# Will Graph Data Management Techniques Contribute to the Successful Large-Scale Deployment of Semantic Web Technologies?

Philippe Cudre-Mauroux

*eXascale Infolab*
*University of Fribourg—Switzlerand*
pcm@unifr.ch

***Abstract*— Most certainly.**

## I. Introduction

The Semantic Web vision ambitions to create a large-scale, interconnected Web of machine-processable data in addition to the current Web of documents currently available online. Its most recent and visible incarnation—the Linked Data movement[1]—is based on three main principles:

- using dereferenceable Web identifiers (HTTP URIs) to uniquely identify the various pieces of data online
- providing semi-structured information—most often using standard formats like RDF [1] and OWL [2]—about the data when dereferencing the corresponding URIs
- providing links to other, semantically related pieces of data.

The result of this grassroots effort is the emergence of a giant, decentralized and interlinked online database. The wide adoption of Linked Data principles opens the door to many exciting applications, including expressive and structured Web search, automated processing of online data, and Web-scale data integration.

While the Web of data is currently capturing a lot of attention because of its potential, it is also being threatened by a series of technical challenges. At this stage, the research community is actively working on novel solutions to store, manage and integrate massive Linked Data sets across the Web. In this short paper, I argue that graph data management techniques will play an essential role in the successful deployment of Semantic Web technologies. I start below by giving a state of the art in Semantic Data Management, before describing a few areas where graph data management is already having an important impact in this context.

## II. State of the Art in Semantic Data Management

Semantic Web Data Management is an interdisciplinary field or research at the intersection of several disciplines, including database systems, artificial intelligence, and Web engineering. A number of key advances have been made in this field over the past ten years, most notably on efficient and scalable RDF/OWL back-end, scalable inference techniques, query languages, and more recently on federated query processing.

### A. Data Back-End

The data model behind the Semantic Web and Linked Data movement is at the same time conceptually simple and technically challenging to capture. RDF data is constituted of bags of $(subject,\ predicate,\ object)$ ***triples***, where the subject and predicate are typically URIs, and the object is either a URI or a literal value. A triple storing the name of an identifier representing a person would for instance look as follows:

( *http://semanticweb.org/pcudremauroux*,
*http://xmlns.com/foaf/name*,
*"Philippe Cudre-Mauroux"* )

RDF data can as such be considered as collections of ternary structures. Many approaches use this perspective to store RDF data in giant *triple tables* (e.g., in relational tables having three attributes, one for the *subjects*, one for the *predicates*, and finally one storing the *objects*). The GridVine [3], [4] system, for instance, uses a triple-table storage approach to distribute RDF data over decentralized P2P networks using a distributed hash-table. More recently, Hexastore [5] suggests to index RDF data using six possible indices, one for each permutation of the set of columns in the triple table. Similarly, RDF-3X [6] creates various indices from a giant triple-table, including indices based on the six possible permutations of the triple columns, and aggregate indices storing only two out of the three columns. 4Store[2] and AllegroGraph[3] are two further examples of database systems following this philosophy.

A second important approach clusters RDF data in various tables based on their properties (i.e., based on the *predicates* of the triples). Wilkinson *et al.* [7] propose the use of two types of property tables: one containing clusters of values for properties that are often co-accessed together, and one exploiting the type property of subjects to cluster similar sets of subjects together in the same table. Chong *et al.* [8] also suggest the use of property tables as materialized views, complementing

[1] http://linkeddata.org/

[2] http://4store.org/

[3] http://www.franz.com/agraph/allegrograph/

a primary storage using a triple-table. Going one step further, Abadi *et al.* suggest a fully-decomposed storage model for RDF: the triples are in that case rewritten into $n$ two-column tables where $n$ is the number of unique properties in the data. In each table, the first column contains the subjects that define that property and the second column contains the object values for those subjects.

### B. Query Processing

Very early on in the development of the Semantic Web, the World Wide Web Consortium (W3C)[4] standardized a series of languages to express and query Semantic Web data. The predominant query language in that space is SPARQL [9], which is a declarative query language *à la* SQL to query (and update) RDF data. The basic query construct behind SPARQL is the *triple pattern*, which looks like an RDF triple except that any of its subparts can be replaced by a variable. A simple SPARQL query retrieving the name associated to a given resource would look like this:

```
SELECT ?x
FROM example.rdf
WHERE {
    http://semanticweb.org/philippecudremauroux
    http://xmlns.com/foaf/name
    ?x .
}
```

A large fraction of Semantic Web queries can be expressed by writing conjunctions of such triple patterns. The queries are then typically answered by iteratively looking up bound variables in the patterns, and joining up intermediate results.

In addition to local queries, distributed queries involving series of autonomous SPARQL end-points (i.e., *federated* queries) are rapidly gaining in popularity. This is a natural consequence of the emergence of the Web of Linked Data, since local data systematically include links to distant but related data. A growing number of approaches exist to optimize such operations (see for instance [10] for a recent survey and [11] for a state of the art approach).

## III. Graph Data Management and the Semantic Web

While RDF data can be seen as a collection of triples, researchers realized very early that it can also be modeled as a graph (see for example Angles & Gutierrez [12] for an early paper on that topic). Figure 1, for instance, gives a graph representation of two RDF triples (the figure adopts the usual convention to represent such a graph, where URIs are represented by ovals, literals by rectangles, and predicates are represented by directed edges connecting subjects to objects).

Adopting this formalism, the Linked Data movement can be represented as a gigantic graph of interconnected RDF data sets. Figure 2shows a macroscopic representation of this graph,

[4]http://www.w3.org/

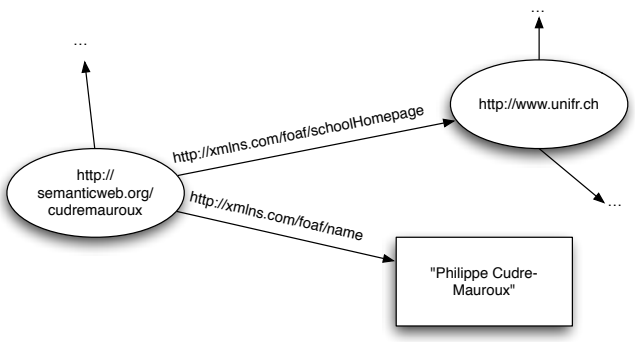


Fig. 1. A small RDF graph representing two triples

where each edge represents an entire RDF data set, and edges represent collections of links between pairs of data sets[5] .

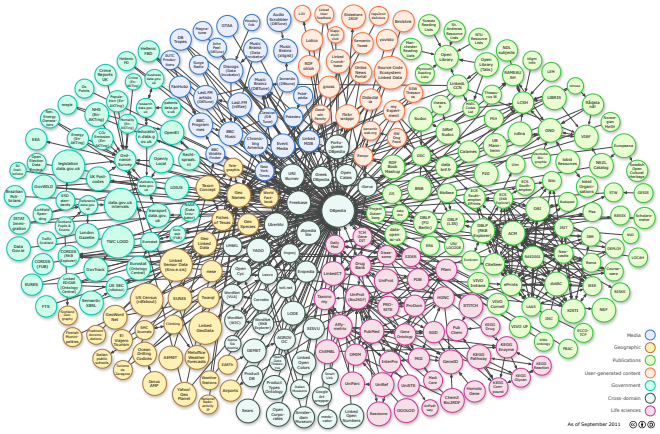

Fig. 2. A macroscopic representation of the Linked Data graph

### A. Data Back-End

Though often depicted as a graph, there has been surprisingly little research and development on graph data infrastructures for the Semantic Web. Oracle led one of the most visible efforts in that space, offering RDF support for its 11g database product[6], which stores RDF data using two main relational tables, one for the edges (predicates), and another one storing data about the nodes (subjects, objects).

Recently, some further approaches took advantage of subgraph storage to efficiently and compactly store RDF data. My own experience with dipLODocus$_{[RDF]}$ [13], a hybrid back-end [14] for RDF, shows that collocation of data values based on the proper identification of RDF subgraphs can speed up query processing by one order of magnitude. Along conceptually similar lines, Yan *et al.* [15] suggest to divide the RDF graph into subgraphs and to build secondary indices (e.g., Bloom filters) to quickly detect whether some information can be found inside an RDF subgraph or not. gStore [16]

[5]Linking Open Data cloud diagram, by Richard Cyganiak and Anja Jentzsch. http://lod-cloud.net/

[6]http://www.oracle.com/technetwork/database/options/semantic-tech/index.html

is a recent system storing RDF data as a large, labeled, and directed multi-edge graph; SPARQL queries are then executed by being transformed into subgraph matching queries, that are efficiently matched to the graph using a novel indexing mechanism.

### B. Query Processing

Beyond triple pattern queries—which can be efficiently processed as subgraph matching queries on graph back-ends (see above)—a number of further SPARQL queries can take advantage of graph data management techniques. Path queries, for instance, are getting extremely popular. They come in different flavors, but typically are defined as lists of predicates (e.g., *(address, town, zip code)* for a path query going from a node having and address to a zip code literal) defining series of edges to follow in the RDF graph. Such queries are extremely costly to resolve using triple table or property table approaches, since they require a (self) join for each new property defined in the query. Graph or subgraph systems can often resolve path queries more efficiently by limiting the number of joins (e.g., by joining subgraphs [13]).

Beyond the classical path queries mentioned above, SPARQL is currently being enhanced with additional features to express more complex path queries, transitive closures and reachability queries (see the SPARQL 1.1. W3C Working Draft [17]). The following SPARQL query (taken from the W3C draft), for example, expresses a reachability query as two triple patterns and uses a new syntactic construct (+) to express a transitive closure on the *foaf:knows* transitive property starting from a given node (Alice):

```
{ ?x foaf:mbox mailto:alice@example .
  ?x foaf:knows+ / foaf:name ?name .}
```

Reachability queries for RDF have not been widely studied so far but are currently getting a lot of attention because of those recent developments. Shortest-path queries are not yet supported by SPARQL, but are already receiving some attention [18] and are supported by a few systems (like Virtuoso[7]).

In addition to local graph query processing, an increasing number of researchers are starting to adapt well-known graph techniques to optimize RDF federated queries on wide-area networks. Recent efforts [19], [20], for example, apply well-known graph algorithms (Edmonds' algorithms or Minimum-Spanning-Tree algorithms) to optimize distributed SPARQL queries.

## IV. Conclusions

With the rapid adoption of Linked Data principles, a giant graph of structured data is emerging from the first time on the Web. The language used to express such data (RDF) can itself be considered as a graph data language. In this short paper, I gave a series of concrete examples showing how graph data management techniques are gaining increasing attention in that context. Additional examples can be found in a recent tutorial comparing both social networks and the Web of data from a graph data management perspective [21].

[7] http://virtuoso.openlinksw.com/

## V. Acknowledgment

This work is supported by the Swiss National Science Foundation under grant number PP00P2_128459.

## References

[1] F. Manola and E. Miller (Ed.), "RDF Primer," W3C Recommendation, February 2004, http://www.w3.org/TR/rdf-primer/.

[2] D. McGuinness and F. van Harmelen (Ed.), "OWL Web Ontology Language Overview," W3C Recommendation, February 2004, http://www.w3.org/TR/owl-features/.

[3] K. Aberer, P. Cudre-Mauroux, M. Hauswirth, and T. van Pelt, "GridVine: Building Internet-Scale Semantic Overlay Networks," in *International Semantic Web Conference (ISWC)*, 2004.

[4] P. Cudré-Mauroux, S. Agarwal, and K. Aberer, "Gridvine: An infrastructure for peer information management," *IEEE Internet Computing*, vol. 11, no. 5, 2007.

[5] C. Weiss, P. Karras, and A. Bernstein, "Hexastore: sextuple indexing for semantic web data management," *Proceeding of the VLDB Endowment (PVLDB)*, vol. 1, no. 1, pp. 1008–1019, 2008.

[6] T. Neumann and G. Weikum, "RDF-3X: a RISC-style engine for RDF," *Proceedings of the VLDB Endowment (PVLDB)*, vol. 1, no. 1, pp. 647–659, 2008.

[7] K. Wilkinson, C. Sayers, H. A. Kuno, and D. Reynolds, "Efficient rdf storage and retrieval in jena2," in *SWDB'03*, 2003, pp. 131–150.

[8] E. I. Chong, S. Das, G. Eadon, and J. Srinivasan, "An efficient sql-based rdf querying scheme," in *International Conference on Very Large Data Bases (VLDB)*, 2005.

[9] E. Prud'hommeaux and A. Seaborne, "SPARQL Query Language for RDF," W3C Recommendation, January 2008, http://www.w3.org/TR/rdf-sparql-query/.

[10] O. Grlitz and S. Staab, *Federated Data Management and Query Optimization for Linked Open Data*, ser. New Directions in Web Data Management. Springer, 2011, vol. 1.

[11] A. Schwarte, P. Haase, K. Hose, R. Schenkel, and M. Schmidt, "Fedx: optimization techniques for federated query processing on linked data," in *International Semantic Web Conference (ISWC)*. Berlin, Heidelberg: Springer-Verlag, 2011, pp. 601–616.

[12] R. Angles and C. Gutierrez, "Querying rdf data from a graph database perspective," in *In Proceedings of the Second European Semantic Web Conference*, 2005, pp. 346–360.

[13] M. Wylot, J. Pont, M. Wisniewski, and P. Cudré-Mauroux, "diplodocus[rdf] - short and long-tail rdf analytics for massive webs of data," in *International Semantic Web Conference (ISWC)*, 2011, pp. 778–793.

[14] M. Grund, J. Krüger, H. Plattner, A. Zeier, P. Cudré-Mauroux, and S. Madden, "Hyrise - a main memory hybrid storage engine," *PVLDB*, vol. 4, no. 2, pp. 105–116, 2010.

[15] Y. Yan, C. Wang, A. Zhou, W. Qian, L. Ma, and Y. Pan, "Efficient indices using graph partitioning in rdf triple stores," in *International Conference on Data Engineering (ICDE)*, 2009, pp. 1263–1266.

[16] L. Zou, J. Mo, L. Chen, M. T. Oezsu, and D. Zhao, "gstore: Answering sparql queries via subgraph matching," *PVLDB*, vol. 4, no. 8, 2011.

[17] S. Harris and A. Seaborne, "RSPARQL 1.1 Query Language," W3C Working Draft, May 2011, http://www.w3.org/TR/sparql11-query/.

[18] A. Gubichev and T. Neumann, "Path query processing on very large rdf graphs," in *WebDB*, 2011.

[19] B. P. Vandervalk, E. L. McCarthy, and M. D. Wilkinson, "Optimization of distributed sparql queries using edmonds' algorithm and prim's algorithm," in *International Conference on Computational Science and Engineering (CSE)*, 2009, pp. 330–337.

[20] X. Wang, T. Tiropanis, and H. Davis, "Evaluating graph traversal algorithms for distributed sparql query optimization," in *Joint International Semantic Technology Conference (JIST2011)*, 2011.

[21] P. Cudré-Mauroux and S. Elnikety, "Graph Data Management Systems for New Application Domains," in *International Conference on Very Large Data Bases (VLDB)*, 2011.