

# From Spreadsheets to 5-star Linked Data in the Cultural Heritage Domain: A Case Study of the Yellow List\*

Audun Stolpe

audus@ifi.uio.no

Martin G. Skjæveland

martige@ifi.uio.no

Department of Informatics, University of Oslo

## Abstract

This paper presents the results and lessons learnt from converting the Cultural Heritage Management Office in Oslo's Yellow List of architecturally and culturally valuable buildings into a Linked Data server. The study was performed in the context of the Semicolon II project, as a foray into the more general problem of lifting public sector information. The paper argues, that whilst the benefits of adopting the Linked Data distribution model are many and important, the required transformation to RDF is not unproblematic. Specifically, one needs to ensure that the information content of a dataset is preserved undistorted despite the change of representation. This can not be assumed to be the case by default, since a transformation will often alter the very structure of the dataset. The paper draws attention to the concept of a bounded RDF homomorphism as a means to address this problem.

## 1 Introduction

In recent years the political pressure for reusable public sector information has built tremendous momentum. Internationally, pride of place must be awarded to the European Public Sector Information Directive of 2003, which at the time of writing is implemented into national law by all member states as well as Norway. In Norway specifically, the Ministry of Government Administration, Reform and Church Affairs has included publication of government data as one of five common regulatives for all government agencies as of 2011. Yet, whilst these initiatives are beginning to pay off—more government data is available on-line now than ever before—there is at the same time relatively little attention to formats and best practices. Data is published, granted, but formats vary from simple HTML tables to data in proprietary formats such as PDF and Excel, making it hard to combine, compare and reuse information on a large scale.

The World Wide Web Consortium (W3C)<sup>1</sup> advocates a solution based on the Linked Open Data paradigm (LOD),<sup>2</sup> which is a recommended best practice for exposing, sharing, and connecting pieces of data, information, and knowledge on the Web using the RDF data model (about which more will be said in Section 2). To promote open data in

---

*This paper was presented at the NIK-2011 conference; see <http://www.nik.no/>.*

\*This research was partly funded by the Norwegian Research Council through the Semicolon II project.

<sup>1</sup><http://www.w3.org/>

<sup>2</sup><http://linkeddata.org/>

general and Linked Open Data in particular, Tim Berners-Lee suggested a 5-star deployment scheme in 2009 [2] which, slightly modified, looks like this:

- ★ make data available on the Web (whatever format) under an open license
- ★★ make it available as structured data (e.g. Excel instead of image scans)
- ★★★ use non-proprietary formats (e.g. CSV instead of Excel)
- ★★★★ use URIs to identify data items, so that they can be referenced on the Web
- ★★★★★ link your data to other's data to provide context

It should be stressed that this scheme does not require existing data repositories—be they spreadsheets or relational databases—to be replaced with RDF representations. It is not reasonable to expect public agencies in particular to do so: every public agency faces unique challenges in collecting, managing, and making information and services available electronically. These issues include policies which control, at times in specific and procedural detail, how information must be handled, who has access, if or not it can be distributed, and if it can, when distribution is allowed [1]. Most institutions evolve their own distinctive bureaucratic culture to deal with these problems which often becomes deeply immersed in a particular technology—commonly relational databases or spreadsheets. Fortunately, RDF, as an abstraction over tabular data, is ideally suited for *exposing* data without actually altering it. The Linked Data guidelines therefore recommend that “you leave the existing system undisturbed, and find a way of extracting the data from it using existing export or conversion facilities. You add, a thin shim to adapt the existing system to the standard” [3].

This is the approach we have taken in the current case study, which has been performed in the context of the NCR-funded Semicolon II project.<sup>3</sup> The case in question is the Yellow List of architecturally and culturally valuable buildings in Oslo [7]. This list is maintained by the Cultural Heritage Management Office in Oslo and published monthly as an Excel spreadsheet. The Yellow List constitutes an excellent case study for several reasons. In general, curators in the cultural heritage domain have much to gain from making their holdings as widely available as possible (given, of course, that it is legally open). As of today, cultural heritage data is often gathered and stored in separate and isolated repositories that reflect how the sector is organised (cf. [4]), without particular attention to the conceptual connections between different archives. The ideal would be to store this information in such a way that web agents could query across physically distinct repositories to compile information based on the conceptual content of the query. As regards the Yellow List in particular, it will serve well as a case study since it contains non-aggregated ‘raw’ data about geographically locatable entities. As such it has great potential for reuse, e.g. in geographical aware devices such as smart phones. Moreover, since the data is legally open, no legal or organisational hurdles has to be cleared before republishing it as Linked Data. This has allowed us to focus primarily on the conceptual challenges of modelling the Yellow List list in RDF, on which it is the purpose of the present paper to report.

The paper is organised as follows: since we wish the paper to be accessible to a non-specialist audience, we will not assume familiarity with RDF and the Linked Data paradigm, and essential background is therefore provided in Section 2. Section 3 reports the most important findings wrt. the modelling of the Yellow list in RDF. It argues firstly that there are good reasons for deviating from the plain tabular structure of the Yellow List in the RDF translation of it, and secondly that this raises the general question of how

---

<sup>3</sup><http://www.semicolon.no/>

to maintain the primary nature of the data despite the change of representation. Section 4 explains how the translation of the Yellow List is displayed and visualised on the Web, whilst Section 5 reports on the modelling of the meta-data.

## 2 The Linked Data Paradigm

In order to understand the concept and value of Linked Data it is instructive to compare it with the document Web, that is, with the Web as one would normally understand it. The document Web is built around a small set of standards, namely Uniform Resource Identifiers (URIs) as a globally unique identification mechanism, the Hypertext Transfer Protocol (HTTP) as a universal access mechanism and the Hypertext Markup Language (HTML) as a format for describing the *layout* of a Web page [6]. The Web as a *web* is created by setting hyperlinks between different documents that may reside on different servers. This in turn tends to foster a high degree of connectedness between relevant content, making it easily findable and searchable.

The essential idea behind the Linked Data paradigm is to apply this very general architecture, not to documents, but to *data*. More specifically, Linked Data refers to a set of best practices for publishing structured data on the Web that are commonly summarised by the so-called Linked Data principles (slightly modified from Berners-Lee's original design note [2]): 1) use URLs as names for things, 2) let the URLs resolve to pages that document or aid the interpretation of that thing, 3) provide an RDF description of the same thing at the same URL, 4) include links to other relevant data. Linked Data thus defined represents a conservative extension of the document Web with two new features: URLs are used to refer not only to Web documents but to *things* more generally (principle 1), and URLs resolve to *different representations of a thing* depending on who is asking; a human reader is to be directed to a conventional Web page with human readable content, whereas an electronic agent is to be offered a *machine-readable* description of the same thing in the form of an RDF document (principles 2 and 3).

Using URLs as names for things is a kind of Columbi egg solution that solves a whole range of problems in one fell swoop. First of all, the use of URLs as names for things hard-wires the ability to track the origin and interpretation of data into the Linked Data paradigm [6]. URLs perform dual service as names of things and documentation of those names, so anyone can dereference a particular URL to determine what the issuer says about the entity that that URL denotes. The effect of this is on the one hand to move meta-data upstream to the source of the data, and on the other, to ensure that the meta-data stays with the data throughout its life-cycle. Secondly, since URLs resolve uniquely—that is, every URL constitutes a single unique node in the Web—there is little risk of names colliding. Stated differently, URLs as names may be thought of as keys that are valid in a Web-wide scope. Linked Data can therefore in principle be mashed-up with any other piece of Linked Data on the Web without having to qualify names first.

As regards principle 3, the Web is intended to be an information space that may be used by humans as well as machines. Both should be able to retrieve representations of resources in a form that meets their needs. Usually this means HTML for humans and RDF for machines. The choice of which kind of representation to return to a client, HTML or RDF, is delegated to an HTTP mechanism called *content negotiation*. The basic idea of content negotiation is that HTTP clients, such as e.g. ordinary Web browsers, sends a header with each HTTP request that indicates what kind of content it prefers. The server inspects that header and selects a response that most closely matches the client's preferences [6]. In this way Linked Data makes information on the Web consumable

by humans *as well as* machines. Moreover, data is not only *on* the Web as something appended to Web pages. It is part of *the very fabric* of the Web, meaning that it stretches in all directions establishing points of contact between relevantly similar datasets. In short the Linked Data Web is no longer primarily about hypertext it is about *hyperdata*.

As regards RDF (*The Resource Description Framework*) it is essentially a language for describing relationships between URLs or between URLs and literal data values, such as strings and numbers. Since in the Linked Data paradigm URLs serve as names for things in general, RDF may be thought of as a language for talking about the relation between entities of any kind referenced by URLs. Importantly, RDF treats the relation itself as one such thing. That is, in RDF a relation is considered an item in its own right and is assigned a URL for unambiguous reference. Why? A quick comparison with HTML will bring the point to the fore: whereas hyperlinks in HTML indicate that two documents are related in some way, they leave it entirely up to the reader to infer the nature of this relationship. In contrast, RDF allows the issuer of a dataset to state explicitly how a connection is to be interpreted by hooking it up with explanatory meta-data just as one would with an ordinary data item such as a protected building or a company, say. In this paper RDF will be represented using the Turtle serialisation. Example:

```
<http://sws.ifi.uio.no/gulliste/kulturminne/209/367/6643181/596689>
  hvor:bydel <http://dbpedia.org/resource/Sentrum,_Oslo> ;
  geo:long "10.7291557399684"^^xsd:decimal ;
  geo:lat "59.914761205120215"^^xsd:decimal .
```

URLs enclosed in angular brackets denote URLs written in full, while words without quotes, e.g. `hvor:bydel` (eng: borough), denotes shorthand URLs written using declared prefixes. Words in quotes are literals. The URL on the first line denotes a building on the Yellow List. The second line relates this building to a borough in Oslo via a predicate `hvor:bydel`. The third and fourth lines supply the building with a pair of decimals that constitute its geographical coordinates. Note that the least unit of assertive significance is a *triple* consisting of a subject (the building), a predicate (e.g. `hvor:bydel`) and an object (e.g. the borough ‘Sentrum’). Any set of such triples is called an RDF *graph*, since it may be represented in the form of a directed graph with labelled edges.

### 3 Modelling the Yellow List in RDF

The Yellow List contains approximately 12300 structures, most but not all of which are buildings. There are also cooking pits, burial mounds, remnants of streets, and more. The spreadsheet has 27 attributes that record information about a wide variety of aspects relating to a building or other structure, e.g. year of construction, the type of structure it is, and the architect that designed it; an excerpt translated to English is found in Table 1:

Property	Street Name	House No.	Borough	Type	Northing	Easting
30/150	Abbedikollen	2	3	House	6644446	593153
233/435	Svingen	12	14	Apartment block	6641752	599503
218/13	Akersbakken	16	4	Cemetery	6644221	597721

Table 1: Excerpt from the Yellow List [7].

The attributes of Table 1 from left to right are: the cadastre code identifying the real property on which the structure is located, street name or place name, house number,

borough id, the type of the structure, northing and easting in the UTM coordinate system. In converting the Yellow List to RDF we used the XLWrap<sup>4</sup> package, a spreadsheet-to-RDF wrapper which is capable of transforming spreadsheets into arbitrary RDF graphs based on a mapping specification. It supports Microsoft Excel and OpenDocument spreadsheets such as separated values files, and it can load local files or download remote files via HTTP. It also comes with an extensible function library which is handy for processing cell values. We used the function library e.g. to transform UTM coordinates to latitudes and longitudes, since the latter are more suitable for Web development and data visualisation.

As regards the transformation itself, it is well known that there exists a simple algorithm to convert any single table to RDF. The steps are: 1) assign a unique URL  $u_r$  to each row  $r$  in the table, 2) assign a unique URL  $u_c$  to the each column  $c$  in the table, and 3) extract the value  $v$  from the cell at row  $r$  and column  $c$  and add the triple  $(u_r, u_c, v)$  to the RDF graph. We shall refer to this as the *naïve* algorithm. The naive algorithm produces an RDF graph that is *isomorphic* to the table in question, in the obvious sense that each cell translates to a unique RDF triple, and the resulting RDF graph contains no other triples.

However, isomorphy is actually too severe a restriction in most cases, since there are usually other sources of information than that which is explicitly stated in the spreadsheet and which have an important bearing on how the data ought to be rendered. In the case of the Yellow List, important information is, first of all, implicit in what may be considered the underlying logical structure of the spreadsheet. This is easily seen by considering what the primary keys in the spreadsheet are. To ensure that each row is assigned a unique URL in the RDF representation, it is generally useful to encode a primary key from a table into the URL representing that row. This has the additional benefit of giving the URL a certain value as a mnemonic device for the entity in question. In the Yellow List the only primary key is the pair consisting of the UTM coordinates. Importantly, cadastre codes are *not* keys, as several rows may have the same cadastre code. This many-to-one relationship is indicative of the underlying *mereological structure* of the data: different buildings, identified by their spatial location, may form part of the same real property, identified by the cadastre code.

## The Implicit Mereological Structure of the List

In order to express the mereology implicit in the list, we first designed two sets of URLs, one to represent physical structures, as given by geographical coordinates, and one to represent the more abstract juridical concept of a real property, as represented by cadastre codes. Thus, for every cadastre code, say 218/13, and every pair of geographical coordinates, say (6644221, 597721), the URL `<http://sws.ifi.uio.no/gulliste/kulturminne/218/13/6644221/597721>` refers to the physical structure which is located at that geographical point, whilst `<http://sws.ifi.uio.no/gulliste/kulturminne/218/13>` denotes the real property of which that physical structure forms part. Note that it is not strictly necessary to include the cadastre code in the URL that identifies the physical structure, since the geographical coordinates are already a key for it. However, in addition to its mnemonic benefits as a pointer to the associated real property, this representation gives a RESTful feel to the published data, since any look-up of a building may be used to retrieve the real property of which that building forms part by simply chopping off the geographical coordinates that identify it.

---

<sup>4</sup><http://xlwrap.sourceforge.net/>

Next, we marked up the relation of parts to wholes by embedding the Yellow List in the CIDOC Conceptual Reference Model (CRM).<sup>5</sup> The CIDOC CRM is promoted by The International Community of Museums (ICOM), a non-governmental organisation maintaining formal relations with and having a consultative status with UNESCO, and provides definitions and a formal structure for describing the implicit and explicit concepts and relationships used in cultural heritage documentation. There is thus a strong and independent reason (independent of the mereological issue, that is) for linking the Yellow List to the CIDOC CRM vocabulary. The closest we could get to the notion of a man-made structure, paradigmatically a building or a house, in the CIDOC CRM is the class `E25.Man-Made_Feature` whose documentation reads “This class comprises physical features that are created by human activity (...) No assumptions are made as to the extent of modification required to regard a feature as man made”. There is no class for buildings specifically. For the concept of a real property we chose the type `E27.Site`, which is a subtype of `E72.Legal_Object`. The documentation for the latter reads “This class comprises those material or immaterial items to which instances of `E30.Right`, such as right of ownership or use can be applied”. Finally we related features to sites using the relations `P46B.forms_part_of` and `P46F.is_composed_of`, which are permissible relations between `Man-Made_Features` and `Sites`, and compatible with `Legal_Objects`. This yields the following core RDF structure:

```
<http://sws.ifi.uio.no/gulliste/kulturminne/218/13/6644221/597721>
  rdf:type cidoc:E25.Man-Made_Feature ;
  cidoc:P46B.forms_part_of <http://sws.ifi.uio.no/gulliste/218/13> .
<http://sws.ifi.uio.no/gulliste/218/13>
  rdf:type cidoc:E27.Site ;
  hvor:gnr "218"^^xsd:int ;   hvor:bnr "13"^^xsd:int ;
  cidoc:P46F.is_composed_of
    <http://sws.ifi.uio.no/gulliste/kulturminne/218/13/6644221/597721> .
```

This code represents *one* physical structure as it relates to the real property it is part of. The only attributes that are attached to the real property as such, are the Norwegian cadastral divisions, i.e. ‘gård’, most akin to the English *lot*, and ‘bruk’, which subdivides lots. These concepts are represented in the code as `hvor:gnr` `hvor:bnr`, respectively.<sup>6</sup> All other information in the Yellow List is most aptly considered attributes of individual physical structures (which are usually but not always buildings) for instance addresses: the same real property may comprise buildings with different addresses, whence an address is not primarily an attribute of the real property.

---

<sup>5</sup>We are grateful to one of the anonymous reviewers for bringing this standard to our attention.

<sup>6</sup>The trade-off between reusing elements from existing vocabularies and the level of precision with which a dataset is represented is a matter of great importance that should always be considered carefully. On the one hand, reusing well-established terms tends to improve the coherence of distributed data and to facilitate interoperability between disparate systems. On the other hand, it also usually incurs a loss of precision wrt. the information one wants to convey. This is particularly true of concepts that are defined in a particular administrative standard, such as for instance the concept of a real property in the Norwegian cadastre. A cadastre commonly includes details of the ownership, the tenure, the precise location, the dimensions, the cultivations if rural, and the value of individual parcels of land. Usually, the concept of a real property varies from one code of law to the next, for instance wrt. cadastral divisions. In order to correctly convey the legal context of a record, these terms need to be represented faithfully, which usually favours custom-made vocabularies over preexisting ones (cf. Section 5).

## Faithful Representation

Making the mereological structure of the Yellow List explicit, requires *adding* information beyond that which is stated in the list itself. There are also natural points at which the data could be joined with external sources to enhance the overall information value of the dataset (in conformity with the fourth of the Linked Data principles), in effect adding even more content. Consider for instance then column **Borough** in Table 1. The column gives a numerical code for the borough in Oslo within which the structure in question is located. Each of Oslo’s boroughs are recorded in DBpedia<sup>7</sup>, which is a database of structured content extracted from Wikipedia and served on the Web as RDF. DBpedia is often regarded as the Linked Data server *par excellence* and a very good resource to link to. Now, for every number in the **Borough** column in the Yellow List, there corresponds a borough which is in turn identified by a URL in DBpedia. For instance the number fourteen corresponds to the borough of Nordstrand, which is `<http://dbpedia.org/resource/Nordstrand>`. As mentioned in Section 2, Linked Data is supposed to be hyperdata—not only *on* the Web, but *in* the Web, as part of its sprawling Web-like fabric. It is therefore a good idea to exploit such links whenever the opportunity presents itself, for instance by assigning the URL `<http://dbpedia.org/resource/Nordstrand>` as the `hvor:bydel` (i.e. the borough) of a building whenever the number in the column **Borough** is 14.

All this raises a very serious concern: how do we know that our additions and transformations do not corrupt the information content of the dataset? How do we maintain the primary nature of the data in the face of the change of representation and gestalt? If measured by the distance from the output of the naive algorithm, our changes are substantial: the naive algorithm produces a fan-like shape such as that of Fig. 1, where the  $s$  denotes a culturally

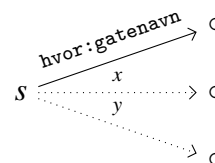


Figure 1: The “naive” algorithm.

valuable building, and each arrow corresponds to an attribute in the Yellow List of that building. For instance, `hvor:gatenavn` (eng: street name) corresponds to column 2 in Table 1. Our representation, in contrast, produces an RDF graph with the general shape given in Fig. 2, where  $p$  denotes the real property of which  $s$  is a part, and the dotted arrow labelled  $z$  may point to an element that is new to the Yellow List. To be sure, any graph of the form of Fig. 1 is a sub-graph of the corresponding graph of the form of Fig. 2, so no information is lost. However we do not have any guarantee that the added structure extends the graph *conservatively*, that is, that it does not contradict or otherwise distort the information in the list. There lurks a general problem behind this, namely the problem how to keep the information content of a dataset unchanged—or only harmlessly changed—whilst changing the *representation* of the data. This problem seems to have gone largely unnoticed by the Linked Data community so far.

In a separate paper [9] we claim that the central notions involved can be made precise using homomorphisms between RDF graphs. A homomorphism of an RDF graph  $G$  to RDF graph  $H$  is a function from vertices and edges in  $G$  to vertices and edges in  $H$  which is such that if  $(a, p, b) \in G$  then  $(h(a), h(p), h(b)) \in H$ . RDF homomorphisms, as homomorphisms, have the property that they simulate the source (say  $G$ ) in the target (say  $H$ ). That is, for every transition or edge between vertices of  $G$  there is a matching transition between vertices of  $H$ . Thus, the homomorphism reflects the structure of the

<sup>7</sup><http://dbpedia.org/>

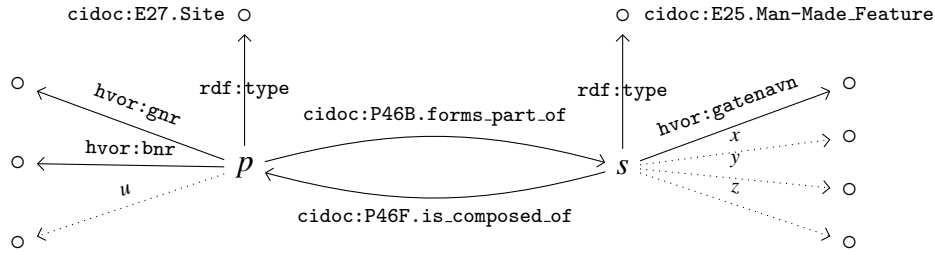


Figure 2: The final structure of list items in RDF.

source in the target. It is well-known, however, that homomorphisms are very liberal wrt. the precise form of a simulation. Indeed, every undirected binary graph has a homomorphism onto a single reflexive point, and it is easy to show that every RDF graph has a homomorphism onto any pair of inversely connected triples. Needless to say, such an RDF graph cannot always be said to capture the information content another graph in any sense we can make of the term ‘information content’. What is needed is a homomorphism that also reflects the structure of the target back into the source, making sure that the simulation is *reciprocal* in the sense of satisfying one of the following three conditions for an RDF homomorphism  $h : G \rightarrow H$ :

- Strong:**  $\langle a, h(p), b \rangle \in H \Rightarrow \langle a, p, b \rangle \in G$
- Liberal:**  $\langle a, h(p), h(b) \rangle \in H$  or  $\langle h(a), h(p), b \rangle \in H \Rightarrow \langle a, p, b \rangle \in G$
- Weak:**  $\langle h(a), h(p), h(b) \rangle \in H \Rightarrow \langle a, p, b \rangle \in G$

These restrictions, or *bounds* as we called them, place requirements on a homomorphism  $h$  with decreasing degrees of strictness, that reflect the structure of the target back into the source. Strong and liberal boundedness each entails weak boundedness, so all three criteria are subsumed by the latter notion, which may be given the following intuitive justification: let  $h$  be a weakly bounded homomorphism from an RDF graph  $G$  to RDF graph  $H$ . Call  $h(a)$  the *representative of  $a$  in  $H$* , where  $a$  is an element of  $G$ . Then boundedness allows representatives of vertices to be related by representatives of edges, but representatives of vertices may not be related by representatives of edges *in new ways*. Thus, boundedness is perfectly faithful to that part of the information that is phrased exclusively in terms of representatives of data items from the original source. Stated differently, the original data is not interrelated in any way in the target graph if it was not so related in the source. It is in this sense that bounded RDF homomorphisms are conservative.

The bounds may be used to exercise detailed and differentiated control over the vocabulary that is involved in a transformation of one RDF graph to another. More specifically, different predicates may be restricted in different ways depending on the intended interpretation of those predicates. *Strong boundedness* is suitable for that part of a dataset to which one would wish to remain absolutely faithful, typically the domain-specific information that is collected and managed by the issuer of the dataset (e.g. the relation between protected buildings and UTM coordinates in the Yellow List). *Liberal boundedness*, on the other hand, is suitable for situations where you would want to *merge* domain-specific knowledge from two different sources whilst keeping the information from each of the sources unchanged. It is more forgiving than strong boundedness in the sense that it allows a relation to grow as long as every added pair relates new objects only (e.g. new buildings with new geographical coordinates added to the Yellow List). Finally,

*weak boundedness* would typically be applied to vocabulary elements that are most aptly considered a part of the *logical* or general-purpose vocabulary. For instance, applied to `rdf:type`, it allows types to be added to source elements as long as those types are not already used in the source. In other words, weak boundedness allows additional typing as long as the added types are fresh (e.g. adding the type `cidoc:E27.Site` to the URLs that correspond to the cadastre codes).

We have implemented this theory in a Web application<sup>8</sup> which can be used to validate that a change of representation is conservative in the requisite sense. Mapper Dan, as it is tentatively called, takes two RDF graphs as input, lets the user specify which bounds to apply, and checks whether there is a map under the given bound between the two graphs. In the cases where conservativeness is violated Mapper Dan offers guidance as to how to adjust the restrictions or the data. Our RDF representation of the Yellow List passes all tests, whence we conclude that it does not distort the original information.

## 4 Displaying and Visualising the Data

Our RDF version of the Yellow List is published on the Web in the form of a SPARQL endpoint (more specifically, a Joseki<sup>9</sup> triple store). An endpoint, also called an RDF server, is a service that accepts and processes SPARQL queries and returns results in different formats (depending on certain supplied parameters). SPARQL itself is both an RDF query language [8] and a protocol [5]. The protocol defines the means of conveying SPARQL queries from clients to SPARQL endpoints over HTTP. The fundamental capabilities of a SPARQL endpoint, as compared to a conventional relational database, is thus the ability to serve data over the Web without the use of a language specific programming API. Our triple store has been fitted with a browsable frontend and content negotiator that makes it fully LOD-compliant. The portal page at <http://sws.ifi.uio.no/project/gulliste/> provides a jumping-off point.

To enable the user to explore the data in an intuitive manner, we wrote a small Web application<sup>10</sup> that based on the UTM coordinates in the Yellow List plots the results of SPARQL queries onto a Google Map. Admittedly, this is not a particularly novel idea in itself. Nevertheless, one aspect of this work might deserve mention in the context of the present paper, namely the use of what we chose to call *significant SPARQL variables*, as a general interface between a SPARQL endpoint and the Google Chart Tools. All Google charts display data from a common code structure called the Google Chart Tools `DataTable`, which is essentially a multidimensional array in JavaScript Object Notation (JSON). Each type of chart places certain requirements on the form of the table. For instance Google Maps requires that the table contains the latitude and longitude values followed by the name of the entities that are to be plotted in that order. A map representation of those entities is then obtained by making an asynchronous HTTP request to the corresponding Google service passing along the table.

Now, SPARQL endpoints usually have the capability to return the results of SPARQL queries in the form of a JSON table—this is true of Joseki in particular. This welcome cohesiveness harbours the possibility to use the SPARQL language as a means to explore an endpoint visually by hooking up the JSON result sets with the appropriate Google service. In fact all we need to do is to cast an appropriate subset of the result table to a Google Charts `DataTable` and to indicate which of the columns that are to be plotted. What

---

<sup>8</sup><http://sws.ifi.uio.no/MapperDan/>

<sup>9</sup><http://www.joseki.org/>

<sup>10</sup><http://sws.ifi.uio.no/project/gulliste/plotter/>

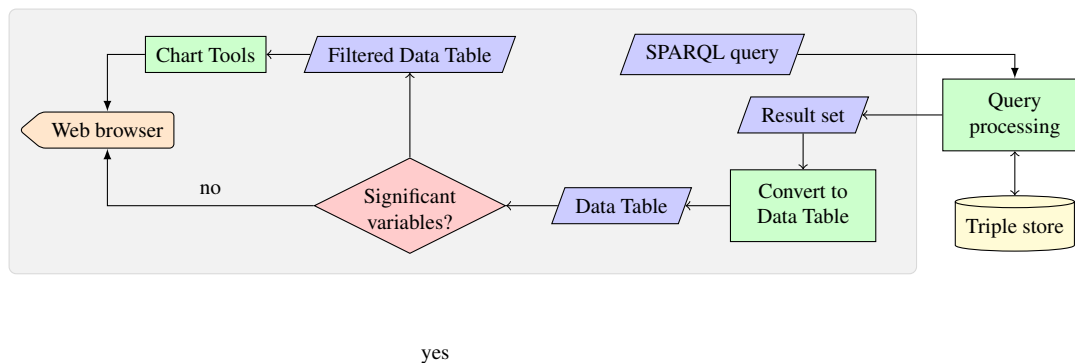


Figure 3: gulliste/plotter web application architecture.

the significant variables do is to tell the application which exact subset this is by naming columns. In the case of our Google Maps-based application, the significant variables are `?lat`, `?lon` and `?name`. If these variables occur in the query, the application will try to plot the result set onto the map by converting the corresponding columns to a `DataTable`, otherwise not. All other variables are treated as usual. Their bindings show up in the display table as well as in the tool-tip that appears when a user hovers over a plotted item on the map. Fig. 3 gives an overview of the system. This architecture is easily adapted to other Google Charts such as line, bar, pie and radar charts, Venn diagrams or scatter plots. All that is required is to alter the names and order of the significant variables

An interesting aspect of the resulting system is that all features of Google Maps are at one's disposal when one explores the data, e.g. the Street View feature. This compares favourably with other applications in the cultural heritage domain in Norway, for instance *Kulturminnesøk*<sup>11</sup> which is maintained by the Norwegian Directorate for Cultural Heritage and based on maps from the Norwegian Mapping and Cadastre Authority. Although this is an impressive piece of work, and of course much more developed than our proof-of-concept application, the maps are rather static and featureless compared with those provided by Google. No doubt they are also better and more detailed. Yet, maximum detail may not always be needed, for instance when plotting buildings onto a map of Oslo. There is also the issue of openness: An RDF-based system is not essentially tied to any one datasource. Our application in particular may equally well be used to explore other sources, or to mashup data from different providers.

## 5 Supplying Meta-data

Behind the term 'meta-data' it is possible to discern at least four very different concepts. *Structural meta-data* documents what may be called the skeletal make up of a dataset considered as a data structure, e.g. the schema and standards it conforms to and the types of information it records. *Provenance meta-data* documents such things as the timeliness of the data, the issuer of the dataset, how the data was changed or processed, and so forth. *Normative meta-data* concerns the conditions under which the data may be used, i.e. licenses and waivers and such things. Finally, what is most aptly called *meta-content* aids the interpretation of individual data items: if a set of observed statistical values, say, is organised along a group of dimensions (e.g. time, area), the meta-content will explain the meaning of those dimensions, and thus of the recorded value itself. Each kind of meta-data is important in facilitating the adoption and reuse of a dataset. We have added

<sup>11</sup><http://www.kulturminnesok.no/>

two sources of meta-data to the Linked Data version of the Yellow List: one is in the form of a VoiD-description of the dataset itself, the other is in the form of the published vocabulary ?Hvor which is used to describe the addresses (cadastral addresses as well as street addresses) of buildings on the Yellow List. Each will be discussed in turn below.

The Vocabulary of Interlinked Datasets (VoiD)<sup>12</sup> is an RDF vocabulary that enables the description of e.g. the links a dataset has to other datasets, its logical partitions (i.e. the subsets), as well as the vocabularies used in a dataset to represent its content. VoiD also allows example URLs to be given as jumping-off points for browsing and querying. As such it is an example of structural meta-data. In the Linked Data version of the Yellow List a VoiD description is attached to the URL <<http://sws.ifi.uio.no/gulliste/dataset>> which we use to refer to the dataset itself. As an example, consider the Turtle snippet below:

```
<http://sws.ifi.uio.no/gulliste/dataset>
  void:subset <http://sws.ifi.uio.no/gulliste/gulliste2DBPedia> .
<http://sws.ifi.uio.no/gulliste/gulliste2DBPedia> rdf:type void:LinkSet ;
  void:linkPredicate hvor:bydel ;
  void:objectsTarget <http://www.dbpedia.org/> ;
  void:subjectsTarget <http://sws.ifi.uio.no/gulliste/dataset> .
```

This description tells us that the dataset identified by <<http://sws.ifi.uio.no/gulliste/dataset>>, that is the Yellow List, has a subset which consists of links into DBpedia. It further tells us that the bridge between these two datasets, that is the link itself, is the hvor:bydel property. VoiD descriptions can be used in many situations, ranging from data discovery to cataloguing and archiving of datasets, and it helps users find the right data for their tasks.

As regards the ?Hvor<sup>13</sup> vocabulary, this is an example of meta-content that adds value and precision to the interpretation of data. It is a *published* vocabulary, and a Linked Data resource in its own right, in the sense that it is served in the form of a content-negotiated set of descriptions. A user may thus retrieve either a machine readable RDF description or a human-readable HTML representation of any of the vocabulary elements of ?Hvor by resolving the URL that acts as a name for it. Note how this adds to the overall explanatory value of the data set: as illustrated by Table 1, the Yellow List Excel spreadsheet contains a column called **House Number** that records house numbers, but it does not say, however, what a house number is, what is to count as a house number, or how house numbers are used. In contrast, the Linked Data version of the list represents the **House Number** attribute with the vocabulary element hvor:husnummer (eng: house number) which abbreviates <<http://vocab.lenka.no/hvor#husnummer>>. This, in contrast, is an interactive, resolvable piece of Web fabric that will produce its own credentials when clicked.<sup>14</sup>

As regards its *rationale*, we found, in the process of working with the Yellow List, that there was a need for a specifically Norwegian vocabulary for representing addresses, places and cadastral units. First, no existing vocabulary is sufficiently granular to represent all the individual components of *any* address it seems. The vCard<sup>15</sup> vocabulary,

---

<sup>12</sup><http://www.w3.org/TR/void/>

<sup>13</sup><http://vocab.lenka.no/hvor>

<sup>14</sup> The ?Hvor vocabulary was developed on Neologism (<http://neologism.deri.ie/>) which is a publishing platform for the Web of Data, with a focus on ease of use and compatibility with Linked Data principles. With Neologism, you can create RDF classes and properties, which are needed to publish on the Web of Data, integrate them into terminological hierarchies, and ontologies, and generate graph visualisations of the vocabulary. Neologism handles the details of content negotiation automatically.

<sup>15</sup><http://www.w3.org/Submission/vcard-rdf/>

for instance, has a property `vcard:street-address`, but it does not have the means for breaking it up into a street name and a house number. Hence vCard alone can not support queries that ask for, say, all houses on the same street, or only for those with even house numbers. Secondly, certain aspects of Norwegian addresses and places more generally, are idiosyncratic for Norwegian administrative standards, e.g. the cadastral unit ‘festenummer’. It would be a misrepresentation of this concept, therefore, to use any existing vocabulary element to denote it. As a response to this, ?Hvor was designed to meet the following two requirements: to be sufficiently expressive to describe Norwegian street addresses and cadastral addresses with maximal and exhaustive granularity, and to conform to Norwegian administrative standards.

## 6 Conclusion

Care needs to be taken when publishing public sector information as Linked Data. There are sometimes good reasons for deviating from the plain tabular structure of the original data repository when building the corresponding RDF graph. However, the need for elaborating the structure of the data must not result in a distortion of the original information content. The notion of a bounded RDF homomorphism gives a way to balance the two. There are few technical obstacles in the way of publishing public sector information as 5-star data. In particular, the technology stack consisting of XLWrap package together with the Neologism gives a suite of tools that makes it relatively easy to publish a system of data and meta-data that conforms to the Linked Data principles.

## References

1. Frank Bannister and Regina Connolly. “The Trouble with Transparency: A Critical Review of Openness in e-Government”. In: *Policy & Internet* 3.1 (2011).
2. Tim Berners-Lee. *Linked Data*. June 2011. URL: <http://www.w3.org/DesignIssues/LinkedData.html>.
3. Tim Berners-Lee. *Putting Government Data Online*. July 2011. URL: <http://www.w3.org/DesignIssues/GovData.html>.
4. Kate Byrne. “Relational Database to RDF Translation in the Cultural Heritage Domain”. 2008. URL: <http://homepages.inf.ed.ac.uk/kbyrne3/docs/rdb2rdfForCH.pdf>.
5. Kendall Grant Clark, Lee Feigenbaum, and Elias Torres. *SPARQL Protocol for RDF*. W3C Recommendation. W3C, 2008. URL: <http://www.w3.org/TR/rdf-sparql-protocol/>.
6. Tom Heath and Christian Bizer. *Linked Data: Evolving the Web into a Global Data Space*. Morgan & Claypool, 2011.
7. Oslo kommune. *Byantikvarens Gule liste*. Aug. 2011. URL: [http://www.byantikvaren.oslo.kommune.no/gul\\_liste/](http://www.byantikvaren.oslo.kommune.no/gul_liste/).
8. Eric Prud’hommeaux and Andy Seaborne. *SPARQL Query Language for RDF*. W3C Recommendation. W3C, 2008. URL: <http://www.w3.org/TR/rdf-sparql-query/>.
9. Audun Stolpe and Martin G. Skjæveland. *Conservative Repurposing of RDF Data*. Poster paper. ISWC, 2011.