

Versioned Objects

Dag Hovland, Fredrik Chrislock

Bouvet Norway, Bergen, Norway

In our work at Equinor, an oil and gas operator company, we are involved in the exchange of information between contractors and operator during design and planning of new facilities, or *capital projects*. We are part of a larger effort to get away from transferring documents and rather transfer smaller pieces of information. The goal is that information about the facility becomes less disconnected and that more of it can be processed by machines. It is also a goal that the interaction between operator and contractor can be faster and more small-grained. That is, smaller pieces of information can be transferred, not only complete documents.

RDF is a strong candidate for the transfer and storage of this information. However, it does not fulfill our needs for maintaining the same dataset in different organizations. Specifically, parts of the model, in different stages of development, will be exchanged between organizations (e.g. operator and contractor) and between units, the parts will be extended, changed and commented on simultaneously in these different organizations. It is not an option to maintain a single RDF dataset that is directly edited by all partners. There are two reasons for this: i) The contractor and operator do not wish to share their complete graph with each other and ii) Multiple, differing, versions of the same model/graph must be possible to maintain, to be able to explore different design choices. A consequence of this requirement is that decorating properties with identifiers of the dataset version, like in [3], is not sufficient.

In relational data warehousing this type of data is called *slowly changing dimensions*[2] and there are 7 named patterns for maintaining them. These patterns can also be implemented in RDF, but since we are no longer restricted to fixed, tabular formats, some patterns therefore are probably not useful in RDF.

The options we have considered for transferring the model as RDF between contractor and operator

1. Send the whole RDF graph every time
2. Send instructions about what triples to delete and what to insert
3. Send the RDF graph for each *object* that has been changed

Before explaining what we mean with objects, let us explain why the first two options do not cover our needs: Transferring the whole RDF graph means that concurrent changes to the graph are complicated and would need a strong locking mechanism on the RDF graph at the operator, preventing changes to the parts of the graph that are currently under work by the contractor. This is a too strict requirement.


ISWC 2022 Industry Proceedings, October 23 – 27, 2022, Hangzhou, China

✉ dag.hovland@bouvet.no (D. Hovland); fredrik.chrislock@bouvet.no (F. Chrislock)

🆔 0000-0002-3569-8838 (D. Hovland)



© 2022 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

 CEUR Workshop Proceedings (CEUR-WS.org)

The second option is not allowed by the users: The engineers want to make sure that whoever makes a change to an object in the model, is aware of the state of the object they are changing. This is not possible in a distributed setting if single triples can be inserted or removed independently. This also means that Auer and Herre's atomic changes [1] does not solve our problem, although they could supplement the solution by being used to describe the changes on objects.

This leaves us with our suggested solution, which is to agree upon a fragmentation of the data into *objects*. Every triple in the RDF graph is a member of exactly one such object, and there must be agreed between the collaborators rules about the sizes of these objects. It must further be possible to easily distinguish between different versions of an object, and the provenance history of that specific version. Note that *a version of an object* is a RDF graph. These requirements appear intuitive to the engineers and have led us to the following design decisions:

1. Data is transferred as JSON-LD, constructed using an agreed JSON-LD-Frame. We found JSON-LD Framing convenient for expressing and agreeing upon the sizes of the objects.
2. We distinguish between a *persistent IRI*, which identifies a domain object, and a *versioned IRI*, which identifies an RDF graph describing a specific version of an object. The latter are immutable.
3. To identify and distinguish versioned IRIs and objects we use a custom scheme where the versioned IRIs consist of the persistent IRIs suffixed with the string `"/version/"`, a hash of the object, and then the date
4. Edges between objects use the versioned IRIs, and are reified, using `rdf:subject`, `rdf:predicate` and `rdf:object`
5. PROV-Os `prov:wasDerivedFrom` is used to link an object to the previous version of an object

The representation of versions of objects as separate entities with their own IRIs is necessary to be able to support the transfer of changes of data between organizations. When a system receives an object, it can compare with the previously stored version of the object, and, importantly, see both what triples should be removed and which should be added. Our approach has no impact on how data is read from the RDF Graph, only on how data is written into it.

An implementation of this versioning scheme is available at <https://github.com/equinor/versioned-object>.

References

- [1] S. Auer and H. Herre. A versioning and evolution framework for RDF knowledge bases. In I. B. Virbitskaite and A. Voronkov, editors, *Perspectives of Systems Informatics, 6th International Andrei Ershov Memorial Conference, PSI 2006, Novosibirsk, Russia, June 27-30, 2006. Revised Papers*, volume 4378 of *Lecture Notes in Computer Science*, pages 55–69. Springer, 2006.
- [2] R. Kimball. *The Data Warehouse Toolkit: Practical Techniques for Building Dimensional Data Warehouses*. John Wiley, 1996.
- [3] X. Ma, C. Ma, and C. Wang. A new structure for representing and tracking version information in a deep time knowledge graph. *Computers & Geosciences*, 145:104620, 2020.