Having Triplets – Holding Cultural Data as RDF

Kate Byrne

School of Informatics
University of Edinburgh
k.byrne@ed.ac.uk

Abstract. Great hopes are cherished for the Semantic Web. It is intended to make linking data as easy as HTML makes linking Web documents. For the cultural heritage world seamless linking of related but independently curated datasets has long been a dream. This paper looks at the practical and theoretical issues involved in converting data to RDF so that it can become part of the Semantic Web. I propose a simple generic design for cultural heritage data, and discuss the options for including published heritage thesauri. Scaleability is a key requirement for curators of large collections, and this work uses the whole of a real archive (the National Monument Record of Scotland) rather than a pilot sample.

Key words: RDF, Semantic Web, relational database, RDB2RDF, cultural heritage, scaleability

1 Introduction

Semantic Web developers in academic departments and specialist companies have been producing data management solutions based around RDF triple stores for a number of years now, but it is still the case that the relational database (RDB) is by far the commonest tool for practical data curation in national and local cultural heritage bodies. In this respect cultural archives are no different from the majority of commercial and government organisations, which still overwhelmingly rely on RDB solutions for their day-to-day operations.[1] There are many pilot projects and demonstrators in cultural organisations around Europe and further afield, but very few large collections are yet available on the Semantic Web. The CultureSampo project [1] in Finland is a notable and exciting exception. One reason for the slow take-up of Semantic Web [2, 3] technology is the effort involved in exposing large data collections as RDF.

Despite its historical popularity, the relational database is by no means ideal for managing cultural data. As well as fixed fields that can be mapped straightforwardly into a relational design, cultural data abounds with notes, free text

[1] One can roughly measure real usage by seeing what IT skills are in most demand. A search with keywords "relational database" and "SQL" produces around 1,000 hits on `http://www.planetrecruit.com/` and 780 at `http://www.computerweekly.com/jobs/`, compared with a single result from the first site and zero from the second for "RDF", "Semantic Web" or "SPARQL".

documents, associated multimedia archive items and so on, which are often awkward to represent in RDB tables. Web-based interfaces for public querying usually have to be restricted to just a small subset of the actual RDB schema, and the rest of the data, typically including all the free text, is inaccessible to queries (though of course it can be included in the results). Linking separately curated data – museum finds, site-based archives, library catalogues, and so forth – is technically difficult with RDBs, though it has long been an ideal to aim for.

For these reasons and others, many cultural organisations are looking to the Semantic Web (or Data Web) as a way forward. An approach that is being widely examined is to convert the diverse semi-structured data into one simple format, an RDF graph (Resource Description Framework, specified in [4]) which is made up of a network of interconnecting triples of the form *subjectNode–propertyArc–objectNode* (often called SPO triples). Triples become connected whenever the object of one is the subject of another. However, only "resources" (which have unique URIs, as each represents a specific resource on the Semantic Web) can be the subject of triples. Conventionally, resource nodes are represented using ovals. If the object of a triple is not a resource but a literal (such as a string value), it is represented as a rectangle. (See Fig. 1 for an example.) Literal values cannot be the subject of new triples and so are always at the edge of the graph. RDF data can be held in a triple store which, at its simplest, is just a table with three columns, for "Subject", "Property" (or "Predicate") and "Object".

Every "fact" from the database can be held as a triple, such as "*Skara_Brae–hasLocation–Orkney*". In principle, these facts can be extracted from the free text as well as from the structured RDB fields (as described in [5]). If the same subject or object node appears in another RDB graph held by a different organisation, the two datasets are automatically linked – this is what underlies the Semantic Web. For example, a site-based archive like the RCAHMS (The Royal Commission on the Ancient and Historical Monuments of Scotland) dataset used in this research[2] would be connected to a museum database of archaeological finds, if they both maintained an RDF graph containing the same *Skara_Brae* node.

This paper examines the process of RDB to RDF conversion in Sect. 2, and suggests a generic mechanism for cultural material. Section 3 deals with the inclusion of published thesauri in the RDF graph. Scaleability is discussed in Sect. 4. This a key requirement for curators of large collections, and it was felt important to use the entire RCAHMS dataset, with all the practical implementation issues that entails.

2 RDB to RDF Conversion

Automatic conversion software is available for translating RDB data into RDF but there are drawbacks to the basic procedure, as discussed below. For archive organisations it is worth inserting a manual design step to reap benefits later.

[2] The data is a snapshot of the entire National Monument Record of Scotland (NMRS), maintained by RCAHMS – see `http://www.rcahms.gov.uk/`.

The gains will be partly from producing a smaller, more streamlined structure, as explained in Sect. 2.2, and partly from using a schema specifically designed for cultural heritage data, as described in Sect. 2.3.

2.1 The Basic RDB2RDF Process

At its simplest, the RDB2RDF conversion process is straightforward and is illustrated in Fig. 1, based on a cut-down version of the RCAHMS database `SITE` table. Starting with the first row of the table, one takes each cell in turn and generates a triple, using the column name as the property arc and the cell value as the object node. There is no obvious label for the subject, but we can see that each triple from the first row should have the same subject, as each represents a different property of the same thing: *X–siteNo–"1"*, *X–name–"Dirleton Castle"* and so on. RDF has a special type of node called a "bnode" (or blank node) for this purpose: we have no obvious label to give it but we know that it represents an instance of the `SITE` entity.

The result is that each database tuple (or table row) becomes a cluster of triples grouped around a bnode whose type (implemented as an `rdf:type` property) is a class (`rdfs:Class`) corresponding to the table name. Hence this conversion is sometimes called "Table to Class" transformation.

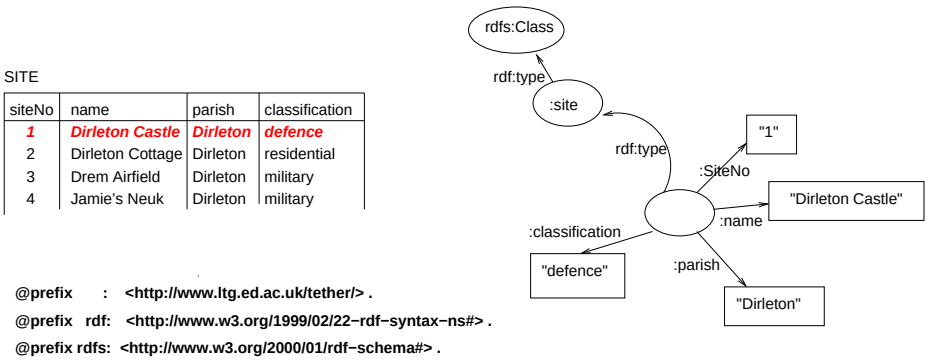

Fig. 1. Translation of a relational table tuple to RDF, at its simplest.

There is a growing number of automatic conversion tools that work on these general principles, such as D2RQ [6], Dartgrid [7], Dan Connolly's `dbview` program,[3] R_2O [8] and DB2OWL [9] with more or less scope for customisation by the user in each case. The mapping to RDF does not necessarily have to be instantiated. Instead a virtual graph may be constructed, saving space and ensuring data currency, at the expense of increased processing cost at query time.

[3] `http://dig.csail.mit.edu/2006/dbview/dbview.py`

SquirrelRDF and R2D2 (a sister project of D2RQ) are examples of tools that construct a graph "view" of a relational database, in effect allowing SPARQL queries to be run instead of SQL ones. D2RQ and Virtuoso [10] give the user the choice of an instantiated graph or a virtual one. Whether the RDF graph is physically instantiated or not, the RDB2RDF principles are the same.

As yet there are no definitive guidelines on best practice in RDB2RDF conversion, though the W3C has recently set up a group to work on it.[4] On the generation and serving of "cool" URIs for RDF there is already plenty of guidance available (see [11], [12]).

2.2 Shortcomings of the Basic Process

The procedure outlined above corresponds to the approach proposed by Berners-Lee [13], where database attributes are translated to RDF property (or predicate) arcs with full provenance information about their database source. One immediately wonders whether the relational database is expected to remain the "master" copy. Surely if the Semantic Web comes into its own this will not be the case: each RDF predicate will represent a specific binary relation that can exist between resources anywhere, irrespective of the vagaries of a particular relational schema (which may, after all, change). This is just one of the places where the "Table as Class; Column as Predicate" method seems lacking.

A step-by-step discussion of the problems with the basic RDB2RDF conversion process is given in [14], and a checklist is offered for the RDF designer embarking on this process. In brief, the issues examined are:

1. The use of bnodes and so-called "duck typing" (defining a resource by its properties rather than giving it a URI), which inhibits linking of subgraphs. Instead my suggested design uses a URI generated from the relevant RDB primary key.
2. The generation of suitable URIs. A fundamental point about URIs in RDF is that they must be distinct when and *only* when they refer to distinct resources. Much of the benefit of using RDF is completely thrown away if we lose the chance to link graphs on shared nodes, as will happen if URIs encode the database location of values.
3. The use of RESTful URIs. Encoding the class hierarchy as part of the URI structure promotes web service access using REST (Representational State Transfer – see [15, 16]).
4. The handling of many-to-many and one-to-many relational joins. The basic procedure introduces redundant "key to key" triples which, when the linking keys have no intrinsic properties of their own, can be pruned. For the RCAHMS data this saves 2.8 million triples.
5. Dealing with concatenated keys in the RDB. Where these exist, it is more efficient to generate new surrogate keys for the RDF graph.

[4] `http://www.w3.org/2005/Incubator/rdb2rdf/`

6. Avoiding duplication of schema metadata. The basic procedure will repeat provenance information in the subject node, predicate arc and object node (unless the object is a literal) of a given triple.
7. Preventing the sterilisation of the graph through use of literals. It is suggested that resource URIs should be generated for database values, as otherwise there can be no further RDF connections from them.
8. Distinguishing metadata from true content. Surrogate keys (typically running numbers) are frequently used as primary keys in RDB tables. In RDF one can take advantage of "serendipitous linking" of identical data values (such as "Auchtermuchty" occurring in a `Site.Parish` field and in a `FoundObject.Location` field), but care must be taken to avoid such merging with surrogate values (such as `site123` and `find123`).
9. The handling of null fields. In general these can be entirely excised from the RDF graph, but schema knowledge is required, as it is possible in RDBs to assign meaning to the absence of a value. Relational databases typically operate with the Closed World Assumption (what is not stated as true is false), whereas in the Semantic Web world the Open World Assumption pertains (what is not stated is unknown).[5]
10. Denormalisation and handling of coded values. Good RDB design often uses coded values with corresponding lookup tables. In an RDF graph such codes translate to redundant nodes, which can be eliminated by linking directly to the value node.
11. Transposing arcs and nodes. It is sometimes helpful to think of the RDF triple as *"subject–verb–object"*, so the graph arcs are verb phrases and the nodes are nouns. My design transposes all of the predicates of Fig. 1 into classes or "nouns", as they are in fact "things" with a hierarchical class structure (a parish is a place, for example).

Figure 2 illustrates some of the points just made, using the same small sample of data as Fig. 1. The bnode has been replaced by a URI based on the surrogate RDB key, the literals are now resources with RESTful URIs and labels, and the database columns have become RDF classes in a class hierarchy. For this tiny sample the number of triples has increased, but over the whole dataset many millions of unnecessary triples are saved through using shareable nodes.

These issues apply to any RDB to RDF conversion process. Just as relational schema design is a skilled task requiring thought and experience, so too the RDF schema should be carefully planned to avoid redundancy and minimise complexity. A solid and flexible design will last many years and amply repay the initial effort required. As with RDBs, an overcomplicated schema makes for inflexible query interfaces, whereas one aim of the Semantic Web is to permit ad hoc exploration of data by remote software agents. In the future there may be sufficient computing power for reasoning software, based around Description Logic say, to cope with very complex schemas, but reasoning over multi-million triple stores is barely feasible for cash-strapped cultural bodies as yet.

[5] See `http://wiki.esi.ac.uk/The_Closed_World_of_Databases_Meets_the_Open_World_of_the_Semantic_Web` for discussion of this issue.

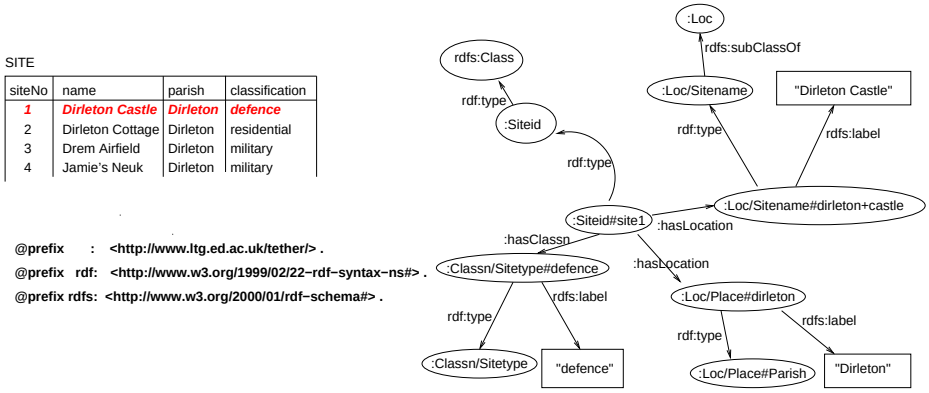

Fig. 2. Part of the *Tether* design.

2.3 A Schema for Cultural Data

In addition to the general RDF transformation issues just described, there are design points that apply particularly to the cultural heritage domain. The conversion of an archive to RDF for publication through the Semantic Web is an opportunity to standardise the design and hence maximise interoperability with other cultural datasets. Using the RCAHMS dataset as a test-bed, a simple generic design has been devised, based on the "Who? What? Where? When?" or "People, Places, Things and Events" models often used in heritage data management.

The design is instantiated in a triple store named *Tether*,[6] derived from a snapshot of the entire RCAHMS database of around 250,000 historical sites with some 1.5 million associated archive items and bibliographic material. This translates to an RDF store containing more than 21 million triples. Two parallel physical implementations have been created, using the Jena[7] and AllegroGraph[8] systems.

The ontology has 11 main classes: for *sites*, *archive items*, *collections* of archive items, *bibliographic items*, *bibliographic references*, *agents* (people and organisations), agent *roles*, geographical *locations*, *temporal references*, *events* and *classification terms*. There are three other general classes at the top level: for *identifiers* (metadata that has acquired meaning through long use such as, in the RCAHMS dataset, the "NMRS number"), *flags* or indicator fields, and *descriptive notes*. There is one other top level class, for *linear features* (such as walls, roads etc.), that is specific to the RCAHMS dataset. All classes but this one are used throughout the heritage domain and collectively the set provides coverage of the basic data needs of that domain. (To test this, it was applied

[6] "Tether" is a dialect word for three, used in the north of England for counting sheep.
[7] `http://jena.sourceforge.net/`
[8] `http://agraph.franz.com/allegrograph/`

without alteration to a set of museum data.) To capture the specific characteristics of the RCAHMS database some 45 further classes are needed, hooked into the generic upper ontology of 15 classes in a structure that is no more than three levels deep. This contrasts with the many hundreds of classes that would result from a purely automatic translation from RDB tables to classes.

The basic framework of relationships between the classes needs only nine different predicates, such as *siteArc* to link instances of the Site class to Archive items, and so on. This framework corresponds directly to a simplified database entity relationship diagram illustrated in Fig. 3.

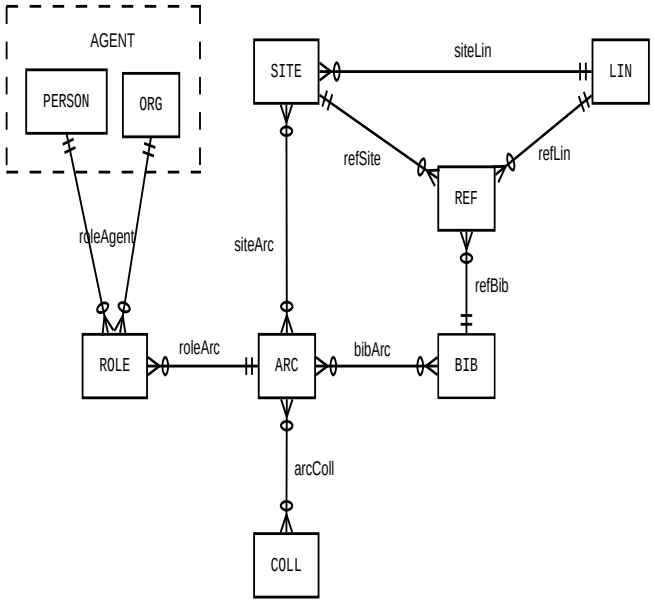

Fig. 3. Relations between the main ontology classes of the *Tether* RDF schema.

A further eight predicates were chosen for the relationships required to express the database contents: *:hasAgent*, *:hasAgentRole*, *:hasClassn* (links to a graph derived from domain ontologies, as described in Sect. 3), *:hasDesc* (for descriptive notes), *:hasLocation*, *:hasPeriod*, *:hasId* (for identifiers like the NMRS number mentioned above), and *:hasFlag*. Three more were needed to cater for the text relations that will be added later: *:hasEvent*, *:hasPatient*, *:partOf*. In addition, some standard predicates from published vocabularies for RDF, RDFS and OWL were used, e.g. for typing and subclass relations. The tiny size of the predicate set is probably the single most significant contrast with the basic translation process outlined in Sect. 2.1, where the predicate set directly reflects the RDB attributes and therefore will usually be enormous.

Through a combination of the measures described in Sect. 2.2, the rigorous limiting of schema size, and exploitation of the fact that there is typically a great deal of duplication of data values in cultural heritage data, the final *Tether* RDF set, at around 21 million triples, was only one tenth the size that a simple "`rows x columns`" calculation on the RDB would predict.

3 Thesaurus Conversion

The principal thesauri used by RCAHMS are for Monument and Object types. (There is also a Maritime Thesaurus, not so far converted for *Tether*.) The RDB structure employs the standard (if slighly awkward to use) way of expressing a hierarchy in relational tables, using a technique known as "tree walking" over a set of pair relations between an item and the item above it. (A tree is of course a particular example of a graph structure, so RDF allows a much more natural representation of a hierarchical thesaurus.)

The RCAHMS thesauri are based on MIDAS Heritage [17], the UK historic environment data standard, maintained by English Heritage. In turn, a mapping[9] is available from MIDAS to the CIDOC-CRM [18]. For *Tether*, each thesaurus was converted to a named graph (see [19]) held within the triple store. The archive data is linked to the thesauri at specific nodes, for example instances of *:Classn/Sitetype* link to the Thesaurus of Monument Types (see Fig. 2).

Being small and carefully structured, the thesauri were straightforward to convert to RDF compared with the much less disciplined archive content. The procedure is readily automated using the principles described in Sect. 2.2, especially the point about de-referencing codes to their values (in this case to the thesaurus terms). The Monument Thesaurus produces a graph with fewer than 17,000 triples, and the Object Thesaurus is under 4,000. Nevertheless there are a number of issues to be considered when converting a thesaurus to RDF.

3.1 Thesaurus or Ontology?

A thesaurus usually contains three core relation types: Hierarchical ("broader term", "narrower term"), Associative ("related term") and Equivalence ("preferred term", "non-preferred term") – see, for example, [20]. There are also typically Scope Notes providing a short text glossary of each term.

The obvious vocabulary to use when converting thesauri to the Semantic Web is SKOS,[10] the Simple Knowledge Organisation System, which was designed for that purpose – see [21] for a clear exposition of the framework and some of the issues involved in using it.

Hyvönnen et al. [22] argue that SKOS mirrors thesaurus relations too closely, including the anomalies sometimes present. For example, the "broader term" (BT) relation may not be transitive, and may confuse instances and classes:

[9] See `http://cidoc.ics.forth.gr/docs/MIDAS_mapping_notes.doc` and `http://cidoc.ics.forth.gr/docs/midas_map.xls`.

[10] `http://www.w3.org/2004/02/skos/`

their example, from what must be a rather casually constructed thesaurus, is "Halley's Comet → BT → comet → BT → solar system". They prefer a strict ontology structure, susceptible to logic processing, and using *type*, *subClassOf* and *partOf* relations as appropriate.

The FinnOnto approach is attractively clean and logical, but nevertheless the SKOS framework was chosen for *Tether*. The latest version of SKOS allows the user, in effect, to choose whether to make the *broader* relation transitive or not. Also, the fact is that the structure to convert is a thesaurus, not a strict ontology, and reflecting that seems an advantage. Finally, since the objective is to facilitate links with other cultural datasets, it makes sense to use the vocabulary favoured in that domain.

3.2 Schema Elements

The core element of SKOS is the *Concept* (defined in the June 2008 version of the schema as "a unit of thought"!) rather than the *term*, which is taken to be a label for a Concept. This has implications for logic processing and also, which concerns *Tether* more nearly, means that only the Hierarchical and Associative thesaurus relationships can be expressed: *skos:broader* (with inverse *narrower*) and *skos:related* link *skos:Concept*s to one another much as one would expect, but the Equivalence relation can only be expressed through *skos:prefLabel* and *skos:altLabel*, which link Concepts to literals. This makes sense if one wishes to distinguish a concept (in the ordinary English sense) from the name of a concept (its label), but it means that there is no way of including a non-preferred term as a resource, which limits its potential in the RDF graph. The *Tether* design therefore, whilst using *prefLabel* and *altLabel* in the established manner, also includes a local Equivalence relation between Concepts. (In acknowledgement of the slight irregularity, it is called *prefTerm* rather than *prefConcept* – though of course this string is merely a label with no intrinsic significance.)

The SKOS documentation acknowledges the need for relations between labels, and recommends using a specially introduced *LabelRelation* class and associated properties, in what seems a rather convoluted arrangement. This way of looking at the problem stems directly from the SKOS notion of what a *Concept* is, and seems slightly at cross-purposes if one has thesaurus terms in mind. After some thought, the pragmatic solution just described was adopted, which in no way precludes a *LabelRelation* approach alongside.

The RCAHMS thesauri happen to include an upward pointing link from each term to its "top term", i.e. to the thesaurus elements that *skos:hasTopConcept* points downwards (from the ConceptScheme) towards. Rather than lose these occasionally useful links another local predicate was created, named *topTerm*.

Apart from these two non-standard elements, the rest of the schema is simply a subset of SKOS. The two classes used are *skos:ConceptScheme* and *skos:Concept*, and the seven predicates needed are *skos:broader*, *skos:related*, *skos:prefLabel*, *skos:altLabel*, *skos:scopeNote*, *skos:inScheme* and (though the thesaurus top terms are slightly anomalous) *skos:hasTopConcept*. The top terms (such as "Civil", "Domestic") are closer to what SKOS calls "node labels" and

models with a *skos:Collection* class. However, this introduces a number of complications, such as removing them from the hierarchy, that seemed simply unnecessary for *Tether*.

3.3 Linking to the Archive Content

Clearly the converted thesaurus is only useful insofar as it can be connected to the graph derived from the RDB content. Taking the Monument Thesaurus as an example, the connection points are the *:Classn/Sitetype* nodes, such as *:Classn/Sitetype#chambered+cairn*. How should these be tied to SKOS Concepts?

Several options were considered:

1. Each *:Classn/Sitetype* could be tied to the corresponding concept by a *skos:exactMatch* predicate, which is designed for linking matching Concepts without asserting that they are the same node. (For practical loading purposes, one would need to decide whether the link points this way (RDB node to thesaurus) or the other.)
2. The nodes could be logically merged, using an *owl:sameAs* predicate.
3. A thesaurus node could be generated within the RDB graph, replacing the existing *Sitetype* instances (involving re-engineering of the RDB2RDF process).
4. The thesaurus could be built around the existing *Sitetype* nodes where these are already present. Where there is no existing matching node, the thesaurus node would still use the same form, such as *:Classn/Sitetype#new+site+type* where "New site type" is a putative term from the thesaurus that doesn't actually occur in the data content.

The last-mentioned option was chosen, in line with the principle listed in Sect. 2.2, that coinciding concepts should receive identical URIs whenever possible. The *exactMatch* and *sameAs* variants would work perfectly well in the presence of a reasoner, but for practical purposes this is infeasible. They require an extra link in the SPARQL query and an extra piece of schema knowledge – both of which are to be avoided if possible.

4 Scaleability and Practical Issues

The data used for this work[11] is a complete snapshot of a real archive, rather than a subset of relatively "clean" material such as it would be sensible to use for a pilot or demonstrator. The collection has been assembled over ten decades (2008 is RCAHMS' centenary year) by field survey and research and it has characteristics that are probably familiar to cultural data curators everywhere.

[11] The data has been made available by RCAHMS for research purposes. It is in various formats, the source being an Oracle database. A corpus of 1500 documents taken from text fields has been annotated for named entities and for binary relations.

Recording methods have changed over the years so the format varies and many database fields are sparsely populated. Typographical errors are common, partly because of lack of data entry controls in the past, and partly because the bulk of the data was captured by Optical Character Recognition almost twenty years ago. There are a great many text "notes" fields, of greatly varying length.

The database has been moved from platform to platform as technology developed: twenty years ago an electronic archive of this size had to be held on a big central mainframe. Character set changes and the untraceable bugs that are bound to occur over such a period mean that spurious non-printable characters crop up at random in the data; they must all be dealt with in the RDF conversion. Most fields will contain plenty of valid characters that still need to be encoded to appear in URIs. Dealing with embedded paragraph breaks in "notes" fields is particularly tedious, especially when accessing an Oracle database using SQL via JDBC from a Java program, which involves nesting the different escape mechanisms for special characters within one another. Although the aim was to translate the data unchanged, some minimal data cleaning had to be done, like removing multiple trailing spaces and linefeeds from text fields. In a practical conversion project it is worth noting that such apparently trivial character coding issues can take a disproportionate amount of time.

Small RDF graphs can be held as XML documents and loaded into memory for processing, but this is not practical for a large archive. The graph must be held in persistent storage in a specially designed store that permits small sets of triples to be extracted by queries without attempting to build the entire graph in memory. Formal benchmarking of triple stores was not one of the objectives of this work, but two separate systems, Jena and AllegroGraph, were used in parallel, in order to provide some feel for performance considerations. These two platforms were chosen after a survey of those available, chiefly for their design features, robustness, good documentation and usable APIs. When doing repeated loading and formatting experiments with a multi-million triple set, AllegroGraph's remarkably good loading and indexing performance is attractive. Loading the 21 million triples into AllegroGraph takes just 33 minutes and indexing a further 20 minutes on a standard PC. Loading and indexing the same set in Jena takes almost 15 hours, though it does include a de-duplication check that AllegroGraph omits. (On the same machine, exporting the RDB data from Oracle as RDF triples takes 40 minutes.)

Query times vary greatly, depending on the complexity of the query, and experiments so far indicate disappointing performance compared with SQL against the RDB. For SPARQL queries against an AllegroGraph store of this size, Franz Inc. recommend a 64-bit platform, but even on a 64-bit machine with dual AMD Opteron processors and 16 GB of RAM, quite simple queries take several minutes. A series of further query experiments is planned.

5 Summary and Conclusion

Issues arise in handling complete datasets that may be missed when picked subsets are used. This paper highlights practical conversion issues that may absorb a surprising amount of time when real archives, built over decades, are processed.

Using a complete set also allows one to measure the worth of steps taken to prune redundant nodes from the RDF graph. Over a small set of data these steps may actually increase the graph size and complexity, but following the series of techniques described in Sect. 2.2 reduces the RCAHMS graph size tenfold. The notional figure produced by multiplying the number of RDB columns used by the number of rows is over 234 million, but the finished *Tether* graph size is actually under 22 million. This makes a colossal difference to performance in terms of storage, loading and query times. The larger the orginal archive the more important is a compact graph.

The inclusion of standard thesauri is important for the cultural domain, and a conversion process based around the SKOS framework has been described. The Monument and Object thesauri used here are based directly on the MIDAS standard, which in turn is linked to the CIDOC-CRM. Thus standardised query terms could be applied across any Semantic Web systems.

The *Tether* schema has been designed to be generic for cultural heritage data, whilst being as small and simple as possible. This makes it potentially much easier to build query interfaces uniting separately curated triple stores in different archive organisations. In essence, the same data value occurring in two graphs that use this schema (such as *:Loc/Place#skara+brae* for "Skara Brae") would automatically link the two datasets.

References

[1] Eero Hyvönen, Tuukka Ruotsalo, Thomas Häggström, Mirva Salminen, Miikka Junnila, Mikko Virkkilä, Mikko Haaramo, Eetu Mäkelä, Tomi Kauppinen, and Kim Viljanen. CultureSampo–Finnish Culture on the Semantic Web: The Vision and First Results. In K. Robering, editor, *Information Technology for the Virtual Museum*, pages 25–36, Berlin, November 2007. LIT Verlag.

[2] Tim Berners-Lee, James Hendler, and Ora Lassila. The Semantic Web. *Scientific American*, 284:34–43, 2001.

[3] Lee Feigenbaum, Ivan Herman, Tonya Hongsermeier, Eric Neumann, and Susie Stephens. The Semantic Web in Action. *Scientific American*, 297:90–97, December 2007.

[4] Resource Description Framework (RDF): Concepts and Abstract Syntax. W3C Recommendation, 10 February 2004.

[5] Kate Byrne. Tethering cultural data with RDF. In *Proceedings of the Jena User Conference 2006 (JUC2006)*, Bristol, UK, May 2006.

[6] Christian Bizer and Richard Cyganiak. D2RQ - Lessons Learned. In *Proceedings of the W3C Workshop on RDF Access to Relational Databases*, Cambridge, MA, USA, October 2007. W3C. Position paper.

[7] Zhaohui Wu, Huajun Chen, Heng Wang, Yimin Wang, Yuxin Mao, Jinmin Tang, and Cunyin Zhou. Dartgrid: a Semantic Web Toolkit for Integrating Heterogeneous Relational Databases. In *Semantic Web Challenge at 4th International Semantic Web Conference (ISWC 2006)*, Athens, USA, November 2006.

[8] Jesús Barrasa, Óscar Corcho, and Asunción Gómez-Pérez. R_2O, an extensible and semantically based database-to-ontology mapping language. In *Proceedings of the 2nd Workshop on Semantic Web and Databases*, Toronto, Canada, August 2004.

[9] Nadine Cullot, Raji Ghawi, and Kokou Yétongnon. DB2OWL: A Tool for Automatic Database-to-Ontology Mapping. In *Proceedings of 15th Italian Symposium on Advanced Database Systems (SEBD 2007)*, pages 491–494, , 17–20 2007., Torre Canne, Italy, June 2007.

[10] Virtuoso: A Superplatform for Merger Driven Data Integration and Business Process Services. Internet White Paper, 2006.

[11] Cool URIs for the Semantic Web. W3C Working Draft, 17 December 2007. Contributors: Danny Ayers and Max Völkel.

[12] Best practice recipes for publishing RDF vocabularies. W3C Working Draft, 23 January 2008.

[13] Tim Berners-Lee. Relational Databases on the Semantic Web. Internet note, 2006. v 1.22 2006/02/01 (originally published September 1998).

[14] Kate Byrne. Relational Database to RDF Translation in the Cultural Heritage Domain. Internet, May 2008. Based on chapter of PhD thesis.

[15] Roy Thomas Fielding. *Architectural Styles and the Design of Network-based Software Architectures*. Doctoral dissertation, University of California, Irvine, 2000.

[16] Leonard Richardson and Sam Ruby. *RESTful Web Services*. O'Reilly, 2007.

[17] Edmund Lee, editor. *MIDAS Heritage — a data standard for the historic environment*. Forum for Information Standards in Heritage (FISH), 2007.

[18] Nick Crofts, Martin Doerr, and Tony Gill. The CIDOC conceptual reference model: A standard for communicating cultural contents. *Cultivate Interactive*, (Issue 9), Febrary 2003.

[19] Jeremy J. Carroll, Christian Bizer, Pat Hayes, and Patrick Stickler. Named Graphs, Provenance and Trust. In *Proceedings of the 14th International World Wide Web Conference (WWW2005)*, pages 613–622, Chiba, Japan, May 2005. IW3C2.

[20] Douglas Tudhope and Ceri Binding. A Case Study of a Faceted Approach to Knowledge Organisation and Retrieval in the Cultural Heritage Sector. *Digicult – Thematic Issues*, (6: Resource Discovery Technologies for the Heritage Sector):28–33, June 2004.

[21] Mark van Assem, Véronique Malaisé, Alistair Miles, and Guus Schrieber. A Method to Convert Thesauri to SKOS. In *Proceedings of the Third European Semantic Web Conference (ESWC'06)*, volume 4011 of *Lecture Notes in Computer Science*, pages 95–109, Budva, Montenegro, June 2006. Springer.

[22] Eero Hyvönen, Kim Viljanen, Eetu Mäkelä, Tomi Kauppinen, Tuukka Ruotsalo, Onni Valkeapää, Katri Seppälä, Osma Suominen, Olli Alm, Robin Lindroos, Teppo Känsälä, Riikka Henriksson, Matias Frosterus, Jouni Tuominen, Reetta Sinkkilä, and Jussi Kurki. Elements of a National Semantic Web Infrastructure - Case Study Finland on the Semantic Web (Invited paper). In *Proceedings of the First International Semantic Computing Conference (IEEE ICSC 2007)*, Irvine, California, September 2007. IEEE.