Transformations for aggregating Linked Open Data

Lukas Koster (Library of the University of Amsterdam)
Ivo Zandhuis (Ivo Zandhuis Research & Consultancy)
SWIB 2018, Bonn

Abstract

Linked Open Data (LOD) is usually provided as-is. Institutions make choices how to model the data, including properties, blank nodes and uri's, for valid reasons. If you want to combine data, there are generally two options: 1) do a distributed query and inference on the data 2) aggregate the data into a new, single endpoint. Distribution enables the use of all available up-to-date data structures, aggregation enables more easy-to-use data and better performance. The challenge is how to get the best of both worlds.

For aggregation it is good practice to do transformations to obtain the needed convenience. We give an overview of the transformation types needed, learned in the AdamNet Library Association project AdamLink (http://adamlink.nl), a collaboration of the Amsterdam City Archives, Amsterdam Museum, University of Amsterdam Library, Public Library of Amsterdam and International Institute of Social History. The objective is to create a linked open data infrastructure connecting the member institutions' collections on the topic of "Amsterdam", targeted at reuse by researchers, teachers, students, creative industry and general public.

We discuss the advantages and disadvantages of creating an aggregation vs. distribution of queries. Every transformation type should solve a distribution problem in order to be useful. But transformation probably reduces querying options on the data. We therefore need to get the best trade-off between complexity and usability. An interesting option to investigate is to apply a caching node mechanism, that could combine the best of both worlds.

We distinguish 6 types of transformation:

1. Ontology alignment
2. Authority alignment
3. Object types alignment
4. Adding aggregation management statements
5. Restructuring data
6. Data typing

We will illustrate the transformations with real examples. We will also discuss the issues with feeding back the enriched data in the cache or aggregation to the original data sources.

Ontology alignment

A familiar and classic problem is the mapping of different data modeling schema's. After years of discussion about how to create a general model for all types of objects, we found middle ground where every type of object has its own schema. In addition to that, schema's can be combined to enable the modeling of data that is not in a chosen basic schema.

In our project we decided to use Dublin Core as a base to provide CHO (Cultural Heritage Object) metadata, just like the European Data Model (EDM) does. One of our data providers uses schema.org to describe publications, so for our aggregation some of the statements with schema.org-properties must be transformed into Dublin Core. The property schema:about, for instance, is mapped onto dc:subject and schema:author is mapped onto dc:creator.

Authority alignment

Less familiar is the mapping needed for thesauri and authority lists. The Amsterdam Museum uses a nationally standardized list of artists (called "RKDartist") because of their art-historical nature, while the libraries in the project use the nationally standardized list of authors (which is related to VIAF). In our combined dataset we need to map all these persons to the same URI in order to enable convenient querying. Translating all persons to VIAF or translating all persons into RKDartist would result in the loss of either some artists not in VIAF or some authors not in RKDartist. We solved this issue by introducing our own URI, with links to the other standardized list or both. Our own list only consists of owl:sameAs relations to other authority files.

For object-types we were able to use the Art & Architecture Thesaurus (AAT). Here all the types we needed were included, so we did not need to create our own. We map all locally defined types to an AAT-uri.

Some datasets contain literals for things we'd like define as a URI. For these properties we try to match strings to standardized lists as much as possible. We use strict matching to prevent semantic errors as much as we can. The providing institution can use our matched data as starting point for adding validated data into their own systems. We stimulate this by showing the advantages of using the standardized list.

The use of thesauri and authority lists is one of the main objectives of our project. We believe that the linking of objects through common creators, subjects, types and locations is the big advantage of using LOD as a method. It enables queries that were unthinkable only a couple of years ago and demonstrates to our partners the reasons to shifting to LOD as a data providing method.

Object types alignment

We encountered a similar problem with the formal ontological object typing in the data. With the rdf:type property you define the options and constraints as defined in the ontology. For our purpose we wanted to standardize the object types as much as possible. We define for instance all our CHO's as edm:ProvidedCHO even if the providing institute did not include this typing in its own dataset.

Besides that, we use our own set of object types. We decided for instance to use schema:Person instead of foaf:Person, so we transform rdf:type properties from the latter to the first.

Adding aggregation management statements

By transforming the datasets that were provided by the institutions, we create our own environment of datasets. We therefore add our own aggregation data management metadata for users to query. The main statement we add, for instance, is the void:inDataset property with our own uri for the dataset we created.

Other examples of statements that we add are derivations of statements that were provided by the institution. For our own standardization we need to add a statement with the property dc:type. Sometimes this could be derived from the object typing stated with the rdf:type property. A CHO provided as schema:Book entails a derived statement with dc:type and the appropriate AAT-uri.

Restructuring data

Some ontologies are fairly complex in order to model complex situations. These ontologies could be overly correct in our context, where we want to provide an easy-to-use SPARQL-endpoint. EDM for instance prescribes a distinction between the object itself and its various digital reproductions. The relation between those two concepts decently models the needed metadata of the various reproductions. Subsequently you need a concept (in the case of EDM: ore:Aggregation) that combines the CHO and the reproductions.

This model can make querying tedious and low in performance. We've therefore chosen to add the reproductions directly to the metadata of the object with the foaf:depiction property. This enables easy access and easy querying.

Data typing

Finally we want to touch on the subject of data-typing. Lots of statements have literals as their object. But sometimes they can be data typed. This mainly applies to the dates data type with xsd:dateTime, for instance the date of creation stored with the dc:date property. We did not study this part of the transformation much, but we experienced difficulties in the use of this data, so we probably need to start investigating this soon.

Conclusion: caching nodes

With these six transformation types, linked data aggregations can perform the role of transparent layers in a network of caching nodes, besides acting as service platforms for specific contexts. The role of caching node means that an aggregation does not function as a goal in and for itself, but as a hub in a network of equal hubs where all data originates from the sources. Aggregators then are responsible of curating this network between nodes.