ProLD: Propagate Linked Data

Peter Kalchgruber

University of Vienna, Faculty of Computer Science, Liebiggasse 4/3-4, A-1010 Vienna
peter.kalchgruber@univie.ac.at

Abstract. Since the Web of Data consists of different data sources maintained by different authorities the up-to-dateness of the resources varies a lot. However a number of applications are built upon that. To tackle the problem of out-dated resources, we propose to develop a framework that utilizes the linkage between Linked Data nodes to propagate updates in the cloud. For that purpose we have observed propagation strategies developed in the database domain and have created a list of currently unsolved problems which emphasize the difference between the propagation in the Web of Data and state of the art approaches. Apart from the improvement of the up-to-dateness of data, by following the approach of propagation the network improves and inconsistencies will be reduced.

1 Introduction

The Web of Data is growing continuously[1]. As of today, a large amount of applications[2] already utilize data from the Linked Open Data (LOD) cloud; however, the experience of such applications closely correlates with the data quality of the underlying data. Bizer [1] illustrates that the timeliness beside others (accuracy, completeness) is one of the most popular dimension of information quality.

The Web of Data reflects knowledge about things of the real world. Changes in the real world, in terms of updates, will be executed in the Web, by the maintaining data node owner or local community. The problem is, that currently it takes a long time, until all concerned resources in the cloud become updated and a consistent status is achieved. For example, taken the actual political changes in the country Egypt, it is expected, that the Parliament website, will update the change of government directly. Only after the time Δt, news sites, encyclopedias and other data nodes will update their Web resources. Δt varies depending on the maintaining data node owner or community and the underlying technology. For example, due to the required manual effort, data of the most linked

[1] In May 2009 the Linked Data cloud of 4.7 billion triples [2]. By February 2011 it had grown to 27 billion triples.
http://www4.wiwiss.fu-berlin.de/lodcloud/state/#structure

[2] http://www.w3.org/wiki/SweoIG/TaskForces/CommunityProjects/
/LinkingOpenData/Applications

A. Harth and N. Koch (Eds.): ICWE 2011 Workshops, LNCS 7059, pp. 307–311, 2011.

data node DBpedia is currently only updated about semiannually. The example[3] demonstrates that, real world updates are visible in the Web of Data, but they are not propagated to data sets they are interlinked with.

Our objective is to propagate updates of resources to all linked resources with the same identity in the cloud. Although it is obvious to use database propagation approaches to tackle the problem, we discovered several problems on using them in Section 2. As a possible solution we present our prototypical framework in Section 3.

2 Background and Related Work

Globally, the Web of Data contains huge sets of redundant partially linked data. Thus it bears some similarities with databases, also in context of propagation of updates. But we found a list of differences and open problems displaying the non-triviality of propagation in the Web of Data. Based on the following list, we discuss the approaches in related work and define our field of research:

- Schema mapping (neither unique IDs, nor unique vocabulary)
- Availability of data nodes (online/offline)
- Properties stating equality between resources not always reliable
- Synchronization with relational databases (Wikipedia ↔ DBpedia)
- Trust / Authority (every triple must be read as claim from the corresponding data node not as a fact [8])
- Propagation of triples which do not exist at all nodes
- Behavior if data nodes drop their data and do a general dump import (as DBpedia is built on a regularly basis)

As known from related domains (e.g. Distributed Database Management Systems) updating distributed data sources is a critical issue. The approach of propagation in the Web of Data expose similar characteristics as data propagation [10] and data replication [6]. However the linkage between data nodes is neither symmetric nor complete and due to the AAA principle[4] (data can be incomplete or inconsistent) the use of this traditional approaches are not applicable in the Web of Data.

Data nodes use a mix of commonly used vocabularies for describing their data. This mix is often not sufficient, and therefore expanded with proprietary terms [3]. This results in heterogeneity and problems with the interoperability of metadata. It is still difficult to map metadata with the same semantic meaning together although there are already different approaches to map metadata [9,3] or match ontologies [5]. However our tests have shown that they are not implemented broadly.

[3] Change of government in Egypt: Only the data node Freebase has updated its data about the recent change of the regime. However DBpedia, DBpedia Live, New York Times still hold data of the previous state, although they are linked as identical.

[4] Anyone can say Anything, Anywhere

Another problem is the *non-assured availability* of data sets. Anytime a server can be temporarily unavailable, or stop its service. If a data node is offline for a while, it needs to receive the missed updates, after it has recovered. It cannot know which server has propagated updates in the meanwhile. Since there is no master server, as in traditional distributed databases, this information must be saved in the cloud.

sparqlPuSH [11] allows clients to get informed about data updates in RDF stores via *PubSubHubbub*[5], a simple open server-to-server publish/subscribe protocol. *sparqlPuSH* uses the *PubSubHubbub* infrastructure to notify clients over hubs via a push based approach. *sparqlPuSH* can only be used for the notification use case and could be used in the framework to notify remote datasets about updates.

The property `owl:sameAs` of the Web Ontology Language (OWL)[6] per definition [4] is interconnecting equivalent resources between two data sets. It indicates that two resources exhibiting different URIs actually refer to the same resource - they share the same "identity". But sometimes, historical resources (e.g. the resource *East Berlin*) are set equivalent with new ones [7], or no distinction is made between the context of resources (e.g. Republic of Ireland vs. Island Ireland). Strategies for the aggregation of equivalent resources are proposed in [4]. A more detailed ontology of `owl:sameAs` has been proposed by Halpin [7]. However, the currently inappropriate usage of `owl:sameAs` makes it difficult to build a framework on it.

3 Approach

Our overall objective is to develop an efficient update strategy for Linked Data. Once a resource at a single node become updated, by means of propagation all resources with the same identity in the cloud, should receive this update, too. Our framework called *ProLD* is based on RDF properties that define the equality between linked resources (e.g. `owl:sameAs`). ProLD follows links between same identities on distributed data nodes to propagate the updates in the cloud.

It can be used as an add-on to existing services, to help data node owners to propagate updates made on datasets that are under their control and to receive updates from other datasets, which propagate their updates. The ProLD Framework consists of three elements:

- The *Observer* is responsible to detect local changes at a dataset. It searches for equivalent resources (e.g. marked with `owl:sameAs`), compose a propagation package and handle it to the *Propagator*. The Observer has an interface to commonly used RDF storages such as Virtuoso, Jena, or 4store.
- The *Receiver* receives updates sent from the *Propagator* of other data nodes, does integrity checks and triggers the changes at the local dataset.
- The *Propagator* receives a list from the Observer containing update packages for remote resources. It propagates the updates to the cloud.

[5] http://code.google.com/p/pubsubhubbub/

[6] http://www.w3.org/TR/owl-ref/

The Observer can be informed about changes at the local triple store by change logs created by triple stores, or by adding triggers to the database. Once it becomes informed about a change at a local data set, it scans the local resource for equivalent resources in the cloud. An object with the collected information will be sent to the Propagator. The Propagator creates a unique hash value of the package and saves it with a time stamp in a local buffer. Thereafter, it sends a package to all servers listed in the `owl:sameAs` field of the local resource. The Receiver at the remote data set is listening to receive packages. After the integrity check, it asks the Propagator, whether the package was already processed by comparing hash-values with a buffer list. If the information about the updated resource is new, there are different ways to proceed: Either the tool is configured in automatic mode, it will do the changes at the local resource, scan for `owl:sameAs` values and hand the package (as described above) to the Propagator. Alternatively, in semi-automatic mode the update could be reviewed by local quality review programs, a community or the data node owners.

In case of an update, the modification package will always include the old and the new triples. This enables the Receiver to decide carefully based on rules whether a.) the subject resource of the package is equal to the local resource and b.) both triples refer to the same property. Thereby it use schema concepts such as sub-property or super-property to take a decision. It is also being considered to add more surrounding triples to the modification package, to help to distinguish between similar resources by the identification of the resources by fingerprints.

3.1 Scenario Open Government Data

Governments increasingly use the cloud to expose their data. The owners of *data.gov.eg*, DBpedia and Freebase have installed ProLD on their data nodes. The government of Egypt will update the form of government at the local resource `eg:Government` to "Military junta". The Observer detects the change and sends the required update information to the Propagator. Data nodes with identical resources (e.g. DBpedia or Freebase) listed as `owl:sameAs` at `eg:Government` will receive this propagation package. Before they process the content, each Receiver proofs if the package was not processed earlier by comparing the hash-value and timestamp with the list of already processed packages. If so, it drops the package. Otherwise the Receiver updates the local resource, looks for local `owl:sameAs` resources and forwards the package to their data nodes.

4 Research Methodology

Currently, there is no solution to handle update propagation in the Web of Data. The previous section indicates that there are several solutions for partial problems but also many open problems. But research about a general view of the problem has not been done so far.

In particular we are concerned with the following research questions:

- Which protocols and techniques are available and which are required to allow propagation in the Web of Data?
- How can propagation of updates be performed considering its scalability?

Our methodology to solve the research questions consists of three phases: First additional research in the state of the art and technology evaluation needs to be done. Based on the results a prototype will be developed. It will be improved and extended incrementally through testing in real life scenarios. Finally, the third phase contains the evaluation of the framework. There the time until updates appear in data nodes using ProLD is compared with data nodes not using the framework. Furthermore the rate of successful updates should be compared with error cases.

5 Conclusion

In this paper we have discussed several problems concerning the propagation of updates in the Web of Data. Although the Web of Data can be seen as a big distributed database, research in the field of propagation of updates in the cloud has shown that there are so far several unsolved problems. The combination of existing approaches reveals that propagation is reasonable and possible. A sketched rough concept of our framework gives an overview, how propagation could be done in the cloud. Thus, the time until updates in the cloud are performed can be reduced to a minimum.

References

1. Bizer, C.: Quality-Driven Information Filtering- In the Context of Web-Based Information Systems. VDM Verlag, Saarbrücken (2007)
2. Bizer, C., Heath, T., Berners-Lee, T.: Linked data – the story so far. Int. J. Semantic Web Inf. Syst. 5(3), 1–22 (2009)
3. Bizer, C., Schultz, A.: The R2R Framework: Publishing and Discovering Mappings on the Web. In: COLD 2010, Shanghai (2010)
4. Ding, L., Shinavier, J., Finin, T., McGuinness, D.L.: owl:sameAs and Linked Data: An Empirical Study . In: Proc. of the 2nd Web Science Conference, Raleigh NC, USA (April 2010)
5. Euzenat, J., Shvaiko, P.: Ontology Matching. Springer, Berlin (2007)
6. Gifford, D.K.: Weighted voting for replicated data. In: Proc. of the 7th ACM SOSP, SOSP 1979, pp. 150–162. ACM, NY (1979)
7. Halpin, H., Hayes, P.J.: When owl:sameAs isn't the Same: An Analysis of Identity Links on the Semantic Web. In: CEUR Workshop Proceedings (2010)
8. Hartig, O., Langegger, A.: A database perspective on consuming linked data on the web. Datenbank-Spektrum 10, 57–66 (2010)
9. Haslhofer, B.: A Web-based Mapping Technique for Establishing Metadata Interoperability. PhD thesis (November 2008)
10. IBM. IMS DataPropagator Implementation Guide. IBM, 650 Harry Road San Jose, 3 edn. (2002)
11. Passant, A., Mendes, P.N.: sparqlpush: Proactive notification of data updates in rdf stores using pubsubhubbub. In: 6th Workshop on Scripting and Development for the Semantic Web (May 2010)