Replication for Linked Data

Laurens Rietveld*

Department of Computer Science,
VU University Amsterdam,
The Netherlands
laurens.rietveld@vu.nl

Abstract. With the Semantic Web scaling up, and more triple-stores with *update* facilities being available, the need for higher levels of simultaneous triple-stores with identical information becomes more and more urgent. However, where such Data Replication approaches are common in the database community, there is no comprehensive approach for data replication for the Semantic Web. In this research proposal, we will discuss the problem space and scenarios of data replication in the Semantic Web, and explain how we plan on dealing with this issue.

Keywords: synchronization, replication, decentralized triple-stores, read-write Web.

1 Problem Description

Up until recently, Semantic Web applications often made use of *read-only* triple-stores. These application are now taking up on using SPARQL-1.1 new 'update' facility, allowing users to *write* to triple-stores.

However, if triple-store contents change faster than they currently do, replication of Linked Data becomes a real problem, *i.e.* the challenge of keeping the information consistent between different data hubs. The role of these stores changes from that of a static content delivery system to a read and write content deposit. "Personal data lockers" [19] are an example of this scenario. They allow users to push information for it to be pulled by a variety of consumers. This high-frequency, dynamic information exchange between triple-stores requires efficient replication processes optimized for dealing with at least moderate volumes of data.

Although data replication is a very well-studied issue for databases and in file-synchronization for large-scale programming efforts, there is little work done with a particular focus on Semantic Web infrastructure. In this light, our hypothesis is:

> An efficient and comprehensive Linked Data replication approach requires more than the existing data replication techniques.

* This work was supported by the Dutch national program COMMIT.

P. Cudré-Mauroux et al. (Eds.): ISWC 2012, Part II, LNCS 7650, pp. 415–423, 2012.

2 Relevance

This section explains why Data Replication is relevant for Linked Data, by describing three use cases.

2.1 Triple Store Mirroring

Semantic Web applications (in our case Hubble[1], a Clinical Decision Support prototype using Linked Data) often rely on external triple-stores from the linked open data cloud. When one triple-store is slow or down, this has an effect on the responsiveness of the application. In scenarios where clinicians can walk around in the hospital with Hubble on their tablet, unreliable connections should not hinder application functionality. How can we ensure that most of the application still functions without internet connection? Linked Data Replication allows us to mirror an external triple-store either locally or to another server. This way we avoid direct dependencies on external triple stores: the application uses the mirrored triple-store, and the mirrored triple-store is a full replication of the master. In some cases, *partial* data replication is sufficient, e.g. when the application only relies on a subset of the data provided by the triple-store. Such applications where connectivity is unreliable is becoming more common in the Semantic Web domain, as more and more Semantic Web applications are appearing on hardware such as smart phones and tablets.

2.2 Annotations on Census Data

This use-case involves Dutch historical census data available as Linked Data. Annotation of original sources is one of the core activities of historical researchers. However, they are typically only interested in a subset of the dataset. One can consider a master triple-store containing all the data, and several subsets of the triple-store used by different researchers. Any annotation made on the subset, should be propagated to the master. Vice versa, any change made to the master, should be propagated to all the subsets. This scenario involves a combination of the problems of concurrent editing, dealing with version conflicts, and partial replication.

2.3 SemanticXO and Backups

Where the previous use-cases made use of a central 'master' triple-store, this is not always the case. The following use-case is an example of decentralized partial replication for Linked Data. The XO laptop is part of the One Laptop per Child (OLPC) project which aim is to create educational opportunities for the worlds poorest children by providing each child with a "rugged, low-cost, low-power, connected laptop". *Semantic*XO is a project which aims at providing an infrastructure to integrate the programs running on the XO into the Web

[1] `https://github.com/Data2Semantics/Hubble`

of Data. These programs can then publish and consume content to and from the WoD using the XO as the data provider [9]. The SemanticXO's all contain their own local triple-store. Due to unreliable internet connections, the laptops are not always connected to a central server or to the internet. Therefore communication between laptops in a mesh network has added value: exchanging information without the need to be connected to a network router. Currently, a-synchronous decentralized transfer of data is not possible. This makes tasks such as backup difficult. By using the SemanticXO triple-stores, the graphs in each triple-store can be replicated to other laptops. This way, the application data of each SemanticXO is backed-up on other XO laptops or XO servers. Because the SemanticXO's operate in environments with numerous constraints, the data replication functionality needs to adapt to these constraints [15] (e.g. by deciding which graphs in the triple-store to replicate, and in which order).

3 Related Work

3.1 Database Replication

Database replication is often used to improve performance and/or to improve availability [5]. The majority of database replication techniques are based on the state machine approach [18]. This approach ensures that replicated databases which share the same initial state and execute the same requests in the same order, will do the same thing and produce the same output. Inconsistent networks (i.e. unreliable networks, or networks with replicas which are not always online) often require a 2-phase or 3-phase commit protocol for every request, to maintain a consistent view. These protocols impose a substantial communication cost on each database transaction. Research into group communication protocols [2] reduces this overhead by avoiding the need to use these protocols on a *per action* basis while still maintaining a global persistent order. The state machine approach is usable for Linked Data, but requires server access; something which is not always the case when replicating remote triple stores.

Research into partial database replication [1,10,21] shows that full replication approaches are not directly applicable to partial replication scenarios. One of the problems in partial replication is that insert/update queries might rely on data which is missing on a partial replica. An approach to deal with this issue is described in [10], where the transaction logs are send to every replica, regardless of the replica holding a copy of the modified data. The replica only updates data items for which it holds a copy. If data items are referenced for which is does not hold a copy, the replica requests this information from the original server. As a downside, the replica might often receive transactions which it will not execute, thus creating unneeded overhead of network traffic. Additionally, because of the connectivity in graph structures, the approach of requesting missing information from the original server is not trivial task. In a Semantic Web scenario, an insert/select query executed on a partial replica has no way of knowing whether

an empty results from the *where* clause is caused by missing information on the partial replica, or whether this information should be absent anyway (i.e. is also missing on the original triple-store).

3.2 Ontology Differences

Related work on ontology differences is often inspired by the classical Version Control Systems. In [11], the causes, problems, and an approach for dealing with ontology changes are described, to achieve maintaining of interoperability while ontologies change. Here, the approach for dealing with different ontology versions is by comparing ontological classes, and displaying these side-by-side in RDF/XML.

[4] contains a description on how to formalize the differences betweevoidn graphs. Such differences can then be used for the updating and synchronization of graphs. A distinction is made between *weak* and *strong* patches, where a weak patch is only applicable to the same graph it was computed from, and a strong patch specifies the changes in a more context dependent manner. A weak patch is similar to the database replication methodology described above. A strong patch provides a way to deal with propagating these changes to partial replica.

Other related work on ontology differences is from [6,13] which focus on representing changes made to ontologies. Additionally, work from [8] resulted in an implementation (Protégé plugin) where the semantic differences between ontologies is calculated.

What most of these approaches have in common is the need to calculate the entailment and compare the complete graphs. For Linked Data replication this is often too heavy to perform, especially if close to any-time behaviour is desired. Research on incremental and stream reasoning ([3,7]) however show promising results on the time it takes to calculate the entailment.

Another common aspect of these approaches is their unsuitability for partial data replication, as in such a situation both graphs (the full master, and the partial slave) will always be different.

3.3 Linked Data Replication

One example of work on Linked Data Replication is RDFSync[22]. Here, full data replication between triple stores is achieved by decomposing the graphs into smaller *Minimum Self-Contained Graphs* (MSGs). By comparing the hashes of the MSGs of both triple store, the algorithm selects the MSGs it needs to transfer. This way, only the difference (including a certain amount of overhead) between triple stores is transferred. This approach however does not cover the complete problem space of Linked Data replication. RDFSync does not take into account partial data replication, and it requires installation on both servers; something which is not always possible.

An example of partial data replication is [16], where partial data replication in a master/master network is applied in the domain of mobile devices. Relatively heavy operations such as conflict resolution and merging, is done on the server

(i.e. triple store), which lead to low hardware requirements for the mobile devices. This approach requires a high level of server access on the triple store, something which is not always possible. This work is continued in [23], where the partial replication is made context-dependent (e.g. by user location or language).

Work on p2p Semantic Wiki's focusses mainly on concurrent editing and resolving conflicts in a full replication scenario. Some approaches (e.g. giki) use the GIT versioning system as underlying tool to deal with concurrent editing. Another approach is done by [20], where collaborative editing techniques from regular (non-Semantic-Web) p2p wikis (e.g. WOOT [12]) are extended to deal with the RDF model. These full data replication approaches deal with master/master networks, and all require a high level of server access to perform.

Finally, strongly related to Linked Data Replication is sparqlPuSH [14], which provides a mechanism to get notifications when the content of a triple-store is updated. Although this does require server access (installation) on the triple store server, it might be useful in the context of partial data replication, as the update mechanism supports notifications on subsections of the content.

4 Problem Space

4.1 Dimensions

We consider the problem space of data replication for RDF data to contain the following six dimensions: network structure, partiality, size, difftype, access level and time granularity.

Network Structure (Master-Master vs. Master-Slave): A network of master-master nodes contains nodes which can all perform updates. The changes each node makes are propagated to the other nodes. A master-master network introduces problems such as concurrent editing. How can such a network deal with conflicts when the same information is changed at the same time on two triple stores. The alternative to a master-master network is a network of master-slave nodes, where only the master has permission to update a graph, and the slaves have read-only rights. Data is then replicated from the master to the slave.

Partiality (Full vs partial data replication): Partial replication increases the data replication task considerably. How to detect changes related to the subset being replicated is one of the challenges, and how to define and support the views of data that should be replicated. Can we use rankings in the data to select the 'important' part of the graph to replicate, or use application/user profiles to detect what the information needs of the applications or users are.

Network Size : The more nodes there are in the network, the more urgent problems such as complexity and performance become. This is especially the case for master-master networks.

DiffType : Where the dimensions above are of a technical or infrastructural nature, and contain (almost) binary classes, this dimension is more targeted towards logics and is more scalular than binary. On one end of the scale we have the structural difftype, where the other values on the scale (increasing in complexity) makes use of semantics. The structural difftype essentially compares serializations of two triple stores. This does not account for the same knowledge represented differently in both stores.

Access level (No access vs. Black Box vs. White Box): The possible approaches for data replication depend on the abstraction level of the triple-stores. Some scenarios might require data replication to be implemented using a black box approach: the replication framework should work on all kinds of triple stores, and have no access to the lower level functionality of those stores. Other scenarios might require data replication where a lower level of functionality and triple-store access is required. This often results in different implementations for different triple-store vendors, as the architecture of the stores differ. Alternatively, there are situations where one might want to replicate a triple store without server access, and with only SPARQL access. This decreases the possible solutions considerably, as there is no way to install for instance a custom middle-layer (e.g. used to track changes to the triple store) on the server.

Time Granularity : How often does a triple-store change? Can changes occur any minute, or is it only updated once a year? The requirements and available solutions differ greatly between both.

4.2 Methods

There are three methods for Data Replication. These methods differ in applicability for each of the dimensions above.

1. Copying the complete graph. This is relatively inefficient, as often just a part of the graph changed.
2. Propagating the update queries, or bulks of update queries. This approach is difficult for partial data replication, as update queries on the full triple stores have a different context than the same queries on the partial triple store. This can result in different data being inserted in both triple-stores.
3. Propagating the actual difference between triple stores, either after a change has been made or at larger intervals (e.g. depending on the update frequency/time granularity of the triple-store). This requires knowing what has changed, and a formalization of this difference.

4.3 Replication for Linked Data vs. Database Replication

The replication scenarios of Linked Data and database replication differ greatly. Database replication scenarios often involve a closed network with large control over the different database servers. Linked Data however is an open network,

with one public query protocol standard, where there is often no control over external triple-stores.

These differences makes Linked Data replication partially a conceptually different problem than database replication. Applying the state-machine approach to Linked Data requires a certain level of server access, something which is often not feasible. Additionally (as explained in section 3.1), the graph structure and inference functionality of Linked Data presents issues in *partial* replication which are not covered by current database research.

5 Research Questions

The main question of this research is **How can we achieve Linked Data replication for all possible dimensions?**.

The different dimensions shown in section4 present a large problem space with different questions for each of them. In this doctoral research, we chose to focus on the following questions: **Can we distinguish between general data replication scenarios for Linked Data, and how do these relate to the different dimensions?** This provides a specific set of requirements for the different replication scenarios, and a roadmap with which to guide this research.

How to decide which part of the data to replicate? For partial data replication, the selection of what to sync might not be obvious. Therefore, a selection of the graph needs to be made which needs to be replicated, for example using query logs, user profiles, or by rankings in the dataset.

How to efficiently use existing semantic diff algorithms in a data replication scenario? Existing research on semantic differences mostly have an analytical perspective, which might not fit the data replication requirements. Vice versa, in data replication scenarios we might make assumptions on datasets, which makes the semantic diff task easier. Something which is often not possible from an analytical point of view.

This introduces another question, namely: **How to calculate the semantic difference between a triple store and its partial replica?** It is not a trivial procedure to calculate the semantic difference between stores when on store is a subset of the other. After all, we are only interested in the difference of the *subset* of the original triple store, and the partial replica.

How to efficiently detect changes? This differs depending on the level of server access. No server access to a server means no middle-layer on the server to detect changes. What is the best way to do such change detection using for instance SPARQL?

What is the best 'unit of change'? E.g. synchronizing the update query, batches of update queries, the changed triples, a subset of the graph, or the complete graph. Which scenarios require what kind of change set?

6 Approach

For the actual synchronization of the changes between the triple store, there are several existing tools and platforms to use. In previous work we studied a

basic infrastructure for synchronization of basic RDF triples. We applied existing tools such as rsync, MySQL and GIT in the domain of the Semantic Web, and evaluated their performance using the standard SP^2 Semantic Web query testing environment. Besides these approaches there are other (e.g. Microsoft Sync Framework or the OpenSync) tools and platforms we can use for the actual distribution of data.

We will carry out our research in three phases. The first phase consists of making an overview of the general Linked Data replication scenarios, and their dimensions and requirements.

In the second phase, we will (starting with the scenario estimated as least challenging) use current database and Linked Data techniques to develop a method for data replication. If these techniques are insufficient in solving the problem of data replication, then the research will aim to develop techniques which do support data replication for this scenario.

In the third phase we will evaluate the Linked Data replication method created in phase 2. For experiment validity, all the servers are implemented using a virtual machine (VirtualBox) with the same hardware specifications. We will use a dataset generated by SP^{2}Bench[17], a data generator for creating arbitrarily large DBLP^2-like datasets. We measure the performance by the bandwidth usage in the network of nodes, and the replication latency (i.e. the time it takes for both triple-store to be consistent).

7 Conclusion

We showed the importance and different scenarios of Linked Data replication. There is no other research with a comprehensive focus on data replication for Linked Data. However, as shown in section 3, there is related work on which we can build this research. We believe our previous work on synchronization infrastructures for Linked Data, and the related work, provides a solid base to build this research on.

References

1. Alonso, G.: Partial database replication and group communication primitives (Extended Abstract). In: Advances in Distributed Systems, pp. 1–6 (1997)
2. Amir, Y., Tutu, C.: From total order to database replication. In: Distributed Computing Systems, pp. 494–503. IEEE Comput. Soc. (2002)
3. Barbieri, D.F., Braga, D., Ceri, S., Della Valle, E., Grossniklaus, M.: Incremental Reasoning on Streams and Rich Background Knowledge. In: Aroyo, L., Antoniou, G., Hyvönen, E., ten Teije, A., Stuckenschmidt, H., Cabral, L., Tudorache, T. (eds.) ESWC 2010, Part I. LNCS, vol. 6088, pp. 1–15. Springer, Heidelberg (2010)
4. Berners-lee, T., Connolly, D.: D delta: an ontology for the distribution of differences between rdf graphs. In: WWW (2004), `http://www.w3.org/DesignIssues/Diff`

[2] `http://dblp.uni-trier.de/`

5. Cecchet, E., Candea, G.: Middleware-based database replication: the gaps between theory and practice. In: ACM SIGMOD: Management of Data, pp. 739–752 (2008)
6. Franconi, E., Meyer, T.: Semantic diff as the basis for knowledge base versioning. In: Proc. of the 13th International Workshop on Non-Monotonic Reasoning (2010)
7. Goncalves, R., Parsia, B.: Analysing the evolution of the NCI Thesaurus. In: 24th IEEE International Symposium on Computer-Based Medical Systems (2011)
8. Groza, T.: Semantic Versioning Manager: Integrating SemVersion in Protégé. In: Proceedings of the 9th International Protege Conference, pp. 1–3 (2006)
9. Guéret, C., Schlobach, S.: SemanticXO: Connecting the XO with the World's Largest Information Network. In: Yonazi, J.J., Sedoyeka, E., Ariwa, E., El-Qawasmeh, E. (eds.) ICeND 2011. CCIS, vol. 171, pp. 261–275. Springer, Heidelberg (2011)
10. Holliday, J., et al.: Partial database replication using epidemic communication. In: International Conference on Distributed Computing Systems, pp. 485–493 (2002)
11. Klein, M.: Ontology versioning on the Semantic Web. Stanford University, pp. 75–91 (2001)
12. Oster, G., et al.: Data consistency for P2P collaborative editing. In: 20th Anniversary Conference on Computer Supported Cooperative Work, pp. 259–268 (2006)
13. Palma, R., et al.: Change Representation For OWL 2 Ontologies. In: 6th International Workshop on OWL: Experiences and Directions, pp. 1–10 (2009)
14. Passant, A.: sparqlPuSH: Proactive notification of data updates in RDF stores using PubSubHubbub. In: Scripting for the Semantic Web, pp. 1–10 (2010)
15. Rietveld, L., Schlobach, S.: Semantic Web in a Constrained Environment. In: Downscaling the Semantic Web Workshop, ESWC (2012)
16. Schandl, B.: Replication and Versioning of Partial RDF Graphs. In: Aroyo, L., Antoniou, G., Hyvönen, E., ten Teije, A., Stuckenschmidt, H., Cabral, L., Tudorache, T. (eds.) ESWC 2010, Part I. LNCS, vol. 6088, pp. 31–45. Springer, Heidelberg (2010)
17. Schmidt, M., Hornung, T., Lausen, G., Pinkel, C.: SP2Bench: A SPARQL Performance Benchmark. In: Data Engineering 2009, pp. 222–233 (2009)
18. Schneider, F.B.: Implementing Fault-Tolerant Approach: A Tutorial Services Using the State Machine. ACM Computing Surveys (CSUR) 22(4), 299–319 (1990)
19. Siegel, D.: Pull: The Power of the Semantic Web to Transform Your Business. Portfolio (2009)
20. Skaf-Molli, H., Rahhal, C., Molli, P.: Peer-to-Peer Semantic Wikis. In: Bhowmick, S.S., Küng, J., Wagner, R. (eds.) DEXA 2009. LNCS, vol. 5690, pp. 196–213. Springer, Heidelberg (2009)
21. Sousa, A., Pedone, F.: Partial replication in the database state machine. In: International Symposium on Network Computing and Applications, pp. 298–309 (2001)
22. Tummarello, G., Morbidoni, C., Bachmann-Gmür, R., Erling, O.: RDFSync: Efficient Remote Synchronization of RDF Models. In: Aberer, K., Choi, K.-S., Noy, N., Allemang, D., Lee, K.-I., Nixon, L.J.B., Golbeck, J., Mika, P., Maynard, D., Mizoguchi, R., Schreiber, G., Cudré-Mauroux, P. (eds.) ISWC/ASWC 2007. LNCS, vol. 4825, pp. 537–551. Springer, Heidelberg (2007)
23. Zander, S., Schandl, B.: Context-driven RDF data replication on mobile devices. In: 6th International Conference on Semantic Systems, vol. 3, pp. 131–155 (2011)