

Building a Research Data Commons on Linked Data

Ben Greenwood - u4552016@anu.edu.au (COMP4540, 2013)

Due to personal circumstances, the final stages of this research project were not fully completed. At time of submission, investigation had been completed, hypothesis and approach formed, and much of the initial tooling work completed. Further work involves the completion of the gateway tools and analysis of the constraints, usability and performance implications of the proposed design.

Introduction

The premise of this research project is that Research Data classification, access and reuse can be enhanced by building tools which leverage Linked Data and the Semantic Web to provide discovery and consumption capabilities. We asked the question, to what extent can the expressiveness of semantic web technologies be used to extend research data discovery beyond conventional systems?

The background to this project was the author's contributions to building the infrastructure of the Australian Research Data Commons, a data interchange standard and collection of services based on a harvester architecture and XML technologies (operated by the [Australian National Data Service](#)).

Some limitations of the existing approach are that this fairly uncommon XML schema (loosely based on ISO19115/139) forces a distinction between “data” and “metadata” and restrictive XML validation requires a lossy transformation from rich discipline-specific descriptions of the data. Further, each implementation must understand the schema's various elements and vocabulary in order to make their data available for harvest into discoverability services. This results in metadata duplication and high

barrier to entry for data rich organisations, many of whom may already have existing data descriptions but are inhibited by the transformation to the [RIF-CS standard](#).

This closed-world approach to data management is largely ineffective as these heterogeneous systems are forced to be reconciled by each distinct metadata custodian, under a variety of different influences and assumptions, resulting in disparate and often incompatible data descriptions.

Importantly, in a Linked Data approach, these restrictions are largely overcome through the “open world assumption” and ability to build, share, map and reason between known ontologies.

Approach

Our approach consists of three stages. Firstly, we must investigate a base model to describe research data in RDF as this would be fundamental to any proposed system architecture. It was determined that this area had been extensively researched and that further consideration would be restricted to incremental solutions.

Secondly, we would ensure that data descriptions exist or are compatible with an RDF-based model. It was established (through extensive investigation) that the level of abstractness of existing Linked Data implementations in the Australian Research Data sector differ too greatly to be practical for this experiment. This is partially a consequence of a chicken-and-egg scenario, wherein research organisations have fewer incentives to provide consistent data descriptions in RDF, given the absence of tools that leverage this for discovery.

To resolve this, we looked to options for migrating existing XML-based data into an RDF model and discovered work by Haslhofer¹, including a tool named [OAI2LOD](#) (a mechanism for migrating data from an [OAI-PMH service provider](#) to a RDF-based Linked Open Data model). This made a good candidate for generating test data, as the existing Australian Data Commons provided an existing [OAI-PMH harvest point](#). This data was generated by building a loose mapping to the vocabularies in RIF-CS and the ANZSRC Field of Research codes and these vocabularies were then [published](#).

Thirdly, and most importantly, software tools would need to be built which implement classification and logic reasoning in order to augment data discoverability within the linked data environment. Essentially, the challenge would be to ascertain whether similar data discovery capabilities could be enabled using this technology architecture as are available in existing [Data Discovery Portals](#).

Proposed System

The initial proposed system planned to implement a thin application layer over an existing [RDF Triple Store](#) implementation with [OWL Reasoning](#) support. In order to match the capabilities of the existing XML implementation, the graph store was required to offer capabilities such as temporal and fulltext search, spatial querying and scalability to hundreds of thousands of individuals/records. To augment these capabilities (and validate the research hypothesis), we also looked for an implementation which offered logical reasoning, on the basis that this would provide the ability to join the lookup capabilities with hierarchical reasoning over structures such as the ANZSRC Subject Codes.

¹ Haslhofer, B. *A Web-based Mapping Technique for Establishing Metadata Interoperability*. <http://eprints.cs.univie.ac.at/307/1/phd_haslhofer_final.pdf>.

OWL reasoning profile restrictions

It was discovered that the OWL reasoning capabilities across terminological/schematic structures (TBox) and instance assertions (ABox) were restricted. This prohibited the type of hierarchical reasoning we were hoping to achieve such as *“find me all research data sets which have subject codes anywhere in the branch of Mathematical Sciences”*.

In SPARQL algebra, these queries translate into “hybrid basic graph patterns”, which are [known to be computationally intractable](#) in certain [profiles of OWL](#), most notably [OWL 2 DL](#) (which provides semantics such as class expressions (e.g. “isSubClass of”)).

To resolve the issue OWL reasoning restrictions, it was posited that, given prior knowledge about the structure of the TBox and bounding the structure of the TBox in the ontology design stage, assumptions could be made as to the computational complexity of the hybrid graph pattern. In order to achieve this, we could either modify the reasoner to lift the OWL profile restriction or decompose hybrid BGPs into a logical disjunction of a subquery for matching predicates in the TBox. The former was proven to be trivial, although further experimentation as to the limitations of assumptions around ontology design was required during the evaluation stage.

No “all purpose” Triple Store

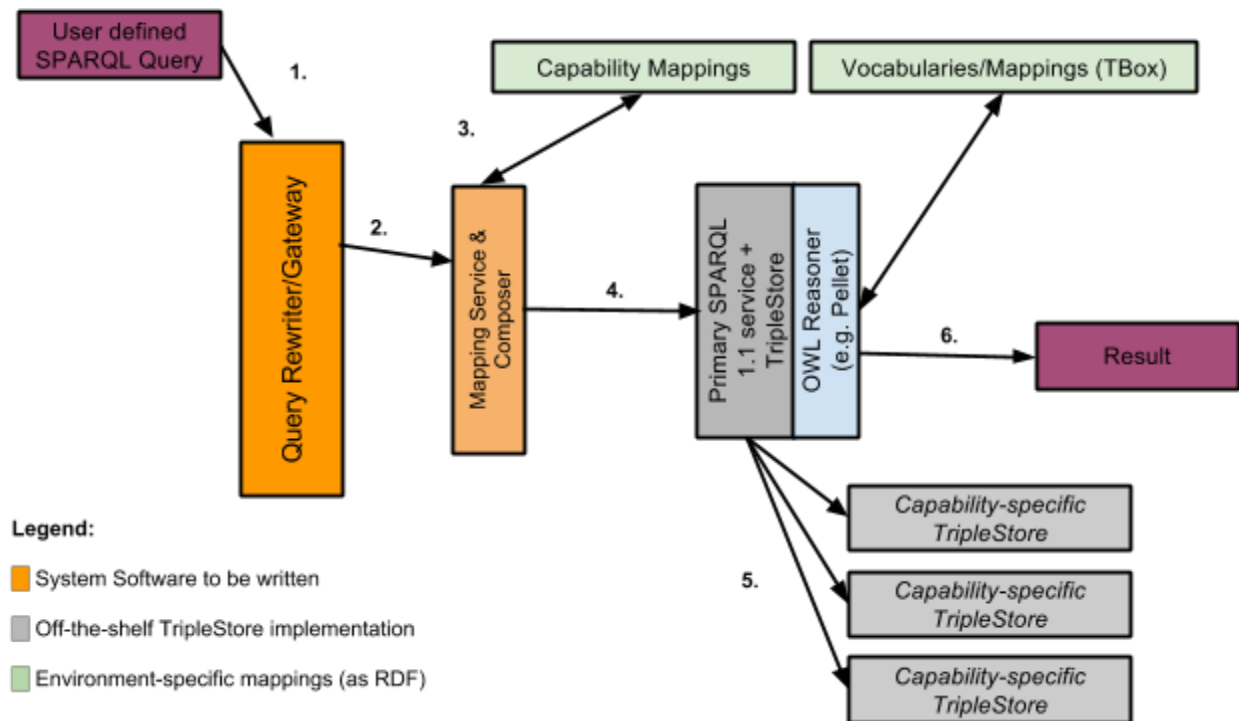
After extensive analysis of [major Triple Store implementations](#), it was discovered that the combination of spatial, temporal, fulltext search and OWL reasoning capabilities were not available in a single comprehensive implementation. Instead, different stores implemented and optimised for different use cases by offering distinct capabilities. The general purpose nature of a Data Commons system made this a fundamental challenge to the project.

This led to the major research challenge: building a mechanism whereby these various capabilities could be exposed through a single SPARQL endpoint in a design which enabled the reasoning and “all purpose” query structure hypothesised above.

The planned implementation architecture used [SPARQL 1.1 Federated Queries](#) mechanism to “contract out” capabilities to known systems where these are supported. Using this standard, the `SERVICE` keyword can be used to indicate to an endpoint that a subcomponent of the query should be executed on a different service endpoint. A list of known mappings from capability-specific predicates to the service endpoints which support them would be integrated into a *query rewriter* which would parse an initial SPARQL query and rewrite it into a form which utilised the federated query mechanism.

In order to achieve this, an initial query would need to be parsed into [SPARQL algebra](#) and then that algebra evaluated to identify matching expressions which could be rewritten.

Proposed Architecture



1. User submits a SPARQL Query (either manually or with the aid of a web UI), for example:
“find me all research data sets which have a subject in the branch of Earth Sciences, created since 2010 and cover the area of South East Queensland”
2. Query Rewriter/Gateway receives this request and parses the SPARQL query into SPARQL algebra. This algebra is passed on to the Mapping Service.
3. The Mapping Service traverses the algebra in an attempt to locate predicates which are known in it’s Capability Mappings. When a match is found, the logic associated with that subject and predicate is transformed out of the main query and into a subquery, contracted to the appropriate capability-specific TripleStore using the Federated Query syntax in SPARQL 1.1.

Once this transformation is completed, the algebra is composed back into a SPARQL query string so that it can be passed to the system's primary SPARQL endpoint for execution. The reason for this is so that the Mapping Service and Composer do not have to implement the execution logic for the algebra or be responsible for optimisation of said algebra.

4. The query string is received by an off-the-shelf Triplestore with SPARQL + OWL reasoning (e.g. [Stardog](#)). The SPARQL service will execute the query and reason across hybrid and TBox graph patterns using the terminological definitions contained in Vocabulary Mappings.
5. Any federated query components are contracted off to capability-specific TripleStores. This requires that results be keyed with a shared identifier, although the subject URI identifying the dataset is an obvious candidate for this.
6. The primary SPARQL service will compile the results based on the query logic and return this to the user in an RDF format.

Software Artefacts

This research project had two principal software artefacts:

1. An experiment workbench ([source code](#))

The chosen Primary SPARQL Service & TripleStore (Stardog) provides only a command-line management interface for interacting with it. This requires the user to compile queries into text files, execute on the command line and then view results in an output RDF file. This is an expensive process and hinders the velocity of experimentation.

To overcome this, I built a web user interface for interacting with the underlying configuration and data in Stardog. This required interfacing with Stardog using a combination of the [SPARQL 1.1 HTTP Graph Store protocol](#) and [Stardog's own proprietary HTTP API](#). This enabled an experiment “workbench” with the following capabilities:

- Web-based query interface with validation and table-formatted result visualisation
- List interface to show and manage current graphs in the store
- Graph editor to edit triples within a specific graph using TTL format
- Settings manager for modifying Stardog's command-line variables (e.g. OWL profile)
- Ability to list all active namespaces
- Ability to purge all data from the current TripleStore

Editing Graph: <http://purl.org/au-research/data/#first-fleet-and-early-settlement-a-digitised-suite-of-first-hand-accounts-diaries-and-letters-written-by-men-and-women-who-emigrated-to-australia-in-the-first-fleet>

TTL input: (regularly used prefixes will automatically be declared)

```
#prefix for: <http://purl.org/au-research/vocabulary/anzsrc-for/2008/> .
#prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
#prefix dct: <http://purl.org/dc/terms/> .
#prefix deat: <http://www.w3.org/ns/deat#> .
#prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .

_:genid1
  a for:for-210303 ;
  rdf:value "Australian History (excl. Aboriginal and Torres Strait Islander History)" ,

_:genid2
  a for:for-21 ;
  rdf:value "HISTORY AND ARCHAEOLOGY" ,

_:genid3
  a for:for-2103 ;
  rdf:value "HISTORICAL STUDIES " ,
```

Update Graph

Graph Representation:

Graph: <http://purl.org/au-research/data/#first-fleet-and-early-settlement-a-digitised-suite-of-first-hand-accounts-diaries-and-letters-written-by-men-and-women-who-emigrated-to-australia-in-the-first-fleet>

```
_:genid1
  → rdf:type → for:for-210303
  → rdf:value → "Australian History (excl. Aboriginal and Torres Strait Islander History)"

_:genid2
  → rdf:type → for:for-21
  → rdf:value → "HISTORY AND ARCHAEOLOGY"

_:genid3
  → rdf:type → for:for-2103
  → rdf:value → "HISTORICAL STUDIES "

_:genid4
  → rdf:type → for:for-210301
  → rdf:value → "Aboriginal and Torres Strait Islander History"

http://researchdata.andis.org.au/first-fleet-and-early-settlement-a-digitised-suite-of-first-hand-accounts-diaries-and-letters-written-by-men-and-women-who-emigrated-to-australia-in-the-first-fleet
  → dct:title → "First fleet and early settlement: a digitised suite of first-
```

Figure 1. Graph editor for a specific sample data record

Query Result:





















?graph_name	?triple_count	
rda:the-akchakhan-kala-kazaki-yaktan-kazakly-yatkan-wail-paintings-collection	"33"^^xsd:integer	 
rda:classic-texts-in-australian-and-international-taxation-law	"13"^^xsd:integer	 
rda:tin-sheds-gallery-archive	"31"^^xsd:integer	 
rda:history-of-nursing-and-nursing-education-archive	"58"^^xsd:integer	 
rda:impact-of-receptor-density-on-calcium-sensing-receptor-casr-mediated-intracellular-calcium-signalling-dataset	"32"^^xsd:integer	 
rda:first-fleet-and-early-settlement-a-digitised-suite-of-first-hand-accounts-diaries-and-letters-written-by-men-and-women-who-emigrated-to-australia-in-the-first-fleet	"16"^^xsd:integer	 
rda:national-recording-project-for-indigenous-performance-data-collection	"37"^^xsd:integer	 
rda:randomised-controlled-trial-dataset-efficacy-and-safety-of-ascorbic-acid-vitamin-c-supplementation-for-children-with-charcot-marie-tooth-disease-type-1a	"39"^^xsd:integer	 
rda:university-of-sydney-art-collections	"60"^^xsd:integer	 
rda:antarctic-paleobath-antarctic-paleo-depth-grids	"38"^^xsd:integer	 

Figure 2. Result list for a specific query

This system leveraged an existing RDF library for parsing SPARQL queries. The library had to be extended to include support for the HTTP Graph Protocol and interact with SPARQL endpoints which require authentication. Further, the workbench required a new [homemade library for interfacing with Stardog's own HTTP API](#).

2. Query Rewriting Gateway & Composer ([source code](#))

The second obvious software artefact was the Query Rewriter gateway and Composer. This early stage tool is able to:

- parse an initial query string into SPARQL algebra
- identify the components which are candidates for rewriting to federated queries using a hard-coded capability mapping ([gateway/experiment_harness.py](#))
- recompose the modified SPARQL algebra into a SPARQL query string ([utils/algebra_composer.py](#))
- print the recomposed query string to screen for manual validation

Building this tool required extension of the current Python rdflib library to add support for the SPARQL 1.1 Federated Query algebra as well as building the “recompose” graph traversal which turns the algebra back into a query string (a feature not previously available in rdflib).

Further work

Software implementation of the Query Rewriter is incomplete:

- the recomposed query string should be sent to the Primary SPARQL service
- the subsequent result proxied back out to the user
- the recompose library does not currently support [evaluation](#) of the entire SPARQL algebra
- the capability mappings should be stored within a graph in the primary TripleStore for better maintainability

Once these steps are completed, work is required to implement a test architecture with multiple triplestores, feed the generated test data (from an OAI2LOD dump) into these stores and configure the appropriate capability mappings.

At this point, experimentation can be conducted to establish the performance of this architecture, the ease of use of the user-generated SPARQL queries and the extent of limitations which are required to be put on the TBox mapping in order to provide an acceptable query response time.