Data Freshness Evaluation in Different Application Scenarios

Verónika Peralta, Mokrane Bouzeghoub

Laboratoire PRISM, Université de Versailles
45, avenue des Etats-Unis
78035, Versailles cedex, FRANCE
{Veronika.Peralta, Mokrane.Bouzeghoub} @prism.uvsq.fr

Abstract. Data freshness has been identified as one of the most important data quality attributes in information systems. This importance increases specially in the context of systems that integrates data of a large set of autonomous data sources. In this paper we describe a quality evaluation framework that allows evaluating the freshness of the data delivered to the user in a data integration system. Concretely, we show the practical use of the framework in different application scenarios and we discuss possible improvement actions for the data integration system in order to fulfill user freshness requirements. In order to illustrate the approach, we discuss data freshness evaluation issues with several examples.

1 Introduction

Data freshness has been identified as one of the most important attributes of data quality for data consumers (Shin 2003)(Wang et al. 1996). Specifically, 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 (Naumann et al. 1999). This paper deals with data freshness evaluation in the context of a Data Integration System (DIS) that integrates data from different independent data sources and provides the users with a uniform access to the data.

Data freshness represents a family of quality factors. With regard to data freshness, two factors have been proposed in the literature: *currency* that describes how *stale* is data with respect to the sources and *timeliness* that describes how *old* is data. In (Bouzeghoub et al. 2004) we analyze these factors and several metrics proposed to measure them.

In (Peralta et al. 2004), we proposed a framework for analyzing and evaluating data freshness based on a calculation dag which abstracts a workflow of integration activities. After a brief recall of this framework, this paper shows how the framework can practically be used in different application scenarios and how the data integration system can be improved in order to fulfill user requirements in terms of data freshness.

The rest of the document is organized as follows: Section 2 briefly describes the data quality evaluation framework. Section 3 discusses how to use this framework through different application scenarios. Section 4 focuses on the possible improvement actions to put on the DIS workflow to achieve user requirements. Finally, section 5 concludes with our general remarks.

2 The Data Quality Evaluation Framework

In this section we briefly describe the framework for data quality evaluation. The framework models the DIS processes and properties and evaluates the quality (particularly the freshness) of the data returned to the user.

The framework consists of: (i) a set of available data sources, (ii) a set of classes of user queries, (iii) the DIS integration processes, (iv) a set of properties describing DIS features (costs, delays, policies, strategies, constraints, etc.) and quality measures, and (v) a set of quality evaluation algorithms.

The DIS is modeled as a workflow in which the activities perform the different tasks that extract, transform and convey data to end-users. Each workflow activity takes data from sources or other activities and produces result data that can be used as input for other activities. Then, data traverses a path from sources to users where it is transformed and processed according to the system logics.

The framework represents the DIS dataflow by means of a *labeled calculation dag (LCDag)* that describes the involved activities, their inputs, outputs and properties. Formally, a LCDag is a dag G = <V, E, P, *propvalue*> defined as follows: The nodes in V are of three types: *source nodes* (with no input edges), *target nodes* (with no output edges) and *activity nodes* (with both input and output edges), which represent sources, user queries and DIS activities respectively. The edges in E represent that a node is calculated from another (data flows in the sense of the arrow). P is a set of properties describing DIS features and quality measures, and *propvalue* is a partial labeling function that assigns a property value to a node or edge of the dag. Figure 1 shows the LCDag graphical representation (nodes and edges are labeled with *property=value* pairs). The three LCDags of figure 1 are discussed in section 3.2.

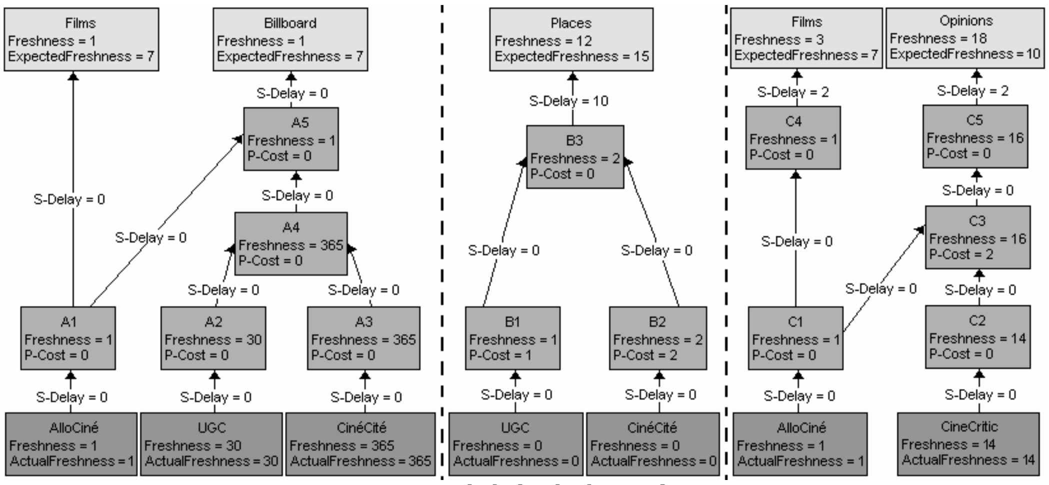

FIG. 1 – *Labeled calculation dags*

The quality evaluation is performed by evaluation algorithms. The input information needed by the evaluation algorithms is contained in the LCDag. It consists of property values, specifically, source quality values (labels of source nodes) and DIS property values (labels of activity nodes and edges). The algorithms take as input a LCDag, combine property values calculating freshness values corresponding to the data returned by queries, and return the LCDag with an additional property (corresponding to the data freshness quality factor).

3 Data Freshness Evaluation

In this section we describe the general evaluation approach. We firstly give an intuitive idea of the freshness calculation strategy and we describe a base evaluation algorithm. Then, we discuss the instantiation of the base algorithm to different application scenarios.

3.1 General Approach

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 execution delay is influenced not only by the processing cost of each activity but also by the delays that can exist between the executions of consecutive activities.

We briefly describe such properties as well as user expectations:

– *Processing cost*: It is the amount of time, in the worst case, that an activity needs for reading input data, executing and building result data.
– *Synchronization delay*: It is the amount of time passed between the executions of two consecutive activities.
– *Actual freshness*: It is a measure of the freshness of data in a source, which can be provided by the source or can be estimated or bounded by the DIS.
– *Expected Freshness*: It is the desired data freshness specified by the user. It measures the extent to which the freshness of the data is appropriate for the task on hand.

Our base algorithm takes into account such properties. It traverses the dag, from sources to queries (the sense of the data flow), calculating the freshness of the data produced by each node. The algorithm idea can be sketched as follows:

→ For a source node A:
  Freshness(A) = getActualFreshness(A)
→ For a non-source node A, and the set of all its predecessors P:
  Freshness(A) = combine {Freshness(B) +getSyncDelay(B,A) /B $\in$ P} + getProcCost(A)

For source nodes, data freshness is the source actual freshness. For the other nodes, the freshness of the data produced by a node is calculated as the freshness of data at the moment of reading it (the freshness of data produced by the predecessor) plus the synchronization delay plus the processing cost. When a node reads data from several input nodes, input freshness values should be combined, for example, taking the maximum value.

We have implemented a data freshness auditing tool that implements the framework and allows evaluating the freshness of the data returned to the user in different application scenarios (Fajardo et al. 2004). Next section illustrates the approach with several examples.

3.2 Illustrating Examples

Consider three simple DIS that deal with information about cinemas and films:

- **DIS_1**: A mediation system that answers queries about films and the cinemas where they are in billboard. Typical queries are “Where can I see a film?” or “Which films are in billboard now?”
- **DIS_2**: A web portal that caches information about cinemas and the availability of places for their performances. Typical queries are “Where are available places to see a film?” or “How many places are available in a cinema?”
- **DIS_3**: A data warehousing system that stores statistic information about films, the number of persons that watch each film and their opinions. Typical questions are “Which films have the best ranking this week?” or “Which film should I watch?”

Users of DIS_1 and DIS_3 are concerned with *timeliness* but users of DIS_2 evaluate *currency*.

Figure 1 shows a simplified version of the three DIS, accessing to a small number of sources. DIS_1 extracts film information from AlloCiné (via wrapper A_1) and cinema information from UGC and CinéCité (via wrappers A_2 and A_3). Activity A_4 merges the information from both cinema sites and activity A_5 joins film and cinema information. DIS_2 extracts place information from UGC and CinéCité. Activity B_3 is the cache core, that receives user requests and asks the sources when the cache needs refreshment (invoking wrappers B_1 and B_2). DIS_3 extracts film audience statistics from AlloCiné (via wrapper C_1) and spectator’s opinions from CineCritic (via wrapper C_2). Activity C_3 reconciles data from both wrappers and activities C_4 and C_5 perform aggregations and calculate statistic data.

In the LCDags of figure 1, source nodes are labeled with their *actual freshness*, target nodes are labeled with *expected freshness*, activity nodes are labeled with *processing costs* (P-Cost) and edges are labeled with *synchronization delays* (S-Delay). In next section we discuss how to obtain such values. Values are expressed in days for DIS_1 and DIS_3 but in minutes for DIS_2. Note that the “zeros” represent negligible values.

The base evaluation algorithm adds the *freshness* property to all the nodes. It first calculates freshness for source nodes, as its source actual freshness (e.g. 1, 30 and 365 days for source nodes of DIS_1). Then, the algorithm traverses the dag calculating the freshness of a node adding the predecessor freshness plus the synchronization delay plus the processing cost. For example, for activity B_2 freshness is 2 minutes (0 +0 +2 =2). When a node has several predecessors, their freshness values are combined. For example, for activity C_3 freshness is the maximum between (1 +0 +2 =3) and (14 +0 +2 =16), i.e. 16 days. However, users of DIS_1 expect to know how fresh is film information, independently to when the cinema data was last updated. DIS_1 assigns priorities to the nodes and the combination function takes the freshness value of the predecessor with highest priority (e.g. the freshness of A_5 is 1 day (1 +0 +0).

3.3 Instantiating the Framework

Data freshness is evaluated based on the source actual freshness, processing cost and synchronization delay properties. But the relevance of these properties depends on the particular scenario considered. A first remark is that its magnitude should not be considered in the absolute but compared to freshness expectations. For example, users of DIS_1 may tolerate data freshness of "7 days" so, the processing costs and synchronization delays ("some minutes") are negligible; however, users of DIS_2 require "extremely fresh" data, so, the processing costs of activities could be relevant. In addition, in the scenarios where the focus is data currency, the source actual freshness is not relevant. For example, in DIS_2, it does not matter "how old is data in the sources"; the focus is in retrieving the same data that is stored in the sources.

Another aspect is how to calculate the source actual freshness, the processing delay and the synchronization delay. Depending on the scenario, different DIS properties may influence their calculation. For example, in DIS_2 the processing cost of the wrappers is dominated by the cost of communicating with the sources. In DIS_3 and DIS_2 the materialization/caching of data introduces important synchronization delays, so the refreshment policies and frequencies are important properties to take into account. In virtual systems as DIS_1, these properties have no sense.

We propose a method for instantiating the base algorithm for adapting to the specific properties of a given scenario. The mechanism consists in overloading the following abstract functions: *getProcCost*, *getSyncDelay* and *getActualFreshness*, which calculate the respective properties according to the specific scenario. For example, for DIS_1 the processing costs and synchronization delays are negligible so the respective functions can return zero. However, source actual freshness is relevant; the *getActualFreshness* function can estimate the values from source update frequencies, for example. For DIS_3 the relevant processing costs are due to the reconciliation process (activity C_3), which may require human interaction (to solve conflicts or errors) and can last two days. Costs can be estimated using cost models or statistics or can be filled in by experts. DIS_3 materializes the data produced by C_4 and C_5; the synchronization delays with target nodes can be bounded by the refreshment frequencies. The *combine* function can also be overloaded. For example, DIS_3 takes the maximum of input nodes freshness while DIS_1 combines input values considering node priorities, as illustrated in previous section.

4 Data Freshness Enforcement

The system should provide at the query level the data freshness expected by the users. To know if user freshness expectations can be achieved by the DIS, we can calculate the freshness values for target nodes and compare them with those expected by users. If freshness expectations are not achieved, we have to improve the system design to enforce freshness or negotiate with source data providers or users to relax constraints. In this section we discuss these ideas.

4.1 Improving DIS design

Observe that for each node, it can exist a path from a source for which we add all synchronization delays and processing costs to the source actual freshness and we obtain the freshness of the node. For example, the freshness of activity C_5 can be calculated adding source actual freshness, processing costs and synchronizations delays in the path [CineCritic,C_2,C_3,C_5]. This path is called the *critical path* and represents the bottleneck for the freshness calculation. The existence of the critical path depends on the definition of the *combine* function; taking the maximum of predecessors' freshness, the critical path always exists.

The freshness of the data delivered to the user may be improved optimizing the design and implementation of the activities in order to reduce their processing cost, or synchronizing the activities in order to reduce the delay between them. Sometimes the changes can be concentrated in the critical path that slows the system. Other times a complete reengineering of the whole system is necessary, either changing the algorithms that implement the activities, the synchronization policies, the decisions of which data to materialize or even the hardware. The synchronization of some activities implies finding the most appropriate execution frequencies for some activities respecting possible source access constraints. The main difficulty resides in the synchronization of activities having several inputs with different refreshment policies.

The auditing tool allows identifying the critical path, changing property values in order to test alternative configurations and re-executing the evaluation algorithms to see the effects of the changes. In this sense, the tool brings an aggregate value to the auditing functionalities.

4.2 Selection between alternative implementations: bottom-up propagation

If the design of the DIS cannot be improved, an alternative is negotiating with users to relax freshness expectations. The freshness values calculated using the evaluation algorithm can help users to know the freshness that the DIS can guarantee for the returned data.

Observe that the evaluation algorithm propagates freshness values from sources to queries, i.e. a bottom-up propagation (following the dataflow of the DIS).

A direct application of this bottom-up strategy is the selection between alternative implementations of the system. The DIS can offer the users several alternative processes to answer their queries and users can choose (or the DIS can choose for them) the process with the best quality. For example, even improving activities design and synchronization, the freshness expectations of the *Opinions* query cannot be achieved because of the actual freshness of the *CineCritic* source. Considering an alternative process that queries other sources can be a solution.

In this line, we have used the freshness evaluation tool within a system that automatically generates mediation queries. The tool was used for evaluating the quality of the generated queries, both in virtual and materialized scenarios, in order to select the best one for answering a given user query (Kostadinov et al. 2004).

4.3 Selection of alternative data sources: top-down propagation

Another alternative to enforce freshness is negotiating with source data providers to relax source constraints. Sometimes the system hardware can be powered to support more frequent accesses to the sources. Other times, this alternative implies demanding and eventually paying for a better service, for example, receiving data with a lower actual freshness.

Analogously to the bottom-up propagation, we can propagate freshness expectations from queries to sources (subtracting processing costs and synchronization delays). The top-down propagation algorithm is similar to the bottom-up one, but the *combine* function must consider nodes with several successors. The propagated freshness expectations can help the DIS designer to know the freshness that he must ask the source provider for.

A direct application of this top-down strategy is the selection between alternative data sources to achieve freshness expectations. For example, propagating down freshness expectations for the *Opinions* query we obtain a bound (6 days) for the actual freshness of the source providing user's opinions. This avoids considering sources as *CineCritic* that have greater actual values.

5 Conclusion

In this paper we addressed the problem of evaluating data freshness in a data integration system. We presented a quality evaluation framework and its practical use for evaluating data freshness in different application scenarios. The framework was implemented in a quality auditing tool that can be instantiated for evaluating data freshness in a concrete scenario. The tool supports the top-down and bottom-up propagation strategies in order to help the user to improve freshness.

We are now working in the development of a toolkit for implementing the instantiation in a semi-automatic way. In the future, our goal is to confront the results with user quality profiles.

References

Bouzeghoub M., Peralta V. (2004), A Framework for Analysis of Data Freshness, in Proc. of the Int. Workshop on Information Quality in Information Systems (IQIS'2004), collocated with SIGMOD'2004, France, 2004.

Fajardo F., Crispino I., Peralta V. (2004), DWE: Una Herramienta para Evaluar la Calidad de los Datos en un Sistema de Integración, in Proc. of the X Congreso Argentino de Computación (CACIC'04), Argentine, 2004.

Kostadinov D., Peralta V., Soukane A., Xue X. (2004), Système adaptatif d'aide à la génération de requêtes de médiation, short paper and demonstration, in Proc. of the 20èmes Journées de Bases de Données Avancées (BDA'2004), France, 2004.

Naumann F., Leser U. (1999), Quality-driven Integration of Heterogeneous Information Systems, Proc. of the 25th Int. Conf. on Very Large Databases (VLDB'99), Scotland, 1999.

Peralta V., Ruggia, R.; Kedad, Z.; Bouzeghoub M. (2004), A Framework for Data Quality Evaluation in a Data Integration System, Proc. of the 19º Simposio Brasileiro de Banco de Dados (SBBD'2004), Brazil, 2004.

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

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