

SWAD–Europe Deliverable 12.4.1: Large Scale Resource Discovery and Presentation Demonstrator

Project name:

Semantic Web Advanced Development for Europe (SWAD-Europe)

Project Number:

IST-2001-34732

Workpackage name:

12.4 Large Scale Resource Discovery and Presentation Demonstrator

Workpackage description:

<http://www.w3.org/2001/sw/Europe/plan/workpackages/live/esw-wp-12.4.html>

Deliverable title:

12.4.1 Large Scale Resource Discovery and Presentation Demonstrator

URI:

http://www.w3.org/2001/sw/Europe/reports/large_scale_demo/

Author:

Dave Beckett

Abstract:

This report describes the development of a demonstrator for showing the results of searching large scale RDF, presenting the results of the search in contexts of the original sources along with metadata about the transaction using the Redland RDF Application Framework and the Redland Contexts, designed for this purpose.

STATUS:

COMPLETED 2004-06-04.

Contents

1. [Introduction](#)
2. [Choice of demonstrator content](#)
3. [Contexts](#)
4. [Applications](#)
5. [Aggregation Demonstrator](#)
6. [Related applications](#)

1. Introduction

The aims of this demonstrator were, as stated in the workpackage outline:

- To demonstrate how simple resource discovery can be done over large scale and rich Semantic Web data
- To evaluate ways to present rich Semantic Web data in familiar ways while preserving the integrity of the data and its provenance
- To consider approaches to displaying Semantic Web data in different environments: for example: web browsers, low bandwidth devices

and to do this by means of a demonstrator building on existing work and tools that were developed for SWAD-Europe and elsewhere. The main tool that would be used as the basis of this demonstrator was the open source *Redland RDF Application Framework* ([\[REDLAND-PAPER\]](#), [\[REDLAND-SITE\]](#)) developed at ILRT since June 2000 and further updated for this demonstrator.

The demonstrator required a source of semantic web data to run on that was large enough and rich to show the goals clearly. Large enough meant more than just a few hundred files created by some individual, rather it had to be thousands or more files that were developed by a variety of people using different software for a variety of purposes. This semantic web data was in the web metadata format, the Resource Description Framework (RDF) but as a general format, the purpose of the RDF had to vary and the content remain unconstrained. This open world approach of semantic web data mirrors how the web works with HTML - there is no pre-coordination of the terms (words on the web page) or how the links are made between the terms (links inside the HTML). So a complex web of RDF information was needed and rich enough in terms of diversity of terms that were used to ensure the demonstrator was not dealing with only a small and constrained problem.

Optimising a particular application for a small set of terms in a known space is not a difficult research problem and can be solved by standard database and information management techniques. When the space of information is as large as the web -- since any RDF could point to any URI -- and with any term -- also URIs -- means that the static database schema approach is not so appropriate for the representation of this dynamic data. SWAD-Europe has investigated this problem in detail in other workpackages, mostly [\[SWADE-WP7\]](#) and [\[SWADE-WP10\]](#). The

detailed reports delivered that are relevant were looking at the software [SWADE-D10-1], the database schemas [SWADE-D10-2], the query technology [SWADE-D7-2] and APIs [SWADE-D7-1] and the practical considerations after from key community members [SWADE-D3-11].

This demonstrator was not concentrating on looking at the issues of APIs or RDF query languages but at demonstrating the storage and retrieval aspects of making a system that allowed simple resource discovery across the semantic web. The resource discovery, or searching aspect was also not the main focus so showing a full text, natural language or other sophisticated search system for the semantic web was not a main goal. These systems can be applied to the free text available by scraping any large corpus of documents and scaling them to large data sets was not the appropriate work here. The novel aspect of semantic web search for large documents considered here is related to knowing about once the semantic web has been made visible in one graph, to know where the parts of that information came from and to handle updating them, since each of the original sources could change the data at any time (no co-ordination) or make claims that would need tracing by an end application. The latter is one foundation requirements for applications that perform "web of trust" calculations on the semantic web, although that requires other technologies such as digital signatures of canonical document formats and tracing authority in trust hierarchies such as via trusted third parties. Such trust ideas are being investigated in greater detail in the deliverables under [SWADE-WP11].

2. Choice of demonstrator content

The most diverse kind of data, like for HTML webpages, is that created by people. The semantic web of data that is being generated by the activity of people can mostly be seen in two areas -- the weblogging world, by the syndicated changes feeds as output of the weblogs in *RDF Site Summary (RSS) 1.0 (RSS 1.0)* [RSS10] format and the *Friend of a Friend (FOAF) project* [FOAF] which encourages people to write description of materials that interest them and relate them to other content.

The RSS 1.0 format is a stable and mature format based on a profile of RDF/XML and has been deployed widely in popular weblogging applications such as *Movable Type* [MOVABLE-TYPE] by *Six Apart Ltd.* which has a very large deployed base. In addition to content from weblogs, RSS 1.0 is also produced from content producers such as news organisations. There are other XML-only RSS formats which have sketchy semantics, poor standardisation which keep changing that can also be used but can only be used with a lot of data scraping and cleanup and are not appropriate to rely on. RSS content is by its nature continually being updated as it is primarily a syndication format, and the feeds or files are updated with new items very frequently, often many times per day. This kind of rapid change requires management of the system that retrieves the content that is typically called a crawler and aggregator, in terms of how often the feeds are fetched, respecting the site's policy about how often this can be done, retrying when errors occur and general management issues that arise from dealing with network failure and retrieval. This is very similar to the type of application that is called a web crawler, reading HTML web pages, following links and then retrieving more pages, indexing the words on the pages to provide a search system. Web crawlers are today large commercial operations in order to crawl large (billions) of web pages. Solving such network issues are not the primary goal of this demonstrator which is aimed more at showing how the results of gathering the data and searching can be shown having some original sources that can be preserved.

FOAF data is more descriptive and personal and does not tend to change as often as RSS, although it is updated as new information is recorded, so does need refreshing. This is more like a web page in that sense in that it is not expected that a FOAF file is changing daily like RSS feeds typically can. FOAF content is generated by a variety of tools and methods similar to RSS, and the content is diverse since it is personal to the user, who may or may not record many different related FOAF vocabularies. There are web based sites that help construction of FOAF files using web forms such as *FOAF-a-Matic* [FOAF-A-MATIC] and the resulting RDF file can be saved, as well as desktop applications in Java that do the same on the client side.

There are several sites that produce, generator and/or consume FOAF data. Some which provide logins and accounts make available FOAF descriptions of the people including *Ecademy* [ECADEMY]. FOAF is also provided by Six Apart's hosted Movable Type site service *Typepad* [TYPEPAD] which has substantial installations worldwide. *PeopleAggregator* [PEOPLEAGG] both collects and allows the creation and hosting of FOAF files on the site. The *Plink* [PLINK] service allows the browsing and viewing of FOAF files and is a pure aggregator and viewer rather than hosting.

Most of the above sites that collect and present FOAF data do not track the source information or allow the user to find out where the parts of the information that made the particular web page came from - the "why?" button. If several pages talk about the same resource and describe it in different ways, people need to be able to distinguish where this came from. RSS also somewhat can use this but the metadata there is very much more constrained, so the only common property is that of citation, or mentioning some URI. In the FOAF data, the terms can be used in describing people, web pages, pictures, geographical locations, events, physical documents and a whole lot more, so the descriptions and the way that they can be used are not constrained. The semantic web way of extending the terms that are being used for description is via RDF schemas and ontologies which allow description of the properties and classes being used, and with OWL, further constraints on how they can be used. This is an open world approach to describing the content rather than saying all that is allowed, describe what particular terms mean in different schema documents and let the applications decide when or whether they use that information to find out if they can use the terms.

The choice was made to make a search and viewing demonstrator initially over FOAF data since it was rich data, there were thousands of different files to read and it had the required problem of considering where the aggregated information came from which needed presenting in the demonstrator result set. This contextualisation of the results is one of the key aspects of the demonstrator technology.

3. Contexts

Figure 1 shows an RDF triple, the element of the semantic web. The three parts of the triple are generally called *nodes* or *nodes* and *arcs* when the predicate part needs to be distinguished from the others. The triples are used in sets to form RDF graphs, such as shown in Figure 2, below, which is often drawn as a directed labelled graph when the triples are joined by drawing the same nodes together.

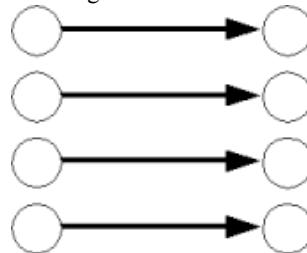


Figure 2: An RDF Graph - a set of RDF triples

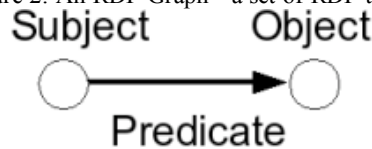


Figure 1: An RDF Triple

It is common to want to merge graphs from multiple sources by joining multiple graphs (sets). When this is done, where there are duplicate triples in some sets, the triple's source provenance is lost in the merge. The RDF graph doesn't preserve this source information, so it needs to be provided by applications. RDF also does not provide scoping for sub-graphs or ways to quote triples (in the graph but not part of the conjunction of triples semantics). RDF does have reification but it is more of a syntax abbreviation than a good way to model such information.

One approach to this is to add a 4th item, 5th, 6th... item to the RDF "triples", changing the RDF model into quads (4-ary tuples) or larger items. This would mean ripping up all the existing RDF software and designing a new non-RDF model that was based about 4-ary items and also losing all interoperability with triples. Other proposals have added more items such as identity for graphs or sub-graphs or for individual triples. These remain mostly research ideas rather than anything immediately compatible with RDF.

There have previously been proposals to make RDF triples have or be part of *contexts* - a general term which has been used in different ways. In *Notation 3* [REF-NOTATION3], Berners-Lee defines an *N3-context* as relating two *N3-formulae* in a triple like this:

```
{s1} foo:prop {s2}
```

where each of *s1* and *s2* are some set of triples, an RDF graph. These are nested graphs so the triples in *s1* and *s2* are not in the outer RDF graph that contains the triple above.

There are also contexts used in the sense of R.V. Guha [GUHA-CONTEXTS], the creator of MCF at Apple/Netscape which greatly influenced the creation of RDF in 1998 of which he worked on, and RDF Schema of which he was the co-editor in 1999. These were left out of the original RDF design probably for the goal of simplicity and not available for current RDF modeling.

It was given that breaking RDF by expanding the triples was not interoperable and adding subgraphs/quoting or Guha-contexts was also going to break things and be hard to reason about; this might impact inferencing systems on top of Redland. A simpler approach was needed.

When adding triples to a graph from multiple sources, it would be easy to allow an identifier to be given at that time so the method would be something like:

```
graph.add(triple, identifier)
```

If this could somehow be associated with that particular addition of the triple, it might be possible to use that later. This identifier, which is a Redland *node* (URI or literal or blank node) as already used as the parts of the triples, is called the *context node* and is the identifier for Redland contexts.

Similarly, it would also be natural to have

```
graph.addTriples(tripleGenerator, contextNode)
```

when adding a set of triples generated by a single operation something such as parsing RDF/XML, a query on a graph etc. In Redland, a sequence of triples is represented by a Redland *stream* and delivers the triples incrementally.

Redland triples (*statements* in the API) are just 3 redland *nodes* (URIs or literals or blank nodes) and are independent of any graph or store. Redland nodes similarly are not attached to any other object. It is in particular, very common and natural to use the same nodes and triples across multiple graphs so tying them to one would have been rather meaningless as well as hard to implement. This means that adding to them with a model/storage specific context would have been a mistake and counter to the overall Redland design.

Redland graphs and stores were always bag of triples (before it was clearly defined that RDF graphs were a set). A search of the form:

```
stream=model.find_statement(s, p, o)
```

```
statement=stream.get_next() # returning the first matching triple
```

looks for a matching triple to the triple (s,p,o) where each of the items can optionally be a wild card, matching anything. This returns a stream of answer triples (s1,p1,o1), (s2,p2,o2), ... that can then be processed.

This concept of a stream has proved useful for contexts since the triples returned represented different *statings* of the same (s,p,o), potentially in different contexts; with different associated context nodes, although not visible from the API. There is a similar mechanism for returning a sequence of nodes as a result, called an Redland *iterator* such as this query to find all the object nodes for a particular subject and predicate:

```
iterator=model.get_targets(s,p)
```

The context information is most useful when such queries are made over the graph so that the application is dealing with a particular triple in a query over some graph. It is at that point that the context (in the graph) of that triple or node becomes important. This is represented in RDF when either a

```
stream=model.find_statement(s, p, o)
iterator=model.get_targets(s, p)
```

or some other API call is done that returns a sequence of answers - sequences of triples (statements) with streams, sequences of nodes with iterators.

Thus, the decision was made to add contexts such that they could be accessed by streams and iterators without disturbing the triple basis of RDF and Redland's core.

In Redland with contexts, each triple can now be marked as having multiple context nodes such as **C1**, **C2**. These are regular Redland nodes so can appear in triples themselves. [Figure 3](#) shows four triples that appear in the subject of two triples. The context nodes are assigned when the triples are added to the redland graph, but otherwise they are no different from any other graph node.

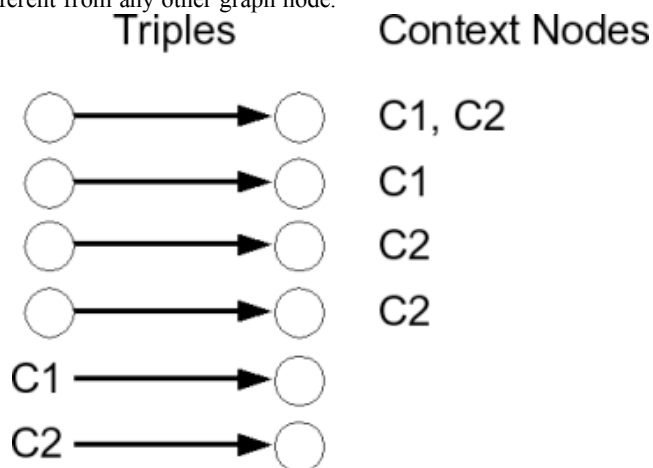


Figure 3: A Redland RDF Graph with context nodes C1 and C2

This meant adding new methods to the stream and iterator classes to get the current context node for the triple or node being returned from the query:

```
stream=model.find_statement(s, p, o)
statement=stream.current()
context_node=stream.context()
```

and new utility methods of the model class to list all the context nodes in a graph, list all the triples with a given context and to bulk add/delete triples associated with a context - a common use case.

Redland contexts can be used for several different methods of recording context depending on how the context nodes are associated with the triples, which is application specific, and is done at the time the triples are added to the graph. The uses could include the following, although this is not an exhaustive list:

Enable true graph merging / updating / demerging

Identify the subgraphs (sets of triples from particular sources) with context nodes.

Triple Identity

Add each triple with a different context node. RDF's model does not assign identity to triples. There is reification also which might be used with this approach.

Triple Provenance

Use the context node as the subject of other triples about the triple that is returned.

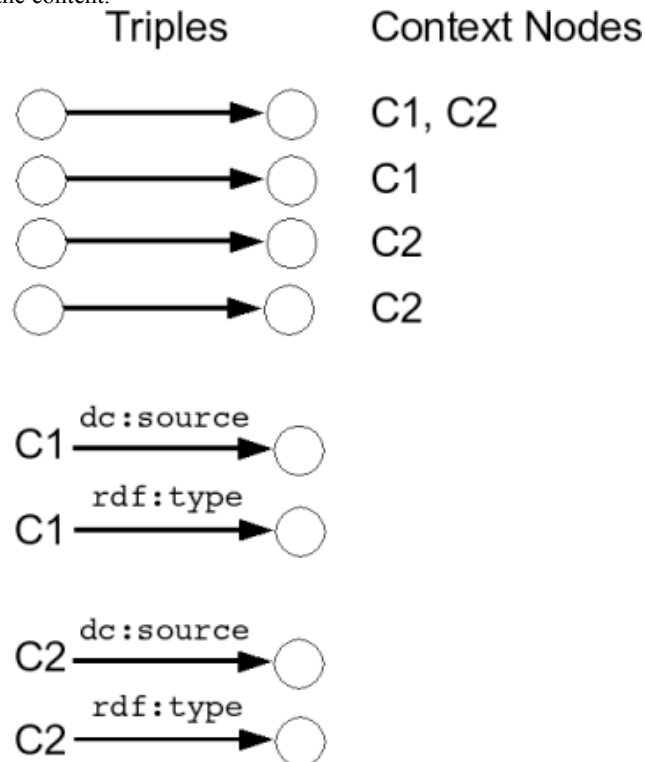
Subgraphs

Similar to the merging approach but consider the RDF graph to be a set of graphs and manipulate them as such.

In this demonstrator the graph merging is need, along with tracking the provenance of particular triples, after receiving them as the result of user searches.

4. Aggregation Demonstrator

The demonstrator was implemented using Redland and the MySQL/BDB store along with python server-side web interface and a python crawler/aggregator for reading the files. In the demonstrator it was chosen to model the context as a blank node in the graph using the Dublin Core source property (`dc:source`) pointing to the source content URI and giving that node a type allowing it to be distinguished in the graph from other blank nodes. The retrieval action information was also recorded on the context blank node including the content last modified time, and last retrieval data of the content.



A Redland RDF Graph with descriptions of two contexts C1 and C2

Importing the data - The first step of importing the data involved writing a simple crawler and page cacher for retrieving the FOAF files. The source of the locations FOAF files was from other FOAF files and lists that allow people to "bootstrap" into the semantic web of FOAF. Once these URIs were retrieved, the standard method of finding out more, the `rdfs:seeAlso` references were also followed to retrieve further URLs. The resulting pages were then parsed as RDF/XML (FOAF) and added to the graph as individual contexts using the modeling style discussed above. The *Raptor RDF parser* [RAPTOR] which is separate module of Redland was used to do the parsing and recover from the inevitable bad XML and RDF/XML files that are found in real-life data. This parser was updated and developed during this and other SWAD-Europe work.

Storing the data - The demonstrator was designed to rely on Redland contexts and these were previously implemented during the development, for the internal Sleepycat/Berkley DB-based triple store, and the in-memory triple store. The BDB store had previously been seen to scale to 2-3 million of triples at the cost of very large disk space and slower retrieval. Redland was designed to be modular in storage and thus alternate stores for larger scale and faster access were developed and investigated:

1. A store based on the *AKTors project* [AKTORS] Triplestore [AKTORS-3STORE], which uses MySQL as the low level store. The AKTors store provides only a restricted interface for contextual information and does not provide the full flexibility of the redland contexts.
2. A MySQL store was developed by a third party open source developer, Morten Frederiksen (MFD Consult, Copenhagen, Denmark) which has full contexts support.

Searching the data - The full-text support from MySQL 3 was used to enable keyword searching of the stored FOAF data, returning the matching triples. This is a restricted and simple search functionality that has a hardcoded list of stop words (in version 3) and a hardcoded minimum word size of 4 (ditto). It is much faster than the equivalent SQL query using `LIKE` and allows the demonstration to get the triple results in a short time, scaling to large numbers of triples.

Presenting the data - The demonstrator returns the triples that match the entered keyword as a table of triples. Each of these triples has link to the contexts (there can be several) that the triple originally was found in. This context can be followed and the contextual node information found; the context node records the source URI, the page last modified and last accessed dates and via the Redland API, allows access to the triples in the context. The entire list of triples for each context is then displayed.

The demonstrator - The demonstrator is available at: <http://www.redland.opensource.ac.uk/contexts>

Figure 4 shows the results of using the demonstrator to list all contexts in the graph (showing just the first 20 here):

Count	Context	Context Info
-------	---------	--------------

1	triples in _:r1074705611r3	triples about _:r1074705611r3
2	triples in _:r1074705611r13	triples about _:r1074705611r13
3	triples in _:r1074705611r21	triples about _:r1074705611r21
4	triples in _:r1074705611r28	triples about _:r1074705611r28
5	triples in _:r1074705611r30	triples about _:r1074705611r30
6	triples in _:r1074705611r60	triples about _:r1074705611r60
7	triples in _:r1074705611r73	triples about _:r1074705611r73
8	triples in _:r1074705611r87	triples about _:r1074705611r87
9	triples in _:r1074705611r97	triples about _:r1074705611r97
10	triples in _:r1074705611r99	triples about _:r1074705611r99
11	triples in _:r1074711178r14	triples about _:r1074711178r14
12	triples in _:r1074711178r31	triples about _:r1074711178r31
13	triples in _:r1074711178r34	triples about _:r1074711178r34
14	triples in _:r1074711178r43	triples about _:r1074711178r43
15	triples in _:r1074711178r46	triples about _:r1074711178r46
16	triples in _:r1074711178r61	triples about _:r1074711178r61
17	triples in _:r1074711178r70	triples about _:r1074711178r70
18	triples in _:r1074711178r73	triples about _:r1074711178r73
19	triples in _:r1074711178r83	triples about _:r1074711178r83
20	triples in _:r1074711178r92	triples about _:r1074711178r92

Figure 4: Listing all Contexts in the Redland Contexts Demonstrator

This was performed via the demonstrator command `list-all-contexts` which used the Redland model method `get_contexts` to return an iterator of the contexts known. The second column lists all the triples with that context node, the third column lists all the triples which have the context node as a subject. Picking one such as context node `_:r1074705611r3` the results are shown in [Figure 5](#) and [Figure 6](#) for both of these queries.

Count	Subject	Predicate	Object	Context
1	(r1074719425r5)	[rdf.type]	[foaf:Person]	_:r1074705611r3
2	(r1074719425r5)	[foaf:mbox_sha1sum]	4b66faad55b917c8c367bd0691b0e2bfd0809ed	_:r1074705611r3
3	(r1074719425r5)	[foaf:weblog]	[http://11eleven.typepad.com/]	_:r1074705611r3
4	(r1074719425r5)	[foaf:nick]	friggint	_:r1074705611r3
5	(r1074719425r5)	[http://purl.org/vocab/bio/0.1/olb]	I serve the tek.	_:r1074705611r3
6	(r1074719425r5)	[foaf:img]	[http://11eleven.typepad.com/eye.jpg]	_:r1074705611r3
7	(r1074719425r6)	[rdf.type]	[foaf:Person]	_:r1074705611r3
8	(r1074719425r6)	[foaf:name]	NetFlix Movies	_:r1074705611r3
9	(r1074719425r6)	[foaf:homepage]	[http://images.amazon.com/images/P/B00005QW5X.01.LZZZZZZZ.jpg]	_:r1074705611r3
10	(r1074719425r5)	[foaf:knows]	(r1074719425r6)	_:r1074705611r3

Found 10 triples

Figure 5: Listing all triples in context `_:r1074705611r3`

Count	Subject	Predicate	Object
1	(r1074705611r3)	[http://purl.org/dc/terms/modified]	1071805823
2	(r1074705611r3)	[dc:source]	[http://11eleven.typepad.com/foaf.rdf]
3	(r1074705611r3)	[rdf.type]	[http://www.w3.org/2001/sw/Europe/200401/bot/terms#PageFetch]
4	(r1074705611r3)	[http://www.w3.org/2001/sw/Europe/200401/bot/terms#etag]	"a4644a8-326-3d591dc0"
5	(r1074705611r3)	[http://www.w3.org/2001/sw/Europe/200401/bot/terms#lastAccessed]	0

Found 5 triples

Figure 6: Listing all triples about context `_:r1074705611r3`

[Figure 6](#) shows the use of redland contexts to store information about the document and the transaction of retrieving that document - the HTTP GET - as RDF triples. These are linked to the triples that resulted by the Redland context, but are not mixed with them.

Related applications

Applications of Redland contexts include Edd Dumbill's [FOAFBot](#) as described in *Support online communities with FOAF* [\[FOAF-SUPPORT\]](#). The new feature is reported in the updated FOAFBot made with Redland and python in *Tracking provenance of RDF data* [\[FOAF-PROVENANCE\]](#) and *Finding friends with XML and RDF* [\[FOAF-FRIENDS\]](#). Morten Frederiksen also uses Redland with PHP and the MySQL store to provide a FOAF browser called [FOAF-EXPLORER](#) [\[FOAF-EXPLORER\]](#).

References

[AKTORS]

[Advanced Knowledge Technologies \(AKT\) Project](#). URL <http://www.aktors.org/>

[AKTORS-3STORE]

[Aktors 3store](#). URL <http://www.aktors.org/technologies/3store/>

[ECADEMY]

[Ecademy](#) trust network for business people to share and exchanging knowledge, contacts and leads.

[FOAF]

[Friend of a Friend \(FOAF\) project](#)

[FOAF-A-MATIC]

[FOAF-a-matic](#), Leigh Dodds

[FOAF-EXPLORER]

[FoAF Explorer](#), Morten Frederiksen, [MFD Consult](#), Copenhagen, Denmark

[FOAF-FRIENDS]

[Finding friends with XML and RDF](#), Edd Dumbill, IBM developerWorks

[FOAF-PROVENANCE]

[Tracking provenance of RDF data](#), Edd Dumbill, IBM developerWorks

[FOAF-SUPPORT]

[Support online communities with FOAF](#), Edd Dumbill, IBM developerWorks

[GUHA-CONTEXTS]

Contexts: A Formalization and Some Applications, R.V. Guha, PhD thesis, 1995, Stanford University
([PostScript](#), 146pp).

[MOVABLE-TYPE]

[Movable Type](#) web content publishing system, [Six Apart Ltd.](#)

[NOTATION3]

[Notation 3 -- Ideas about Web architecture](#), Tim Berners-Lee, 1998-2001

[PEOPLEAGG]

[PeopleAggregator](#) FOAF aggregator

[PLINK]

[plink.org](#) people link FOAF viewer (beta testing, at 2004-01-21), Dom Ramsey

[REDLAND-PAPER]

[The Design and Implementation of the Redland RDF Application Framework](#), Dave Beckett, ILRT, University of Bristol, Proceedings of WWW10, 2001-05-02

[REDLAND-SITE]

[Redland RDF Application Framework](#), Dave Beckett, ILRT, University of Bristol. URL
<<http://www.redland.opensource.ac.uk/>>

[RAPTOR]

[Raptor RDF Application Framework](#), Dave Beckett, ILRT, University of Bristol. URL
<<http://www.redland.opensource.ac.uk/raptor/>>

[RSS10]

[RDF Site Summary \(RSS\) 1.0](#), Beged-Dov et. al

[SWADE-D3-11]

[SWAD-Europe D3.11 Developer Workshop Report 4 - Workshop on Semantic Web Storage and Retrieval](#), Dave Beckett, ILRT, University of Bristol, 2004-01-12

[SWADE-WP7]

[SWAD-Europe WP 7: Databases, Query, API, Interfaces](#)

[SWADE-D7-1]

[SWAD-Europe D7.1 RDF API requirements and comparison](#), Jan Grant, ILRT, University of Bristol, 2003-02-27

[SWADE-D7-2]

[SWAD-Europe D7.2 Databases, Query, API, Interfaces: report on Query languages](#), Libby Miller, ILRT, University of Bristol, 2003-04-01

[SWADE-WP10]

[SWAD-Europe WP10: Tools for Semantic Web Scalability and Storage](#)

[SWADE-D10-1]

[SWAD-Europe D10.1 Scalability and Storage: Survey of Free Software / Open Source RDF storage systems](#), Dave Beckett, ILRT, University of Bristol, 2003-02-17.

[SWADE-D10-2]

[SWAD-Europe D10.2 Mapping data from RDBMS](#), Dave Beckett and Jan Grant, ILRT, University of Bristol, 2003-02-18.

[SWADE-WP11]

[SWAD-Europe WP11: Distributed trust systems](#)

[TYPEPAD]

[Type PAD](#) personal publishing (weblogging) service, Six Apart Ltd.