3): 186-236, September 1990.

[31] Shin, B.: "*An exploratory Investigation of System Success Factors in Data Warehousing*". Journal of the Association for Information Systems, Vol. 4(2003): 141-170, 2003.

[32] Theodoratos, D.; Sellis, T.: "*Data Warehouse Configuration*". In Proc. of the 23rd Int. Conf. on Very Large DataBases (VLDB'1997), Greece, 1997.

[33] Theodoratos, D.; Bouzeghoub, M.: "*Data Currency Quality Factors in Data Warehouse Design*". In Proc. of the Int. Workshop on Design and Management of Data Warehouses (DMDW'99), Germany, 1999.

[34] Wang, R.; Strong, D.: "*Beyond accuracy: What data quality means to data consumers*". Journal on Management of Information Systems, Vol. 12 (4):5-34, 1996.

[35] Widom, J.: "*Research Problems in Data Warehousing*". In Proc. of the 4th Int. Conf. on Information and Knowledge Management (CIKM'95), USA, 1995.

[36] Wiederhold, G.: "*Mediators in the architecture of future information systems*". IEEE Computer, Vol. 25(3):38-49, 1992.

[37] Yang, J.; Karlapalem, K.; Li, Q.: "*Algorithms for materialized view design in data warehousing environment*". In Proc. of the 23rd Int. Conf. on Very Large DataBases (VLDB'1997), Greece, 1997.

[38] Zhuge, Y.; Garcia-Molina, H.; Wiener, J.: "*Multiple View Consistency for Data Warehousing*". In Proc. of the 13th Int. Conf. on Data Engineering (ICDE'97), UK,1997.