A Graph-based Approach to Indexing Semantic Web Data

Xin He[1] and Mark Baker[1]

[1] School of Systems Engineering, University of Reading, Whiteknights, Reading, Berkshire, RG6 6AY, UK
{x.he, mark.baker}@reading.ac.uk

Abstract. To the best of our knowledge, existing Semantic Web (SW) search systems fail to index RDF graph structures as graphs. They either do not index graph structures and retrieve them by run-time formal queries, or index all row triples from the back-end repositories. This increases the overhead of indexing for very large RDF documents. Moreover, the graph explorations from row triples can be complicated when blank nodes, RDF collections and containers are involved. This paper provides a means to index SW data in graph structures, which potentially benefit the graph exploration and ranking in SW querying.

Keywords: Semantic Web, search, query, RDF, resource, ontology.

1 Introduction

The task of querying resources on the Semantic Web (SW) is different to information retrieval from the conventional Web. This is mainly due to the forms in which information is stored differing between the Web and the SW, and the distinct levels of semantic support. Instead of web pages and conventional databases, SW data is stored in Resource Description Framework (RDF) documents. RDF data consists of many triples, each of which contains a subject, a predicate and an object, represented using either Uniform Resource Identifier (URI) or Literal (human-readable text). Many existing SW search systems do not index graph structure, but only make mappings from literals to the corresponding resources [5, 7] or documents [2, 3, 4]. Other systems partially or wholly index the RDF graph structure in the backend semantic data repositories. This structure information is represented as individual triples, and is stored either as inverted indices in conventional IR engines [6, 8] or as database records [1]. However, these systems suffer from the following problems:

- Indexing an excessive number of triples. This is very costly when the search engine is geared towards the future SW.
- Limited support for access patterns in systems. Access patterns (S:?:?), (?:P:O), (S:?:O), and (?:?:O) are not efficiently supported in the systems that index triples using IR engines.
- Not supporting complex graph structures. Most of the SW search systems we have studied are not suitable for RDF graphs that owe to the use of blank nodes, RDF collections, and containers. Without the necessary graph structure information

indexed, exploring graphs that include blank nodes, RDF containers and collections relying on row triples can be very complicated.

By analysing the limitations in existing efforts and considering the specific way that SW data is stored, this paper presents a graph-based approach to indexing SW data.

2 Unit-Graph – Handling SW Graph Structures

The SW is not a simple hierarchical tree representing the subsumption relationships between concepts and their instances, but instead a complicated net-based Directed Labelled Graph (DLG) with mutual relations between nodes possibly existing. Established indexing techniques, such as B-trees and hash-tables, are designed for data with hierarchical structure. It is simple to index textual descriptions on the SW using such techniques, but is impractical to index SW graph structures.

Although a whole RDF graph is normally not hierarchical, we found that by dividing it into fractions, it is always possible to represent each fraction using a hierarchical structure. Therefore, an RDF graph can be indexed as a collcetion of the tree-based fractions. In this paper, such a fraction is called a *Unit-Graph*. Figure 1 illustrates three unit-graphs for resources *uorcs:M_Baker*, *uorcs:X_He*, and *ex:06Pa per*, enclosed using blue, green, and red dashed lines respectively. In each unit-graph, the resource being described is called the *Root node*, while each resource describing the root node is called a *Subnode* (of the root node). From the root node to each subnode, only forward links are included. For example, the unit-graph for *uorcs:X_He* includes three literals (in conjunction with the three edges) and one subnode (in conjunction with edge *foaf:publications*). Edge *dc:creator* is not included.

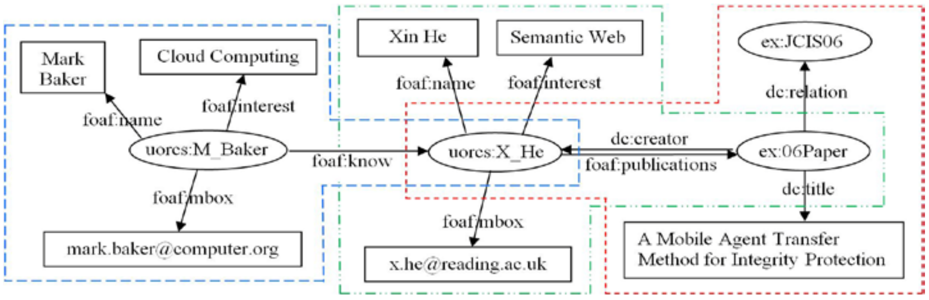

Fig. 1. The unit-graphs for three resources on the Semantic Web

These resources may be described in one SW document, or separately described in multiple documents, and interlinked by semantic links. It should be noted that the only system properties indexed in unit-graphs are *rdfs:label* and *rdfs:comment*. System properties refer to those described by RDF-S and OWL. System properties include properties that are not related to the result resources for each query, such as the restrictions, and ontology versioning. There are also attributes indirectly related to the description of the result resources, e.g. *rdf:type*, *owl:sameAs*, and *rdfs:seeAlso*. These attributes are separately indexed.

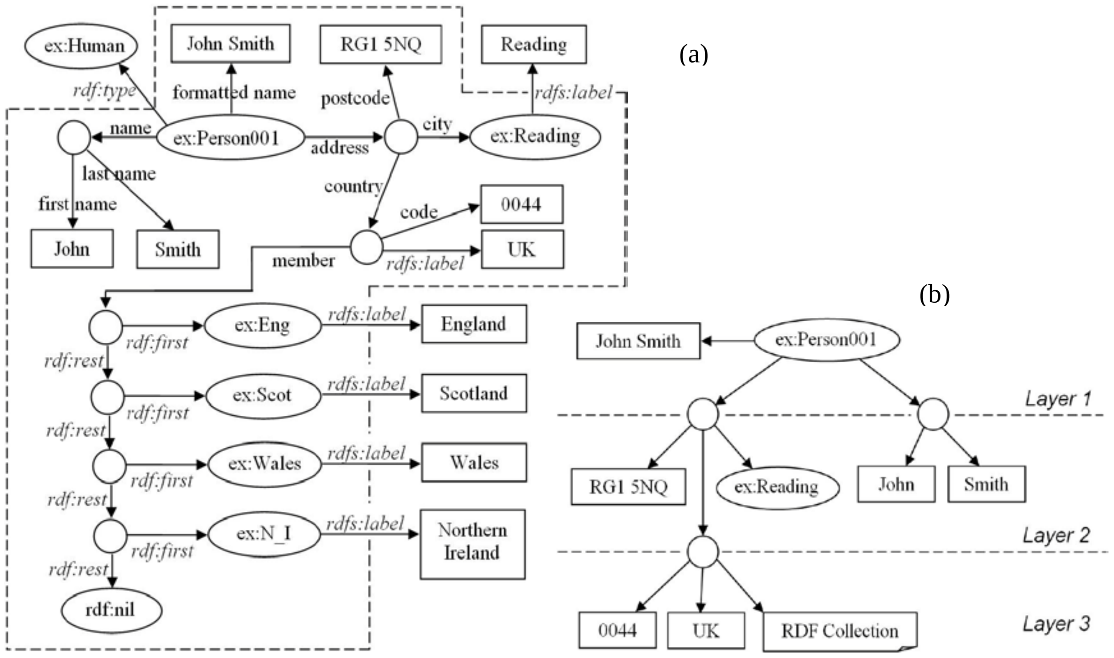

Fig. 2. The unit-graph for a resource that contains blank nodes and RDF collection

Although unit-graphs are strictly hierarchical, they are unsuitable for indexing using existing indexing techniques due to the possible inclusion of blank nodes, RDF containers and collections, e.g. the unit-graph in Figure 2 (a) (enclosed using dashed lines). User defined properties are represented using their label values for simplicity.

Using our approach, each unit-graph is modelled into layers, separated by blank nodes. These blank nodes are not those used in RDF containers and collections. The unit-graph for resource *ex:Person001* (shown in Figure 2 (a)) is modelled into three layers, as illustrated in Figure 2 (b). Property label values are omitted for simplicity.

We can intuitively see that each blank node has a maximum of three types of forward neighbours, that is, subnode, literal, and blank node. Thus, by modelling the root node or each of the blank node as an object, which contains three primitive value lists, storing subnode URIs, literal values, and blank nodes respectively, and letting each of these blank nodes have the type of the same object (as above), the data about the blank nodes in lower layers can be added to the model recursively. Thus, a unit-graph can be indexed layer by layer (from top to bottom) into an object.

In addition, multiple-values for each property (1: n relationship for the subject and object(s) of each property) are supported. Thus, an RDF container or collection can be pre-processed at indexing time and stored in a primitive value list as an item of one primitive value list for its “super” blank node object. The graph structure of RDF containers or collections is not recorded.

Furthermore, in our approach, graph structure and data values are separately indexed. Each literal is assigned an internal identifier, namely *Literal ID*, represented using a positive integer. Two literals (from different graphs) that contain the same content will be assigned the same identifier. Each resource in a unit-graph (typically identified using URI, including the root node, its subnodes and properties) is also assigned an internal identifier, namely *Resource ID*, represented using a positive integer identifier. Thus, all primitive values are represented using integers. This severely minimises the storage size of unit-graphs, and facilitates the reuse of literals and resource URIs in unit-graphs. Moreover, using unit-graphs, graph explorations can be readily performed by matching between subnodes and rootnodes (in different unit-graphs). Due to the use of Resource IDs, the graph exploration process is actually the process of comparing numbers (see whether they are equal) rather than matching long strings (the resource URIs), which is believed to be much more time-efficient.

3 Conclusion

In this paper, we propose an approach to indexing SW data which can address the drawbacks in existing efforts in the same domain. We have presented a detailed tree-based data model to effectively hold RDF graph structures. We clarify how it deals with complex graph structures, especially when blank nodes, and RDF containers and collections are involved. We have presented how internal identifiers are employed to represent literals and resource URIs, and thereby minimises the disk capacity for indexing, and improves system performance. In addition, we explain the advantages our graph-based approach to dealing with RDF graphs has over the triple-based indexing schemes. Our graph-based approach provides ready accesses to the SW graph structures and flexible graph explorations without the need to index an excessive number of triples, and is capable of dealing with complex graph structures.

References

1. Hogan, A., Harth, A., Umbrich, J., Decker, S.: Towards a Scalable Search and Query Engine for the Web. In: Proceeding of the 16th WWW, Poster Session, pp. 1301--1302 (2007)
2. Ding, L., Finin, T., Joshi, A., Peng, Y., Cost, R., Sachs, J., Pan, R., Reddivari, P., Doshi, V.: Swoogle: A Search and Metadata Engine for the Semantic Web. In: Proc. 13thCIKM (2004).
3. Oren, E., Delbru, R., Catasta, M., Cyganiak, R., Stenzhorn, H., Tummarello, G.: Sindice.com: A Document-Oriented Lookup Index for Open Linked Data. In: Journal of Metadata, Semantics and Ontologies, vol. 3, no. 1, pp. 37--52. (2008)
4. d'Aquin, M., Sabou, M., Dzbor, M., Baldassarre, C., Gridinoc, L., Angeletou, S., Motta, E.: WATSON: A Gateway for the Semantic Web. In: Proc. ESWC, Poster Session. (2007)
5. Lei, Y., Uren, V., Motta, E.: SemSearch: A Search Engine for the Semantic Web. In: Proc. EKAW, pp. 238--245, (2006)
6. Zhang, L., Liu, Q., Zhang, J., Wang, H., Pan, Y., Yu, Y.: Semplore: An IR Approach to Scalable Hybrid Query of Semantic Web Data. In: Proc. ISWC2007 + ASWC2007, pp. 652--665. LNCS, vol. 4825, pp. 652--665, (2008)
7. Cheng, G., Ge, W., Qu, Y.: Falcons: Searching and Browsing Entities on the Semantic Web. In: Proc. WWW2008, Poster Session, pp. 1101--1102, (2008)
8. Wang, H., Zhang, K., Liu, Q., Tran, T., Yu, Y.: Q2Semantic: A Lightweight Keyword Interface to Semantic Search. In: 5th ESWC. LNCS, vol. 5021, pp. 584--598, (2008)