RDF Query Languages Need Support for Graph Properties

Renzo Angles[1], Claudio Gutierrez[1], and Jonathan Hayes[1,2]

[1] Dept. of Computer Science, Universidad de Chile
[2] Dept. of Computer Science, Technische Universität Darmstadt, Germany
{rangles,cgutierr,jhayes}@dcc.uchile.cl

Abstract. This short paper discusses the need to include into RDF query languages the ability to directly query graph properties from RDF data. We study the support that current RDF query languages give to these features, to conclude that they are currently not supported. We propose a set of basic graph properties that should be added to RDF query languages and provide evidence for this view.

1 Introduction

One of the main features of the Resource Description Framework (RDF) is its ability to interconnect information resources, resulting in a graph-like structure for which *connectivity* is a central notion [GLMB98]. As we will argue, basic concepts of graph theory such as *degree*, *path*, and *diameter* play an important role for applications that involve RDF querying. Considering the fact that the data model influences the set of operations that should be provided by a query language [HBEV04], it follows the need for graph operations support in RDF query languages. For example, the query "all relatives of degree 1 of Alice", submitted to a genealogy database, amounts to retrieving the nodes adjacent to a resource. The query "are suspects A and B related?", submitted to a police database, asks for any path connecting these resources in the (RDF) graph that is stored in this database. The query "what is the Erdős number of Alberto Mendelzon", submitted to (a RDF version of) DBLP, asks simply for the length of the shortest path between the nodes representing Erdős and Mendelzon. There are manifold examples like this. Surprisingly, current RDF languages mostly do not support this type of queries. A language is said to *support* a feature if it provides facilities that make it convenient (reasonable easy, safe and efficient) to use that feature [Str88].

In this short paper we show that all RDF query languages considered exploit the underlying graph structure of RDF only to a limited extent. Using the documentation and implementations available, we studied seven of the most representative RDF query languages with respect to this point. We then propose a set of basic features that RDF query languages should support to take advantage of the graph-like nature of RDF data and to offer richer querying.

Related Work. There is a large amount of literature on topics related to graph querying. Nevertheless, to the best of our knowledge, incorporating any of these approaches into RDF query languages has not been addressed in a systematic way[1]. There are several web sites[2] and comparative studies [MKA+02,Pér02,Mil03] of RDF query languages, considering features such as their expressivity, robustness, and syntax. The most recent study [HBEV04] devotes a small section to graph properties of these languages, but considers only path expressions of fixed length. A language not considered there is RxPath[3], which goes further in addressing graph features. As the case of other query languages developed for the tree-like XML model, it works well for queries to retrieve pieces of tree paths, but does not address basic graph notions as mentioned above. Finally, queries addressing connectivity between resources in relation to semantic associations are studied in [AS03].

The database community has addressed the issue of querying graph models. In Güting [Güt94] a proposal of modeling graph databases with objects is proposed, including a query language with graph-like features. The drawback of this proposal in our context is that it is strongly tied to the underlying object model. There has also been graph-querying work in the context of the hypertext model, semi-structured data, Web, and XML, see [FLM98,DFF+99]. With respect to algorithmic aspects of querying and indexing graphs, there is a recent survey of the database aspects of tree and graph searching [SWG02].

2 Queries Involving Graph Notions

For classical graphs, we will follow standard notations and concepts, as for example in Diestel [Die97]. For RDF graphs, it is important to recall that they are defined as sets of triples, thus, they are not properly graphs in the classical sense. The representation used by default (see, e.g., [MM04]) represents each triple (a, p, b) as a directed labeled graph $a \xrightarrow{p} b$. (This representation does not put into account that statement properties may also occur as the subjects or objects of other statements [HG04]).

Motivating Examples We present in this section a more systematic set of query examples. Although there are still not very many repositories of RDF data (cf. the survey [Ebe02] for availability and findability) it is possible to outline applications over real-life data for which graph querying is relevant.

EX 1. (*Biology*) Excellent examples for the need for complex queries in large datasets come from biology. These include bio-pathways data, protein interaction networks, taxonomies and phylogenetic trees, chemical structure graphs, food

[1] For a discussion of this topic see, e.g., `http://lists.w3.org/Archives/Public/www-rdf-rules/2003Sep/0001.html`

[2] `http://rdfstore.sourceforge.net/2002/06/24/rdf-query/query-use-cases.html`, `http://www.w3.org/2001/11/13-RDF-Query-Rules/`

[3] `http://rx4rdf.liminalzone.org/RxPath`

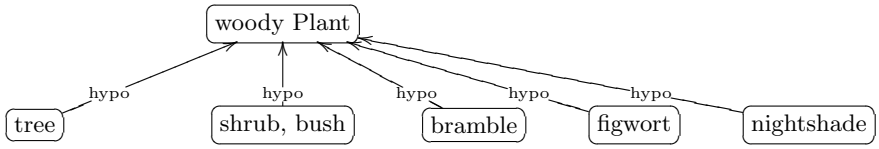

Fig. 1: Coordinate terms of "tree" in WordNet

webs, laboratory protocols, genetic maps, and multiple sequence alignments. Queries in such a data context often include various types of path queries where regular expressions, shortest paths, and matching of subgraphs play a central role [Olk03].

EX 2. The *Photo Metadata Co-Depiction Experiment*[4] provides an interface to explore relations between people depicted on photographies. Two people are co-depicted if there exists some digital image that depicts them both. Querying for two persons will return a path of photos, linked by co-depiction: for example, querying for Tim Bray and John F. Kennedy will return a path of three photos, depicting (1) Tim Bray and Tim Berners-Lee, (2) Tim Berners-Lee and Bill Clinton, and (3) Bill Clinton and John F. Kennedy.

EX 3. (*A Metadata Repository of Bibliographical Information*) A RDF knowledge base containing bibliographic information on scientific publications (such as Citeseer and DBLP[5]) would give rise to a number of interesting queries, such as: "what is the relation between scientists A and B?" This amounts to the computation of paths between resources of a RDF graph, possibly restricted to properties (edges) such as *cites* and *isCoauthor*. A query like "What is the *influence* of article D?" requires the computation of the transitive closure of the *isReferencedBy* (the inverse of *cites*) relation from the root node D.

EX 4. *WordNet*[6] is an online lexical reference system. Words are organized into synonym sets, which are ordered by the hyponym (roughly: sub-concept) relation. One common use of such a data is to find meaning-related clusters of words, such as the query for the *coordinate terms* for a word (i.e., the "sister words", all immediate sub-concepts of the super-concept of a word)—see figure 1. This actually corresponds to a path (`tree`, `hyponymOf`, `X`), (`Y`, `hyponymOf`, `X`), what could be easily done in, e.g., SeRQL [BKvH02]. However, for the arbitrary length of such a pattern (k hyponyms up, k down again) it or any other current RDF query language is not sufficient. Similarly, a simple query such as "are the terms *professor* and *master* related?" could not be answered.

[4] `http://www.rdfweb.org/2002/01/photo/`
[5] `http://citeseer.ist.psu.edu` and `http://dblp.uni-trier.de/`
[6] See `http://www.cogsci.princeton.edu/~wn/` for the WordNet project and `http://www.semanticweb.org/library/` for a RDF representation of it

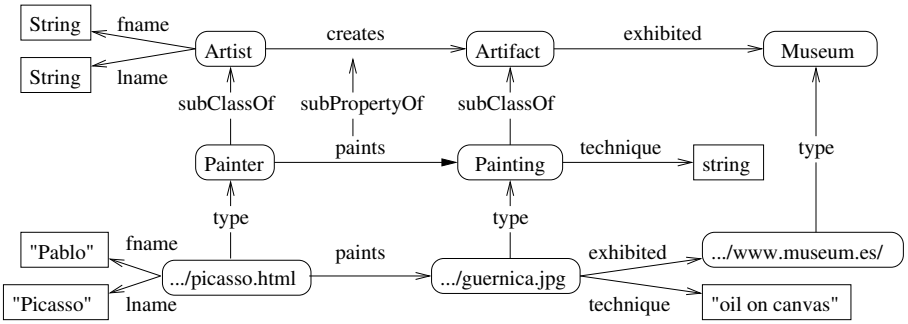

Fig. 2: The subset of the Museum database [FOR03] used for our study

3 Graph Properties in Current RDF Query Languages

We chose seven query languages, six of them already considered in the recent survey [HBEV04], and RxPath. As RDF dataset to query against we chose a subset of the well-known Museum example (see Figure 2), which—despite its small size—proved well able to illustrate our needs. To simplify, we did not consider the issue of support for subclass or subproperty semantics which is orthogonal to this discussion.

Our comparison focuses on the respective language's support for the underlying graph model of RDF, as can be seen in the following list of query examples. Table 1 gives the results of our study.

1. *Adjacent nodes:* "All resources adjacent to the resource `Guernica`". Expected result: `Painting`, `www.museum.es`, `"oil on canvas"`, `picasso.html`. Not all languages support this feature. The problem is that this query can only be expressed as a union of two queries: one for outgoing edges from `Guernica`, another for ingoing edges. Some languages do not support the union operator.
2. *Adjacent edges:* "All predicates of statements involving `Guernica`". Expected result: `technique`, `exhibited`, `type`, `paints`. The problems faced are similar to the previous case. Note that here we probably would like to differentiate schema predicates from data predicates.
3. *Degree of a node:* "Number of predicates involving `Guernica`". Expected result: 4. Same problems as above plus the fact that most languages do not support aggregation at this level. SeRQL for example returns the number, but not as part of the answer.
4. *Path:* "Find paths between `picasso.html` and `www.museum.es`". Expected results: There are several, for example, `picasso.html`–`paints`–`guernica.jpg`–`exhibited`–`www.museum.es`, and `picasso.html`–`type`–`Painter`–`paints`–`Painting`–`subClassOf`–`Artifact`–`exhibited`–`Museum`–`type`–`www.museum.es`.
   None of the languages studied support arbitrary paths like the ones needed for this case. Note that it must be considered whether paths via the schema are regarded as relevant.

5. *Fixed-length paths.* "Find all paths of length 2 beetween ‘‘`Pablo`’’ and `guernica.jpg`." Expected result: `"Pablo"`–`fname`–`picasso.html`–`paints`–`guernica.jpg`. Supported partially by several languages, using a union of all posibles patterns of paths (combination of edge directions) of length 2 between the initial and final resources. In the general case this requires the evaluation of 2^n subqueries for a path of length n.
6. *Distance between two resources:* (length of shortest path) "How far is `picasso.html` from `www.museum.es`?" Expected result: 2. Not supported by any language.
7. *Diameter of a graph:* "Diameter of the museum graph". Expected result: 5. Not supported. It is based on distance and paths.

PROPERTY	RQL	SeRQL	RDQL	Triple	N3	Versa	RxPath
Adjacent nodes	+	+	±	±	+	+	−
Adjacent edges	+	+	±	±	+	−	−
Degree of a node	−	−	−	−	−	−	−
Path	−	−	−	−	−	−	−
Fixed-length Path	±	±	±	±	±	−	−
Distance between two nodes	−	−	−	−	−	−	−
Diameter	−	−	−	−	−	−	−

Table 1: Comparison of RDF query language support for graph properties (+, ± and − indicate support, partial support and no support)

Summary. A triple (a, p, b) is represented as $a \xrightarrow{p} b$, which gives a directed graph representation for RDF data. This direction produces problems when retrieving neighborhoods for languages that do not have a union operator. Some query results violate the query language property of closure [HBEV04] by returning results which are not in RDF format. There are two main problems concerning paths: (a) most languages support only querying for patterns of paths which are limited in length and form (the issue of edge direction blows up the size of the query exponentially, see below); (b) RxPath is able to retrieve only paths starting from a fixed node and with some other restrictions. Aggregated functions like COUNT, MIN, MAX applied to paths could be used to answer queries as for the degree of a node, the distance between nodes, and the diameter of a graph. None of these functions is systematically supported, even though, for example, the original version of RQL has a COUNT function on the number of triples.

4 Conclusions

Essential functionality to be supported. We propose the following set of graph-theoretical notions to be supported by RDF query languages. They reflect in a fair manner the problems and are a wish list of graph features needed for querying RDF data.

1. *Adjacency.* Both node and edge adjacency are important in various contexts. Due to the order of the values of a RDF triple, this simple property is currently not supported by several languages.
   A more advanced notion of adjacency, like *the k-neighborhood of a node*, is necessary in several contexts. The need of 1-neighborhood retrieval in a RDF Graph is argued in [Say04] and [GMM03].
2. *Paths and Connectedness.* Arguments supporting the necessity of paths are given in [GLMB98]; also in the context of analyzing networks in [HAMAS04]. There are several variations involved in this notion, like the *length* of a path and the need of a *restricted version* to avoid meaningless paths through the schema, such as the linkage of any two resources `a`, `b` by triples (`a`, `rdf:type`, `rdfs:Resource`), (`b`, `rdf:type`, `rdfs:Resource`). *Parameters* include: labels of a certain grammar, whether transitivity is desired, and specifying a fixed set of nodes that should be included in the path.
3. A form of *pattern matching* should be supported. Most current RDF query languages do support this feature in the form of conjunction of triples. The extension to matching simple graph grammar expressions deserves to be studied.
4. *Aggregate functions.* Apart of the natural COUNT on triples and/or nodes retrieved by the query, aggregate functions dealing directly with the structure of the underlying graph, such as the degree of a node, the diameter of the graph (or a set of nodes), the distance between nodes, etc. would be useful before submitting more expensive queries.

Implementation issues require a thorough study (consider e.g. [SWG02]). However, efficiency problems do not seem to be the only reason why these features are not yet available. For example, adjacency and degree could be easily precomputed, and a COUNT value on nodes and/or triples is currently retrieved, but not in the form of a RDF assertion. Complexity is indeed an issue for the support of path querying in the RDF graph, when the edge directions shall not be considered. We showed above that the number of subqueries involved are exponential to the path length.

Contribution. We proposed a basic set of functionalities that should be supported, directly or indirectly, in order to give sufficient *expressiveness* to RDF query languages to take fully advantage of the underlying data model. From our study it follows that current RDF query languages are not *adequate*, as they do not use central concepts of RDF's graph model. Some features, like the aggregation function COUNT, which yields the number of triples returned (or the length of the list in Versa), is currently given, for example, in the interface of SeRQL, but not returned as a RDF assertion—RDF query languages should be *closed*. Finally, of the operations on graphs we proposed, asking for paths between two resources is the only one that, if considered with no restriction, is not *safe* (due to possible cycles). This is an important consideration when implementing it.

From this brief study it follows the need to research and experiment on architectures for RDF query languages to include graph features.

References

[AS03] Kemafor Anyanwu and Amit Sheth. ρ-Queries: Enabling Querying for Semantic Associations on the Semantic Web. In *Proceedings of the Twelfth International World Wide Web Conference*, pages 690–699. ACM Press, 2003.

[BKvH02] Jeen Broekstra, Arjohn Kampman, and Frank van Harmelen. Sesame: A Generic Architecture for Storing and Querying RDF and RDF Schema. In I. Horrocks and J. A. Hendler, editors, *The Semantic Web - ISWC 2002, Proceedings*. Springer, 2002.

[DFF+99] A. Deutsch, M. Fernandez, D. Florescu, A. Levy, D. Maier, and D. Suciu. Querying XML Data. *IEEE Data Engineering Bulletin*, 22(3):10–18, 1999.

[Die97] Reinhard Diestel. *Graph Theory*. Number 173 in Graduate Texts in Mathematics. Springer-Verlag New York, Inc, 1997.

[Ebe02] Andreas Eberhart. Survey of RDF Data on the Web. In *Proceedings of the 6th World Multiconference on Systemics, Cybernetics and Informatics (SCI2002)*, 2002.

[FLM98] D. Florescu, A. Levy, and A. O. Mendelzon. Database Techniques for the World-Wide Web: A Survey. *SIGMOD Rec.*, 27(3):59–74, 1998.

[FOR03] FORTH Institute of Computer Science. *RQL v2.1 User Manual*. World Wide Web, `http://139.91.183.30:9090/RDF/RQL/Manual.html`, 2003.

[GLMB98] R.V. Guha, Ora Lassila, Eric Miller, and Dan Brickley. Enabling Inferencing. *QL '98, The Query Languages Workshop*, December 1998.

[GMM03] R. Guha, Rob McCool, and Eric Miller. Semantic Search. In *Proceedings of the 12th International WWW Conference*. ACM Press, 2003.

[Güt94] R.H. Güting. GraphDB: Modeling and Querying Graphs in Databases. In J.B. Bocca, M. Jarke, and C. Zaniolo, editors, *Proceedings of VLDB'94, Santiago de Chile*, pages 297–308. Morgan Kaufmann, 1994.

[HAMAS04] C. Halaschek, B. Aleman-Meza, I.B. Arpinar, and A.P. Sheth. Discovering and Ranking Semantic Associations over a Large RDF Metabase (Demonstration Paper). In M. Nascimento, editor, *Proceedings of VLDB 2004*. Morgan Kaufman, 2004.

[HBEV04] Peter Haase, Jeen Broekstra, Andreas Eberhart, and Raphael Volz. *A Comparison of RDF Query Languages*. World Wide Web, `http://www.aifb.uni-karlsruhe.de/WBS/pha/rdf-query/rdfquery.pdf`, 2004.

[HG04] Jonathan Hayes and Claudio Gutierrez. Bipartite Graphs as Intermediate Model for RDF. Technical Report TR/DCC-2004-2, Universidad de Chile, `http://www.dcc.uchile.cl/~cgutierr/ftp/bipartite.pdf`, 2004.

[Mil03] Libby Miller. *Databases, Query, API, Interfaces Report on Query Languages*. WWW, `http://www.w3.org/2001/sw/Europe/reports/rdf_ql_comparison_report/`, 2003. SWAD-Europe Deliverable 7.2.

[MKA+02] Aimilia Magkanaraki, Grigoris Karvounarakis, Ta Tuan Anh, Vassilis Christophides, and Dimitris Plexousakis. Ontology Storage and Querying. Technical Report 308, Institute of Computer Science of the Foundation for Research and Technology - Hellas (ICS-FORTH), April 2002.

[MM04] Frank Manola and Eric Miller. *RDF Primer. W3C Recommendation*. WWW, `http://www.w3.org/TR/2004/REC-rdf-primer-20040210/`, 10 February 2004.

[Olk03] Frank Olken. Tutorial on Graph Data Management for Biology. *IEEE Computer Society Bioinformatics Conference (CSB)*, Aug 2003.

[Pér02] Asunción Gómez Pérez. A Survey on Ontology Tools (Deliverable 1.3), 2002. OntoWeb, Ontology-based Information Exchange for Knowledge Management and Electronic Commerce.

[Say04] Craig Sayers. Node-centric RDF Graph Visualization. Technical Report HPL-2004-60, Mobile and Media Systems Laboratory, HP Labs, Palo Alto, 2004.

[SD02] Michael Sintek and Stefan Decker. TRIPLE - A Query, Inference, and Transformation Language for the Semantic Web. In *Proceedings of the International Semantic Web Conference (ISWC)*, June 2002.

[Sea04] Andy Seaborne. *RDQL - A query Language for RDF (W3C Member Submission)*. World Wide Web, `http://www.w3.org/Submission/2004/SUBM-RDQL-20040109/`, January 2004.

[Str88] Bjarne Stroustrup. What Is Object-Oriented Programming? *IEEE Softw.*, 5(3):10–20, 1988.

[SWG02] Dennis Shasha, Jason T. L. Wang, and Rosalba Giugno. Algorithmics and Applications of Tree and Graph Searching. In *Proceedings of the Twenty-First ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, pages 39–52. ACM Press, 2002.

A Implementations Used

In our experiments we used the following implementations:

RQL is implemented in ICS-FORTH's RDF Suite[7]; the Sesame system implements a subset of it. For our evaluation we used Sesame 1.0[8], which also offers the SeRQL language (Sesame RDF Query Language). Also available in the Sesame system is RDQL, which is currently in the status of a W3C submission [Sea04].

TRIPLE is a query, inference and transformation language for the Semantic Web [SD02]. For our study we used the Triple version of March 14, 2002[9] along with XSB 2.6 for Windows.

N3[10] is a language which is a compact and human-readable alternative to the RDF/XML syntax. N3 is supported by the CWM system, which is written in Python. We used CWM in version 1.147 from March 9, 2004 with Python 2.3.3 for Windows.

The Versa language is supported by 4Suite[11], which is a platform for XML and RDF processing. For the comparison we used 4Suite version 1.0a3 for Windows of July 4, 2003.

RxPath is a language for querying a RDF model using the syntax of XPath. RxPath is part of Rx4RDF[12], which is a set of technologies to query, transform and update RDF. We used Rx4RDF version 0.3.0 from May 12, 2004 along with Python 2.3.3 and 4Suite 1.0a3 for Windows.

[7] `http://139.91.183.30:9090/RDF/`
[8] `http://www.openrdf.org/`
[9] `http://www.dfki.uni-kl.de/frodo/triple/`
[10] `http://www.w3.org/DesignIssues/Notation3.html`
[11] `http://4suite.org/`
[12] `http://rx4rdf.liminalzone.org/`