Exposing Library Data as Linked Data

Martin Malmsten
Royal Library of Sweden / LIBRIS
martin.malmsten@kb.se

Abstract
An overview of the Linked Data implementation at LIBRIS, the Swedish Union Catalogue, is presented along with a rationale for for building it. A minimal API for exporting bibliographic data and relations from an ILS is also described.

Keywords: RDF; linked data; semantic web; SPARQL; SKOS; MARC; Dublin Core

1. Introduction

The highly structured data contained in library catalogues are usually only available to clients using search and retrieve protocols such as Z39.50 and SRU/W. Exposing library data as Linked Data (Berners-Lee 2006) would let libraries be part of a larger movement aiming to accelerate innovation by opening up data silos and make the data available to agents outside the library sector.

The purpose of this article is to describe the steps necessary to make a library catalogue available as Linked Data as well as a rationale for doing so. The Linked Data implementation at LIBRIS[1], the Swedish Union Catalogue, is used as an example. This paper is written with the view that if you can only do one thing, that one thing should be making your data available; that a "data first" approach is better than "perfect metadata first". It is not an exhaustive overview of every technical choice possible, nor a complete description of how to map MARC21 to RDF. It is rather a step-by-step walkthrough of one implementation already in production.

This document is presented as a complement to the Open Source implementation used at LIBRIS, which is available at http://libris.kb.se/semweb.

2. Why Linked Data matters to libraries

Although making a library catalogue available as Linked Data is mostly a matter of technical details and transformation of data, the implications are more profound. Sharing data, especially with communities outside the library sector, creates an interest in our work. Using tools and techniques that are not unique to libraries lets us communicate easier with professionals in other areas.

Linked Data finally, truly, connects libraries to the web.

[1] LIBRIS - http://libris.kb.se

3. Opening up the catalogue

A minimal ILS-wrapper API
The internal structure of a commercial ILS is often proprietary information that may not be shared as a component of Open Source Software. Therefore, the first step is to create an API that can extract the needed data from the ILS. Since this is the one component that you might need help from your ILS vendor to create and/or set up, it is deliberately kept minimal with the single purpose of getting the data out of the catalogue in its native format.

For the purpose of demonstration and for the rest of this article, the ILS-wrapper "lives" under http://example.org/ils/.

Exporting records
Records are exported in their native MARC-format encoded as MARCXML. The format of the URI is *http://example.org/ils/record/<type>/<id>*.

Example: the URI *http://example.org/ils/record/auth/123* retrieves the authority record with the internal id '123' in MARCXML.

Exporting relations
Relations between records are exported in a text format which is essentially ntriples[2], but with everything but type and id removed. The format of the URI is *http://example.org/ils/record/<type>/<id>/relations.*

Example: the URI *http://example.org/ils/record/auth/123/relations* will retrieve the relations between the authority record with internal id '123' and other records.

```
bib/5515 <http://purl.org/dc/elements/1.1/creator> auth/94541
bib/5516 <http://purl.org/dc/elements/1.1/creator> auth/94541
bib/7058 <http://purl.org/dc/elements/1.1/subject> auth/94541
bib/7080 <http://purl.org/dc/elements/1.1/subject> auth/94541
```

fig. 1 - example output from LIBRIS about auth/94541

Existing software
A number of projects exists to facilitate this type of export, a notable example being the Jangle[3] project. The API described above could, to a certain extent, use Jangle as a provider. This approach had at the time of writing not been fully explored.

4. Identifiers

The first and second rule of Linked Data[4] states that you should "use URIs as names for things" and "use HTTP URIs so people can look up those names".

Choosing the right URLs is both a matter of technology and style. Existing infrastructure, web applications and information policies might also restrict possibilities. In the end, one should strive to make the URLs short and as dependable as possible.

[2] NTriples - http://www.w3.org/2001/sw/RDFCore/ntriples/
[3] Jangle - http://jangle.org
[4] Linked Data - http://www.w3.org/DesignIssues/ LinkedData.html

In LIBRIS the URLs take the form: *http://libris.kb.se/resource/<type>/<id>*. A somewhat more elegant form would have been *http://libris.kb.se/<type>/<id>#resource*, or even http://libris.kb.se/<type shorthand><id>. However, deploying the Linked Data implementation under the */resource* path meant that it could be handled by a different webapp, or even different server, than the rest of the web interface. This was deemed important enough to justify the somewhat more cumbersome form.

Examples - URL candidates:
http://libris.kb.se/resource/bib/1234
http://libris.kb.se/bib/1234#resource
http://libris.kb.se/B1234
http://libris.info/B1234

5. Content negotiation - retrieving records by their URL

Through content negotiation multiple formats can be delivered by the same URL (Sauermann, Cyganiak, 2008). By using the *Accept* header in HTTP the server decides what to do with a request. The server can either redirect the request to another URL using a 303 response, or it can simply deliver the requested format directly.

For example the client issuing the HTTP request in fig. 2 will get the original record in MARCXML, while the HTTP request in fig. 3 will deliver the same record in HTML.

```
GET /resource/bib/12345
Host: libris.kb.se
Accept: application/marcxml+xml
----------
HTTP/1.1 303 See Other
Location: http://libris.kb.se/data/bib/12345?format=application%2Fmarcxml%2Bxml
```

fig. 2 HTTP request for content-type application/marcxml+xml

```
GET /resource/bib/12345
Host: example.org
Accept: text/html
----------
HTTP/1.1 303 See Other
Location: http://libris.kb.se/bib/12345
```

fig. 3 HTTP request for content-type text/html

The URL is the same but the content delivered is different, this means that a semantic web aware agent can ask for the RDF while a web browser will get a page in HTML.

6. Transforming MARC21 to RDF

With a mechanism for record retrieval in place it is time to actually deliver and look at some RDF. Describing a complete mapping from MARC21 to RDF is beyond the scope of this article. Our standpoint has always been that it is better to deliver something now than to dwell on details and wait indefinitely for “perfection”.

Ontologies used are SKOS[5], Dublin Core, Bibliotology and DBPedia. The transformation itself is handled by a web application deployed under the path */data* and is implemented entirely in XSLT (Clarl 1999).

6.1 Name authorities

6.1.1 Personal names

For authority records describing persons, i.e not name/title records or subdivisions, the foaf:Person class is used. Since the inverted form with dates normally found in authority records is mostly used by libraries, straight forms are included as well.

```
<http://libris.kb.se/resource/auth/94541> a foaf:Person ;
        foaf:name "Strindberg, August, 1849-1912", "August Strindberg" ;
        foaf:name "Strindberg, Johan August, 1849-1912" , "Johan August Strindberg" ;
        foaf:name "EstrÏndberg, AgÜst, 1849-1912" , "AgÜst EstrÏndberg" ;
        foaf:bane "Strintmperg, August, 1849-1912" , "August Strintmperg" ;
        dbpedia:dateOfBirth "1849" ;
        dbpedia:dateOfDeath "1912" ;
        owl:sameAs <http://dbpedia.org/resource/August_Strindberg> ;
        rdf:seeAlso <http://sv.wikipedia.org/wiki/August_Strindberg> .

<http://libris.kb.se/resource/bib/5516>    dc:creator    <http://libris.kb.se/resource/auth/94541> .
<http://libris.kb.se/resource/bib/7058>    dc:subject    <http://libris.kb.se/resource/auth/94541> .
...
```

fig. 4 - RDF for August Strindberg (http://libris.kb.se/resource/auth/94541)

6.1.2 Corporate names

For authority records describing organizations the foaf:Organization class is used.

```
<http://libris.kb.se/resource/auth/121848> a <http://xmlns.com/foaf/0.1/Organization> ;
        foaf:name "Kungl. biblioteket" ;
        foaf:name "Sverige. Kungl. biblioteket" ;
        foaf:name "KB" ;
        foaf:name "Kungliga biblioteket" ;
        foaf:name "Kungl. biblioteket - National Library of Sweden" ;
        foaf:name "Bibliotheca regia Stockholmiensis" ;
        foaf:name "National Library of Sweden" ;
        foaf:name "Sveriges nationalbibliotek" ;
        foaf:name "Königl. Bibliothek (Stockholm)" ;
        foaf:name "The Royal Library" ;
        foaf:name "Royal Library" ;
        foaf:name "Biblioteca Real de Suecia" .

<http://libris.kb.se/resource/bib/750221>         dc:subject       <http://libris.kb.se/resource/auth/121848> .
<http://libris.kb.se/resource/bib/1491506>        dc:subject       <http://libris.kb.se/resource/auth/121848> .
...
```

fig. 5 - RDF for Kungl. biblioteket (http://libris.kb.se/resource/auth/121848)

[5] Simple Knowledge Organization System - http://www.w3.org/2004/02/skos/

6.1.3 Mapping table - some examples from LIBRIS

Field	subfields	RDF property	comments
100, 400	a,b,c,d	foaf:name	personal name
100, 400	a,b,c	foaf:name	personal name in straight form
100, 400	d	dbpedia:dateOfBirth dbpedia:dateOfDeath	the d subfield is split on hyphen
110, 410	a,b,c,d	foaf:name	corporate name

6.2 Subject Headings/Thesauri

For authority records describing subject headings and thesauri the class skos:Concept is used. Mapping MARC21 to SKOS is described in great detail by Harper (2006), Harper et al. (2007) and Summers et al. (2008).

```
<http://libris.kb.se/resource/auth/154863> a skos:Concept ;
        skos:inScheme    <http://libris.kb.se/sao> ;
        skos:prefLabel   "Mödrar"@sv ;
        skos:altLabel    "Mothers"@en ;
        skos:broader     "Föräldrar"@sv, <http://libris.kb.se/resource/auth/146439> ;
        skos:narrower    "Ensamstående mödrar"@sv, <http://libris.kb.se/resource/auth/144278> ;
...
        skos:closeMatch  <http://lcsh.info/sh85087526#concept> ;
        skos:closeMatch  <http://id.loc.gov/authorities/sh85087526#concept> ;
        skos:closeMatch  <http://lcsubjects.org/subjects/sh85087526#concept> .

<http://libris.kb.se/resource/bib/819617>    dc:subject    <http://libris.kb.se/resource/auth/154863> .
<http://libris.kb.se/resource/bib/2257802>    dc:subject    <http://libris.kb.se/resource/auth/154863> .
...
```

fig. 6 - RDF for the Swedish Subject Heading “Mödrar” (Mothers)

6.2.1 Mapping table - some examples from LIBRIS

Field	subfields	RDF property	comments
150	all except 0,6,8	skos:prefLabel	
550	all except i, w, 0,6,8	skos:broader	subfield g = ‘w’
550	all except i, w, 0,6,8	skos:narrower	subfield g = ‘h’
750	all except 0,6,8	skos:altLabel	

6.3 Bibliographic records

For bibliographic records classes found in the Bibliontology[6] are used. Mainly elements from Dublin Core are used.

```
<http://libris.kb.se/resource/bib/5060570>    a    bibo:Book ;
        dc:title         The difference engine"@en ;
        dc:creator       "Gibson, William, 1948-", "William Gibson" ;
        dc:creator       "Sterling, Bruce, 1954-", "Bruce Sterling" ;
        dc:type    "text" ;
        dc:publisher    "Vista" ;
        dc:date    "1996" ;
        dc:description    "Originally published: London: Gollancz, 1990"@en ;
        dc:subject       "American fiction" ;
        dc:subject       "Steampunk" ;
        dc:identifier    <URN:ISBN:0575600292> ;
        bibo:isbn10      "0575600292" ;
        dc:creator       <http://libris.kb.se/resource/auth/220040> ;
        dc:creator       <http://libris.kb.se/resource/auth/307779> ;
        dc:subject       <http://libris.kb.se/resource/auth/308073> ;
        dc:subject       <http://libris.kb.se/resource/auth/308074> .
```

6.3.1 Mapping table - some examples from LIBRIS

Field	subfields	RDF property	comments
20	a, z	bibo:isbn10, bibo:isbn13, dc:identifier	
22	a, z	bibo:issn, dc:identifier	
100, 700	a,b,c,d	dc:creator	
245	a,b,f,g,h,k	dc:title	
260	b	dc:publisher	
260	c	dc:date	
600, 610, 611, 630, 650, 653	a,b,c,d,q	dc:subject	

6.4 Relations between authority and bibliographic records

Only two distinct relations exist between authority and bibliographic records in LIBRIS: *dc:subject* and *dc:creator*.

6.5 Relations between authority records

Between authority records for subject headings *skos:broader* and *skos:narrower* are used.

[6] Bibliontology - http://bibliontology.com/

7. SPARQL server

While you do not need a SPARQL (Prud'hommeux and Seaborn, 2008) server of your own to publish Linked Data, it makes it easier for other providers to find resources within and explore your data set. There are a number of triple stores that can handle the amount of data in an average library catalogue. The ESW Wiki[7] keeps an up-to-date list of large data stores.

LIBRIS' SPARQL server[8] currently uses the Sesame 2 Native Store as a triple store. Query performance is good, though loading can take a long time when not done either in RAM or on s solid state drive (SSD). The database is reloaded every week. Loading around 70M triples takes aproximately 50 hours on a normal harddrive, three hours on an SSD and 50 minutes in RAM.

8. Linking to external resources

To further enhance the data set, links to external resources are important. For libraries, links to a published and widely used subject heading system such as LCSH makes it easy for others to jump from one data set to another.

DBPedia using SPARQL

DBPedia[9], the semantic web version of Wikipedia, can be used to locate Wikipedia articles about a certain person or concept, and their DBPedia URIs. Since Wikipedia/DBPedia contains information about peoples occupation, narrowing the scope to just authors is simple. The following example serves as a starting point for such matching, however, accuracy will depend on the granularity and completeness of data in the catalogue.

```
select distinct ?uri where {
  ?uri a <http://dbpedia.org/ontology/Writer> ;
          <http://xmlns.com/foaf/0.1/name> "August Strindberg" .
  ?uri <http://dbpedia.org/property/dateOfBirth> ?date .
  FILTER ( xsd:dateTime(?date) >= xsd:dateTime("1849-01-01T00:00:00")) .
  FILTER(  xsd:dateTime(?date) < xsd:dateTime("1850-01-01T01:00:00")) .
}
```

fig. 7 - SPARQL query for finding the URI for a writer named August Strindberg, born in 1849

If we accept the (somewhat naive) technique above, and since it only returns one result, we can create the following statement.

```
<http://libris.kb.se/resource/auth/94541> rdf:sameAs <http://dbpedia.org/resource/August_Strindberg> .
```

LCSH / lcsubjects.org

If your subject headings or bibliographic records include mappings to LCSH, it makes sense to expose these as links to the *Library of Congress Authorities and Vocabularies service*[10] as well as LCSubjects.org[11]

[7] ESW Wiki - http://esw.w3.org/topic/LargeTripleStores
[8] LIBRIS SPARQL endpoint - http://api.libris.kb.se/sparql
[9] DBPedia - http://dbpedia.org
[10] http://id.loc.gov/authorities/

LC does not at the time of writing have a SPARQL server, LCSubjects.org, however, does. Since the data is essentially the same, one can use the SPARQL server at lcsubjects.org to create links to both. The example in fig. 8 is a SPARQL-query trying to find the subject heading for “Mothers”.

```
select ?uri where {
  ?uri <http://www.w3.org/2004/02/skos/core#prefLabel> “Mothers”@en .
}
```

fig. 8 - SPARQL query for finding a LCSH with the preferred label “Mothers”

The answer to this particular query is *http://lcsubjects.org/subjects/sh85087526#concept* which lets us create the two statements in fig. 9.

```
<http://libris.kb.se/resource/auth/154863> skos:closeMatch <http://id.loc.gov/authorities/sh85087526#concept> .
<http://libris.kb.se/resource/auth/154863> skos:closeMatch <http://lcsubjects.org/subjects/sh85087526#concept> .
```

fig. 9 - skos:closeMatch statements

9. Future development

As Open Source projects, such as Jangle, with active communities mature it seems reasonable to move parts of the implementation described here over to those projects.

Summary

To expose the library data within an ILS as Linked Data the following steps must be taken:

1. Find a way to get the record and relations out of the ILS
2. Choose a URL pattern
3. Map to RDF
4. Implement content negotiation / record delivery

References

Tim Berners-Lee. (2006). Linked data. Retrieved April 12, 2008, from http://www.w3.org/DesignIssues/LinkedData.html.

Leo Sauermann, Richard Cyganiak. (2008). Cool URIs for the Semantic Web. Retrieved April 13, 2008 from http://www.w3.org/TR/cooluris/

Harper, Corey. (2006). Authority control for the semantic web. Encoding Library of Congress subject headings. International Conference on Dublin Core and Metadata Applications, Manzanillo, Mexico. Retrieved June 20, 2008 from http://hdl.handle.net/1794/3268.

[11] http://lcsubjects.org/

Harper, Corey, and Barbara Tillett. (2007). Library of Congress controlled vocabularies and their application to the semantic web. Cataloging and Classification Quarterly, 43(3/4). Retrieved June 20, 2008 from https://scholarsbank.uoregon.edu/dspace/handle/1794/3269.

Ed Summers, Antoine Isaac, Clay Redding, Dan Kreech (2009). LCSH, SKOS and Linked Data. International Conference on Dublin Core and Metadata Applications. Retrieved June 5, 2008 from http://edoc.hu-berlin.de/conferences/dc-2008/summers-ed-25/PDF/summers.pdf

Eric Prud'hommeaux, Andy Seaborn. (2008). SPARQL Query Language for RDF. Retrieved April 12, 2008 from http://www.w3.org/TR/rdf-sparql-query/

James Clark. (1999). XSL Transformations (XSLT). Retrieved April 13, 2008 from http://www.w3.org/TR/xslt