The Provenance Store prOOst for the Open Provenance Model

Andreas Schreiber, Miriam Ney, and Heinrich Wendel

German Aerospace Center (DLR),
Linder Hoehe, 51147 Cologne, Germany,
{Andreas.Schreiber,Miriam.Ney,Heinrich.Wendel}@dlr.de,
http://www.dlr.de/sc

Abstract. This paper presents the provenance storing system *prOOst* which uses a semi-structured approach to store the provenance data based on the Open Provenance Model (OPM). It uses the graph database "Neo4j" for storage and the graph traversal language "Gremlin" for querying. Furthermore, it provides a REST interface to record data into the store, and a web front end to query the database. The prOOst provenance system was published as Open Source software and is available on SourceForge.

1 Introduction

In [1] the representation of a provenance system is described as follows: A provenance aware application sends information of interest to the provenance store. From this store inquiries and information is gathered, and possibly given back to the application.

To record the information, different approaches have been investigated. In [2] four different realisations are discussed: Relational, XML with XPath, RDF with SPARQL and semi-structured approaches. They conclude semi-structured approaches to be most promising. In semi-structured systems, the used technology has no formal structure, but it provides means of being queried.

In this work, we present the provenance storing system *prOOst* which uses a semi-structured approach to store the provenance data based on the Open Provenance Model (OPM).

2 The Provenance Store prOOst

The Provenance Store prOOst uses the graph database "Neo4j" [3] for storage and the graph traversal language "Gremlin" [4] for querying. Furthermore, it provides a REST interface to record data into the store, and a web front end to query the database. The prOOst provenance system was published as Open Source software and is available on SourceForge[1].

[1] http://sourceforge.net/projects/proost/

P. Groth and J. Frew (Eds.): IPAW 2012, LNCS 7525, pp. 240–242, 2012.

It is not the first implementation using a graph database for storage technology. In [5] this approach was already successfully tested. Neo4j was chosen as it is a robust, performant and popular choice for graph storage systems. Additionally it readily connectible with the suitable Gremlin query system to meet our requirements. Further information on the implementation of OPM model provenance assertions using these systems are described in the following two sections.

2.1 Graph Database: Neo4j

> "Neo4j is a graph database, a fully transactional database that stores data structured as graphs." (cf. [3])

An advantage of graph databases like Neo4j is that they offer very flexible storage models, allowing for a rapid development. Neo4j is dually licensed (AGPLv3 open source and commercial).

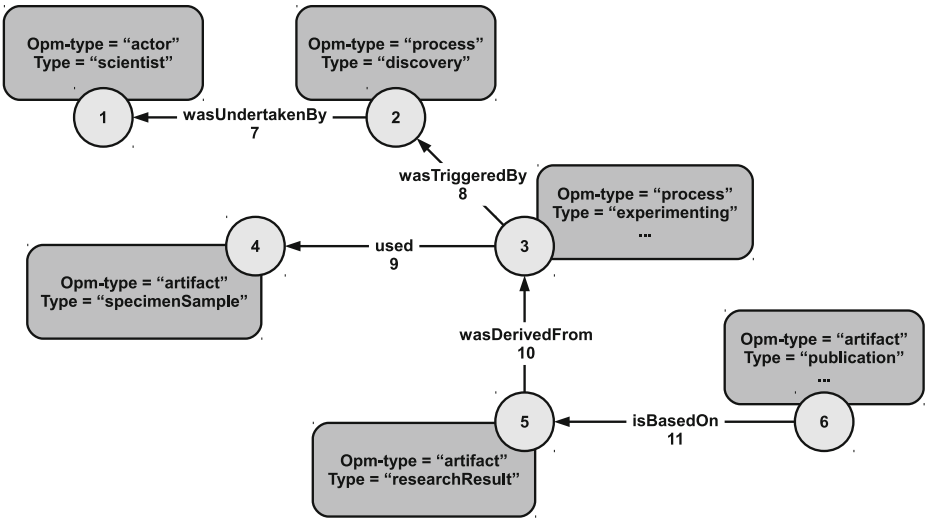

Fig. 1. OPM example in Neo4j

Modelling OPM using Neo4j is described in more detail in [6]. Fig. 1 shows an example of an OPM graph. Each element is represented by a node (vertex) in the database. Nodes are indexed according to the Neo4j standard. The nodes can be annotated with further (OPM specific) information, such as "process" or "artifact". Analogously, also the edges connecting the nodes are indexed and annotated with a label (the OPM relationship).

2.2 Query Language: Gremlin

"Gremlin is a graph traversal language" [4]. Gremlin already provides an interface to interact with the Neo4j graph database. The following example shows its use for querying Neo4j on the example database, searching for the names (identifiers) of all discoveries of a certain `scientistX`:

```
$_g := neo4j:open('database')
$scientists := g:key($_g, 'type', 'scientist')
$scientistX := g:key($scientists, 'identifier', 'scientistX')
$discoveries := $scientistX/inE/inV[@identifier']
```

3 Conclusions and Future Work

Neo4j was chosen as it is a robust, performant and popular choice for graph storage systems. Additionally it readily connectible with the suitable Gremlin query system to easily perform queries. The provenance store prOOst was evaluated and used by two different applications: Recording the provenance of software development processes [6] and recording provenance in an electronic laboratory notebook system [7].

Future work will concentrate on moving prOOst to support the *PROV Data Model* [8] that is specified by the W3C Provenance Working Group.

References

1. Moreau, L., Clifford, B., Freire, J., Futrelle, J., Gil, Y., Groth, P., Kwasnikowska, N., Miles, S., Missier, P., Myers, J., Plale, B., Simmhan, Y., Stephan, E., den Bussche, J.V.: The open provenance model core specification (v1.1). Future Generation Computer Systems (July 2010)
2. Holl, D.A., Braun, U., Maclean, D., Kumar Muniswamy-Reddy, K., Seltzer, M.I.: Choosing a data model and query language for provenance. In: Proceedings of the 2nd International Provenance and Annotation Workshop (2008)
3. Neo4j.org: Neo4j graph database, `http://neo4j.org`
4. Rodriguez, M.A.: Gremlin graph traversal language, `http://gremlin.tinkerpop.com`
5. Tylissanakis, G., Cotronis, Y.: Data provenance and reproducibility in grid based scientific workflows. In: Workshops at the Grid and Pervasive Computing Conference, pp. 42–49 (2009)
6. Wendel, H., Kunde, M., Schreiber, A.: Provenance of Software Development Processes. In: McGuinness, D.L., Michaelis, J.R., Moreau, L. (eds.) IPAW 2010. LNCS, vol. 6378, pp. 59–63. Springer, Heidelberg (2010)
7. Ney, M.: Enabling a data management system to support the good laboratory practice. Master's thesis, Freie Universität Berlin (2011), `http://elib.dlr.de/75261/`
8. Moreau, L., Missier, P., Belhajjame, K., Cresswell, S., Gil, Y., B'Far, R., Groth, P., Klyne, G., McCusker, J., Miles, S., Myers, J., Sahoo, S.: PROV-DM Part 1: The Provenance Data Model, `http://dvcs.w3.org/hg/prov/raw-file/default/model/prov-dm.html`