SWAD-Europe Deliverable 10.1: Scalability and Storage: Survey of Free Software / Open Source RDF storage systems

Project name:
Semantic Web Advanced Development for Europe (SWAD-Europe)
Project Number:
IST-2001-34732
Workpackage name:
10. Tools for Semantic Web Scalability and Storage
Workpackage description:
☞ http://www.w3.org/2001/sw/Europe/plan/workpackages/live/esw-wp-10.html
Deliverable title:
Semantic Web Scalability and Storage: Survey of Free Software / Open Source RDF storage systems
URI:
☞ http://www.w3.org/2001/sw/Europe/reports/rdf_scalable_storage_report
Authors:
☞Dave Beckett, ILRT, University of Bristol, UK.
Abstract:
This report surveys the state of semantic web storage for RDF / triple data using existing free software tools. It takes a practical approach by targeting the work to the needs of developers, answering frequently asked questions related to this subject. The report first reviews previous work in surveying semantic web data, schema and triple stores, then gives an overview of the major systems with their feature set and maturity and then uses that information to provide a set of FAQs with answers related to storing semantic web data.
Status:
Final reportpublished 2002-07-31. Updates will be made over the lifetime of the SWAD-Europe project as tools change, new tools emerge, FAQs are asked and answers need improving. The datasets section is presently mostly pointers to other places; it is planned that sample datasets will be gathered and made available publically for developers.
Comments on this document are welcome and should be sent to ☞Dave Beckett or to the ☞public-esw@w3.org list. An archive of this list is available at ☞http://lists.w3.org/Archives/Public/public-esw/

Contents

1 Introduction

This report is part of ☞SWAD-Europe ☞Work package 10: Tools for Scalability and Storage and addresses the scope, features and purpose of developer tools for providing semantic web data storage using existing systems that are licensed as ☞Free Software or ☞Open Source.

For those in a hurry: go straight to the ☞FAQs section

Scope - Semantic web data is designed to be machine readable, in comparison to the general web which mostly consists of markup or other formats intended to be rendered as text, graphics and perceived by people. Semantic web data needs to carry around the identification of its terms along with it, pointing to the definition or relationship to other terms. This is done by giving the terms URI identifiers and creating descriptions of these terms in what are sometimes called Schemas, Vocabularies, Ontologies or Data Dictionaries.

Storage systems that require fixed schemas may be unable to handle general data such as that from a semantic web where the terms are not known in advance. There are no restrictions on what URIs can be used or what they represent and before the data is entered into the system, no optimisation can typically be done on the system schema

without prior knowledge. This is analogous to not knowing what words will be on a web page before reading it.

Terminology - Semantic web data is usually in a simple form of description of particular resources with properties of them and the values of these properties. These three parts form a triple of (resource identifier, property, value) and semantic web storage generally is described in terms of storing such triples, sometimes called statements. These triples are the semantic web datum for RDF.

This data corresponds to the RDF☞[RDFMS] graph model which describes the graph as a set of triples. In this document a graph is used for a set of triples. A system that stores such triples is usually called a triple-store, RDF database or database. The latter term is very general and will not be used in this report, especially since it may be confused with relational database systems (RDBMSes).

Scalability - Triples are intended to be used for resource description on the web similar to the way in which tags in HTML are used for describing markup in web pages, triples are used for statements as part a description of a resource, so stores for triples must work on the scale of the web.

It is expected that describing a resource will be done with many triples and indeed, might be a complex graph of relationships, including references to other resources - similar to how an HTML web page contains structure and links to other web pages, but the relationships in semantic web data have types. It is therefore necessary that triple stores can deal with large numbers of triples - in the millions.

Dynamism and Network Distribution - Using the network to fetch and store triples also impacts scalability, bringing in the issues of timeliness, caching and general problems from a distributed system that can fail or be delayed. The size of semantic web data transfered between network entities cannot be predicted and will typically vary a lot rather than be in fixed sized packets. The data may be generated dynamically from other forms and vary over time in different ways that, again, may not be predictable in advance before system design.

Semantic web data stores will also tend to be both web client and servers and thus need management of network resources in both fashions, to handle timeouts, network failure, bandwidth use and deal with denial-of-service.

Triple-stores that deal with remote resources or graphs need to address these problems in ways that handle network failure, bandwidth and system usage, management of network resources and dynamic data sizes in a graceful manner as the size of data and traffic grows.

Data Manipulation, Merging and Querying - The data in a triple-store needs to be manipulated - triples added to, modified and removed. This requires an API to the graph and support for identifying triples, querying the graph as well as administering the graph - creating and destroying it and operations for transferring data to and from the network.

The querying of the graph may require indexes to be created for efficient searching, or for providing specialised searches such as text-based ones, or ones based on properties, sub-properties, logical inference.

Data from multiple sources will be merged into single graphs of data, with the relationships between them connected up, to form the semantic web of data. This needs triple-store support in order to handle such merging when the graphs are large and to possibly separate them again later.

Such manipulations by applications using semantic web data will be occurring many times for even simple systems and thus these need to be lightweight, fast and easy to understand conceptually as well as easy to use via APIs from software or web methods.

Unpredictability - As already mentioned, the size of data, terms used in the data, resources being described, rate of change of data and effects of using the network combine to give a high degree of unknown factors for a triple-store. Only in particular, well-known and fixed-schema applications would it be possible to use a hand-coded storage format, such as those provided by specific RDBMS schemas, or using some specialised file format. Although for most semantic web applications, many of the core terms will be known, since that is how applications of the data connect together, there are usually application-specific differences that mean a specific schema cannot be applied more generally.

Semantic web data systems have to expect that unknown terms will be found, which may later on be related to known terms, and hence be applied by the applications.

Provenance - When triples are merged from multiple sources, it may be required to track their original location, if retrieved from the web. This information is part of what is generally called provenance and this may require tracking down to the level of a single triple, since it might also carry other information such as which entity said it, and the description of the individual may also need to be present in triples.

Standards - Wherever possible, each of the areas addressed by a triple-store should be done according to appropriate standards. This does not require that they be implemented in a particular way, but at the interfaces where such triple-stores related to other systems, that is ripe for use of a standard such as a transfer syntax, network interface, or standard language API. These may be existing formal standards from organisations, defacto standards from well used applications or standards developed by the community for particular purposes.

Our Approach - These influences tend to require that triple-stores are very flexible in their functionality, since they are dealing with a very general descriptive format. This tends to lead to a wide range of requirements such as quickly handling single triples as well as scaling to millions of triples. These can conflict and may not be available in a single triple-store system. These features may not always match the existing off-the-shelf tools available to-hand such as RDBMSs which are more static in their use.

Our approach is to survey the features, maturity levels, APIs and details of existing triple-stores, and update this over the lifetime of this project as the systems get updated. These results will be presented in terms of answers to typical situations that have been posed as questions by developers in public RDF fora and in personal communications, rather than based on the feature sets. Recommendations will be given on particular approaches and

tools for the use cases along, with some indication of their fitness for purpose, but not a detailed quantitative analysis of system performance which can be quite involved, and variable depending on the application.

In order to help developers of semantic web systems, some standard RDF datasets of typical semantic web data will be made available along with discussion of how the data can influence the triple-store requirements and performance, with hints on how to optimise.

2 Background

This section reviews existing work on storing triples, whether in triple-stores or other types of storage, looks at the type of semantic web data which may be stored in them where the data may include schema and ontological data such as RDF Schema☞[RDFSCHEMA] and the *DAML+OIL*☞[DAMLOIL] language from the *DAML*☞[DAML] project.

Data - There have been surveys made of RDF data, systems that support RDF and, the features that they provide, features that are needed and surveys of various methodologies and application techniques. This section gives an overview of the existing work in this area.

In ☞[RDFDATA]☞[RDFDATATR1] (and update 2002-08 ☞[RDFDATATR2]) a software robot called the *RDF Crawler*☞[RDFCRAWLER] (after ☞[RDFCRAWL]) is described that was used to survey RDF and DAML+OIL data that it could find on the web. This was done by starting from several RDF and DAML web sites and following the links, then looking up the schema namespace URIs declared in the documents, to find more information. It also supplemented the results by using a general search engine for finding `.rdf` files. The largest data sets found were the *Open Directory RDF content dump*☞[DMOZ] and the *Wordnet in RDF*☞[WORDNETSW] (not ☞[WORDNETWEB]) of which the former was not expected to point to other RDF data, so was not crawled. The resulting data was analysed for the use of URLs for resources and predicates, parsing errors, size of facts and types of literals. The authors found that although the search was not considered extensive, there was not a lot of RDF data on the public web and it was not yet very connected.

The DAML Crawler☞[DAMLCRAWLER] crawls the DAML statements from DAML files submitted to the web site by the project researchers and others. The crawler provides a summary of the data that was found in terms of simple counts of the *triples* and errors in parsing and retrieval but no other analysis at this time. It reports 3.5M triples in around 19,000 DAML documents, of which the majority, 2.5M were from 5 sites as shown in ☞Table 1:

Table 1:: DAML Crawler Top 5 Sites

Sites	Triples	Notes
http://www.daml.org/	1,024,976	DAML Site
http://orlando.drc.com/	685,766	Dynamics Research Corporation, Orlando(*)
http://www.semanticweb.org/	475,287	(probably WordNet, see above)
http://projects.teknowledge.com/	216,517	
http://orl01.drc.com/	213,032	Dynamics Research Corporation, Orlando(*)
Top 5 Total	**(74%) 2,615,578**	
Total	**(100%) 3,526,358**	

(*) Same host

The resulting triples are also made available by ☞Teknowledge as the ☞DAML Semantic Search Service using the *Agent Semantic Communication Service (ASCS)* ☞[TEKASCS]

The *DAML Data Sources*☞[DAMLDATA] site lists DAML instance data for a variety of applications including facts, schemas or DAML+OIL ontologies as well as mixtures of the above. Some of these are "scraped" from other web sites (statically, or on demand) and turned into semantic web data, others were authored as such (☞W3C Publications), or converted from other data formats (such as the ☞CIA World Fact Book in DAML. No analysis or summary of the collected and generated datasets has been made at this time.

In *Loading RDF into the Parka database*☞[PARKALOAD], Reck measured using the PARKA-DB☞[PARKADB] to store RDF data gathered from various ☞data sources. Some of the data had very large numbers of predicates counts that stressed the underlying store, others such as the Open Directory (DMOZ) data☞[DMOZ] had a more moderate number of predicates and more subject and object URIs.

Schemas - The ☞OntoWeb *D1.3 Survey on ontology tools* in ☞[ONTOSURVEYTR],☞[ONTOSURVEY] Section 6: *Ontology storage and querying*, pp74-93 covers substantially similar ground to this survey, although with emphasis on support for ontologies and querying, hence the title.

It describes a wide range of tools in terms of how they are appropriate for storing ontologies as well as (semantic web) data, accessing that data via APIs and especially via query interfaces. It looked at how the tools were are appropriate for ontological rather than for more simpler semantic web data applications. The tools and query languages were described across a variety of features, without giving independent performance figures, except for those provided by the tools.

> *RDF/S modeling primitives are substantially different from those defined in traditional database models such as object, relational or semi-structured, a fact that calls for different treatment ☞[ONTOSURVEY] 6.4 Comparison of Query Language and Storage Tools*

Which seems too strong a claim since object, relational and semi-structured storage systems are used to store semantic web data such as RDF.

Summary of some conclusions related to storing semantic web data from 6.4,6.5 ibid pp91-92:

- RDF/S support is more widespread than for DAML+OIL
- Most query languages are triple based
- Most tools use relational technology - (O)RDBMSs - to store data
- Most tools provided some inference report
- From an industrial perspective, the tools seem immature
- Exhaustive scalability and performance figures are not yet available

It then recommends that:

> *an extensive set of use cases from large-scale Semantic applications is required, as conducted by the W3C XML Query Group☞[XMLQUERYUSECASES]*

and also that a first effort in this area the work of Robie in *The Syntactic Web*☞[SYNTACTICWEB],☞[SYNTACTICWEB2]. The latter is about querying and as such, mostly out of scope for this survey, but transforms Topic Map and RDF data in XML into forms that more represent their data model, and then uses XQuery to operate on the resulting syntax as a query.

The ☞On-To-Knowledge (EU-IST-1999-10132) project *Sesame*☞[SESAMEPROJ] reports in *Sesame: A Generic Architecture for Storing and Querying RDF and RDF Schema*☞[SESAMETR]☞[SESAME] on the architecture of Sesame, a Java system explained more in ☞section 5. As part of the work, an analysis of the requirements for querying and storing RDF(S) data was performed, looking at existing solutions and finding them inappropriate, such as XML systems that handle trees but not graph structures or are clumsy in handling RDF queries.

The Sesame project described in ibid, 4.1 defined a *Storage And Inference Layer (SAIL)* to use over lower level storage layers such as RDBMSs so to abstract from individual storage system detail. The project recognised the following potential stores:

- DBMSs - any kind - RDBMS, ORDBMS etc
- Existing RDF stores such as rdfDB☞[RDFDB], RDFStore☞[RDFSTORE], Redland☞[Redland], ... etc.
- RDF files - flat files
- RDF network services

The Sesame development then concentrated on the first repository, a DBMS back-end based on PostgreSQL which is an object-relational DBMS, so supports relations between tables transitively and thus can be easily used for class and property subsumption in RDFS. Later on MySQL support was added via the JDBC-driver interface. See also ☞Section 5 for more on Sesame.

In *Benchmarking RDF Schemas for the Semantic Web*☞[BenchRDFS] the authors analyse the structure of RDF Schemas found in registries and elsewhere for diverse applications with their Validating Parser(VRP)[☞[VRP] which can analysis the RDF/XML syntax, check whether the schemas and instances satisfy the constraints of RDF Schema☞[RDFSCHEMA]

It divides RDF/S features into the following:

- *Core Classes* - Class, Property, Container
- *Abstraction mechanisms* - subClassOf, subPropertyOf
- *Restriction mechanisms* - domain, range
- *Documentation facilities* - label, comment, isDefinedBy, seeAlso
- *Reification mechanisms* - Statement, subject, predicate, object, type

and then analysed the schemas found against that, also classifying the schemas under their application domain. The main conclusions related to storage of schemes were as follows:

- Mostly only the core constructs are used - defining a few classes and properties
- The use of subPropertyOf is relatively small
- Multiple inheritance for classes is not widely used, but used more than that for properties

In *The RDFSuite: Managing Voluminous RDF Description Bases*☞[MANVOLUME] the authors describe using the *RSSDB - RDF Schema Specific DataBase (RSSDB)*☞[RSSDB] part of ICS-FORTH RDFSuite☞[RDFSUITE] and comparing using the same RDBMS with a *generic representation* using triples approach (with URI interning) versus a RDF schema *specific representation* structure (with indexes), where the tables are customised for the data. In all cases the schema-specific approach used less storage, was faster in loading and querying, and in some queries, very substantially quicker.

Native Triple Stores - There has been only one major survey on RDF Triple data stores (rather than RDBMS schemas for such, see next section) in *Survey of RDF/Triple Data Stores*☞[TRIPSURVEY] which gathered submitted descriptions of several triple stores as implemented in and summarised the results with no further analysis. Some performance figures were received and describe the size of triples data used, loading time and querying speed in triples/second.

Relational Stores - Storing RDF in relational databases has been surveyed in ☞[RDFRELATIONAL1] and records several submitted database schemas but not some already mentioned such as the RSSDB☞[RSSDB] or (below) the ☞[EDUTELLA] work. This will be investigated in more depth in this project by the ☞SWAD-Europe Deliverable 10.2 Mapping data from RDBMS.

Related work - This project has two other surveys that cover related areas to semantic web data storing. The first has already been mentioned ☞SWAD-Europe Deliverable 10.2 Mapping data from RDBMS which will be a more detailed look into the schemas used for RDBMS with recommendations. The second is the ☞SWAD-Europe WP 7 Deliverable 7.2 - Report comparing existing RDF query language functionality which also influences the requirements on underlying stores.

There are three surveys of querying RDF/semantic web data *RDF Query and Rules Status*☞[RDFQUERYEP] which rewrites the same query of RDF data in different query languages, *RDF Query and Rule languages Use Cases and Examples survey*☞[RDFQUERYAR], an ongoing survey and collection of RDF queries use cases with query language examples and the *EDUTELLA project*☞[EDUTELLA] which defined five RDF query exchange language levels and uses them to map to existing RDF query languages.

Other related work on storing semi-structured data (any kind of data that can be represented as a graph)☞[QuerySemi] includes the literature on conceptual graphs, Extensible Markup Language (XML) and XML databases. *The Semantic Web - on the respective Roles of XML and RDF* ☞[SEMWEBXMLRDF] discusses the differences and trade offs between using XML and RDF for semantic web data.

3 Application Requirements

An application looking for a semantic web storage system will have several requirements on the tool that need to be satisfied ranging from absolute requirements to items that would be useful or enhance the work. This section outlines the key requirements that have been found after experience in maintaining the ☞RDF Resource Guide☞[RDFGUIDE], answering questions on RDF mailing lists and IRC.

Implementation Language - The critical question that needs to be considered first is what implementation language or languages are being used. This approach was used when writing the ☞Developer Resources section of the ☞RDF Home Page☞[RDFHOME] to give pointers to developers who frequently ask about which tools to use in general.

This question generally divides the appropriate tools into suitable candidates more quickly than other considerations. However, if the application is more flexible, and the implementation language isn't critical, then the other features can direct more appropriate selections, since at present, not every implementation language has the same level of tool building activity and feature set.

Standard RDF APIs? - A second question that is often asked is whether there is a standard RDF API - like the DOM, SAX, Infoset for XML. There are no formal standardised APIs for RDF at this time, which means selecting one tool over another will tend to mean that it isn't so easy to change later. There are, however, tool APIs that have been deployed and stable for several years with widespread use, have a broad range of functionality and support. An alternative to an API is to use a query language which can be easier to change than numerous API calls.

Working with Existing Systems - In addition to implementation language, it may be that other tools need to work with the storage system such as particular servers or databases. These may only be available to some tools via the standard interfaces (ODBC, Java JDBC, Perl DBI etc.) or particular stores might already require a different database or server.

Licensing - This report describes Free Software☞[FREESOFTWARE] or Open Source☞[OPENSOURCE] systems, which are licensed in different ways. This may not always be compatible with the license of the application, or other subsystems that might be used. For example, one tool license might require publishing of some application source code whenever it is used with the tool and distributed.

Other features - The detailed application choices are more related to the details of the storage system and what features it has which are described in the ☞section 4.

4 Store Features

The *Ontology storage and querying*, pp74-93 Section 6 of ☞[ONTOSURVEY] looks at the features of several storage tools, summarised in Table 6.4. The features cover some of the storage aspects of the tools from the point of view of supporting complex ontological requirements, which are not necessarily needed for all semantic web applications, in the same fashion how XML and HTML do not require validation to be useful.

This section outlines the key features of semantic web stores that apply to choosing an appropriate one for an application, or answering FAQs as described in later sections. The features below are applied against the available free software/open source tools in the next section.

Programming Language and System
: What is it implemented in, what does it run on and support.

APIs
: Triples based API and other interfaces.

Simplicity
: How easy to configure, build and use.

Capacity
: Number of triples, resources and properties supported.

Performance

Speed of storing, retrieval returned by a API operation.

Query Languages
: Query language(s) in the store including updates.

Subsumption
: Support for subClassOf and subPropertyof.

Inferencing
: RDF Model Theory, more advanced (DAML+OIL) logics.

Resource Use
: System resource use such as memory, disk and CPU.

(O)RDBMSs Used
: Database APIs such as ODBC (C), JDBC (Java) and DBI (Perl).

Text Indexing
: Handling literal content and free-text searching of it.

Network/Web
: Retrieval and storing information in the web using network APIs.

Dynamic Schemas
: Support for unknown properties, classes at run-time.

5 Current Systems

Introduction - There are existing storage systems available for semi-structured data in general. These include systems that store XML. However the XML data model is a tree-like structure with elements and attributes in different facets which is rather different from the triple model of semantic web data representing a graph or web of data, where there is no hierarchy. XML has been used to store triple-like data by rewriting the triples into simple 3-part XML element structures and then using existing XML query systems as described in ☞[SYNTACTICWEB],☞[SYNTACTICWEB2]. This works for simple manipulations but cannot handle RDF Schema processing on predicates such as subPropertyOf and is clumsy when used with queries and inference. XML databases with XML Query support, do provide other useful functionality such as support for querying by paths of elements and dealing with literal content that could be exploited for RDF data.

Semantic web data is based on using URIs for identifying all the terms, so any existing system being used for storing such data has to be able to support those rather some more restrictive identifier form. This can be done by adding an extra mapping layer between the other identifier form and URIs.

The data is in the form of triples representing the RDF graph, although the graph itself could be directly represented, the RDF graph does not allow individual nodes or arcs to be used without connecting into complete triples, for example a node must have at least one arc connected to it. An graph-based store could be used as long as the graph could only create RDF graphs in this form.

Some other systems are not being covered in this section which provide importing and exporting of RDF as triples but are not based on the triple model internally. These generally cannot handle storing arbitrary RDF data but can handle application-specific sets of triples and can look like triple stores when used that way. These could be mappings from an existing database, with the restrictions imposed by the database schema or from another model that cannot easily be mapped to a general triple store.

The use of RDBMSs for semantic web storage, the issues and the schemas used is covered in the ☞SWAD-Europe Deliverable 10.2 RDBMSs and Semantic Web Data which is in progress.

4Suite - *4Suite*☞[FOURSUITE] is a Python system for XML and RDF processing. It has substantial RDF support and for that has a triple store interface to a simple dump format for in memory stores, and to RDFBMSes via ODBC or directly with Oracle and PostgreSQL and a path-style query language.

License: ☞4Suite License - Apache/BSD-like with advertising required.

DAML DB - *DAML DB*☞[DAMLDB] is a small C++ library that provides a store for DAML (or RDF) content using BDB using fixed-sized fields (integers) for efficient indexing. It also provides a Java interface using *Jena*☞[JENA] and the Model interface. Has been used with the *The DAML Crawler*☞[DAMLCRAWLER] data which runs into several million triples.

License: none found (public domain?)

EOR - The *EOR Toolkit*☞[EOR] is a Java toolkit that provides basic RDF database functionality and generic support for creating web interfaces to them. It uses MySQL or PostgreSQL to provide the triple store and additional Java components (Java Servlets, Xalan). EOR has no query language, schema or inferencing support but Java provides network access and portability.

License: ☞Dublin Core Open Source Software License

Haystack - *Haystack*☞[HAYSTACKPROJ] is a Java application as described in *Haystack: A Platform for Creating, Organizing and Visualizing Information Using RDF*☞[HAYSTACK] using JDBC, a relational database and a SQL-like language HSQL to talk to it. However that was found to be too slow so the project has developed an in-process RDF database in C++ using JNI to connect to Java. This was optimised for RDF but is seen as a temporary solution until more support is available from commercial databases. A new language Adeline was created to support the Haystack data model along with RDF.

License: it isn't clear if this research project will release any code under a free software or open source license, although it depends on several open source tools for functionality.

Inkling - Inkling☞[INKLING] is a Java implementation of the SquishQL RDF query language. It has an in

memory representation of graps and uses PostgreSQL to store the triples persistently, accessing it via JDBC and maps the query language into Postgres SQL. Inkling via SquishQL does not provide special schema support but can handle RDF graph queries. Being written in Java, it is relatively portability but depends on PostgreSQL.

License: GPL

Jena - Jena☞[JENA] is a Java semantic web toolkit that provides a rich API for manipulating RDF and persistent storage of triples using either Sleepycat / Berkeley DB (henceforth BDB) or via JDBC to talk to a variety of RDBMSs including MySQL, PostgreSQL, Oracle, Interbase and others. It provides triple based and resource or frame-centric APIs, DAML+OIL basic support, RDFQL query language and has a related separate *Joseki* network API as discussed in ☞section 6. Jena has been used with millions of triples in memory (if rather slow) and is limited mostly only by system resources. Jena depends on a selection of standard Java APIs which are widely portable and the large range of storage options allow it to be deployed on many systems.

License: Apache/BSD-style license without advertising.

KAON - The ☞KAON project's ☞KAON server uses Enterprise Java Beans (EJB) and a relational database via JDBC to provide a triple store or RQL-based repository.

The *Karlsruhe Ontology and Semantic Web Infrastructure*☞[KAONWP] describes in section 4.2.1 the *KAON-SERVER* which is the ontology repository for the KAON project. It was created with the requirements of Persistence, Update semantics, Concurrency, Security that were not seen as available in other RDF repositories at the time of design.

The project also recognised that inferencing and querying would be common tasks for RDF data and found it interesting that these were not currently available in other applications. Scalability and performance were also required and thus the need for highly optimized structures for inference engines. It recognised that RDF data may change the schema which can clash with optimization techniques and proposed to build an infrastructure around a basic repository. KAON also provides a separate interface to external inference engines that can register themselves dynamically. It provides service connectors (or Net APIs), presently Java RMI, IIOP and SOAP.

License: Apache/BSD-style with advertising

Parka Database - *The Parka Database*☞[PARKASW],☞[PARKADB],☞[PARKADBIK] - part of the Parka-KB is a knowledge representation system based on semantic network, layered on frames using a relational database beneath (presently an internal one, for speed). It uses a fixed number of predicates to optimise the speed of operation and has optimised structures for handling inheritance of classes and properties. The indexing of properties is moved between disk and memory on demand. The system has been used with over 2M assertions and although applied to the KR world, is now being updated for semantic web data (with URIs) and the RDF-based models.

License: MIT License with advertising not required.

rdfDB - *rdfDB*☞[RDFDB] by R.V. Guha is an RDF system written in C that provides an RDF server, query language and BDB for persistent triple store. It has a simple model and gives high performance since it is all in one application.

License: Mozilla Public License (MPL).

Python RDFLib - *RDFLib*☞[PYRDFLIB] is a pure Python RDF library that provides RDF/XML parsing and triple storage. It comes with an in memory triple store implementation that uses a new feature in Python 2.2 - generators - to give a high level, layered and flexible triple store interface☞[RDFLIBTS]. No persistent triple store is provided at present.

License: Apache/BSD-like license with advertising not required.

RDFStore - *RDFStore*☞[RDFSTORE] is a pure-Perl RDF API and provides persistent store using the standard perl DB_File mechanism, a separate BDB module and using the DBD drivers and DBMSes. The objects are stored via Perl object serialisation into the stores. Supports statements grouped into contexts and uses a Squish-style query language.

License: Apache/BSD-like license with advertising required.

RDFSuite - ICS-FORTH RDFSuite☞[RDFSUITE] part of the IST-1999-13479 ☞C-Web (Community Web) project defined a *RSSDB - RDF Schema Specific DataBase (RSSDB)*☞[RSSDB] in *The RDFSuite: Managing Voluminous RDF Description Bases*☞[MANVOLUME] for RDFSuite. This is an RDF store that uses schema knowledge to automatically generate an Object-Relational (SQL3) representation of RDF metadata. Internally it has tables for Class, Property, SubClass, SubProperty and the particular classes (instances) and properties (source, target) in the RDF schema being used and hadnles XML Schema data types for literal values, grouping and filtering primitives and sorting.

License: Under the ☞RDFSuite License (C-Web license on the web page) which allows free (price) use of the software as long as credit is kept.

Redfoot - *Redfoot*☞[REDFOOT] is a pure Python RDF server, using the *RDFLib*☞[PYRDFLIB] library described above and provides no triple store of its own.

License: Apache/BSD-like license with advertising not required.

Redland - Redland☞[Redland] is a C RDF system detailed in ☞[REDLANDDES] which describes how it uses BDB to store triples in a persistent fashion, after earlier work using rdfDB. It has language interfaces to Perl, Python, Java, Tcl and Ruby in addition to C.

License: GPL / LPGL / Mozilla Public License (MPL)

Sesame - Sesame☞[SESAMEPROJ]☞[SESAMETR]☞[SESAME] is a Java system which defines a *Storage And Inference Layer (SAIL)* to use over lower level storage layers such as RDBMSs so to abstract from individual storage

system detail The DBMS backing store supports RDFS class and property subsumption over PostgreSQL, MySQL and Oracle9i databases, with different functionality depending on deatbase features. The DBMS backend also supports and optimises RDF schemas, has RDF Semantics inferencing and two RDF query languages - RQL and RDQL. A high-performance in-memory SAIL with persistent store is being developed, currently without inferencing capabilities. See also the discussion of Sesame's SAIL repository API in ☞section 6. Sesame has an active development funded by the ☞NLNet Foundation with regular recent releases in 2002 and 2003.

License: GPL/LGPL

Edutella - EDUTELLA☞[EDUTELLA2] is a P2P networking infrastructure based on RDF (ibid) based on the Java JXTA and in *Towards a Modification Exchange Language for Distributed RDF Repositories*☞[MODEXDIST] describes how they distributed RDF repositories via the KAON framework, described above. It is thus a slightly different storage system being distributed and addressing those issues, but based on another storage. It is also written in Java and being is deployed for distributed learning repositories. This is relatively recent work but based on earlier developments in the same area and has substantial thought in the design for query language support, distribution, replication and annotation.

License: ☞Sun Project JXTA Software License - Apache-style license with advertising.

6 Storage and Network Storage APIs

There are no standard storage APIs yet defined for semantic web stores, in the formal sense of being submitted to standard organisations. There are some proposals and well-used APIs that can be considered defacto standard by their widespread adoption. In particular, many RDF APIs have been inspired from the *Mozilla RDF API*☞[MOZILLARDF] which defines several classes that have been picked up by related APIs such as the *Stanford RDF API*☞[STANFORDRDFAPI] *Jena*☞[JENA] *Redland*☞[Redland] and their derivations. RDFStore☞[RDFSTORE] (described above) is explicitly written as a Perl translation of the Java Stanford RDF API. Another early and influential proposal was RADIX☞[RADIX] by Ron Daniel Jr.

This project will be investigating RDF APIs in more detail in ☞SWAD-Europe WP 7 Deliverable 7.1 - Report comparing existing RDF API functionality which will address triple APIs as part of that. There have been a few existing surveys of RDF APIs such as ☞[RDFAPIS] which was done in 2000 and doesn't cover many more recent major tools such as Jena.

Joseki☞[JOSEKIPROJECT] and its RDF NetAPI☞[RDFNETAPI] defines a Java interface for accessing and updating remote repositories, for use with Jena. This is still under development.

Sesame☞[SESAMEPROJ] defines a generic API for RDF (Schema) repositories called *RDF SAIL (Storage And Inference Layer)*☞[SESAMESAIL] used to abstract from the actual storage device. This API has methods for storing, removing and querying RDF in/from a repository. Implementations of this API are used by Sesame's functional modules, but can also be used by stand-alone tools that need some kind of repository for storing their RDF. It is claimed that SAIL implementations can be built on top of any kind of storage, a database, file, or a peer-to-peer network. The current implementations are based on relational databases and in-memory with persistent store.

Sesame also defines a network API for addressing the repository and administering it as described in *Communicating with Sesame*☞[SESAMECOMM] which describes how it uses a simple XML format communicated over HTTP, RQL sent as RDF/XML or HTTP query using RDF/XML as results. Further work has been done by *OntoText*☞[ONTOTEXT] in the *Ontology Middleware Module* (OMM)☞[OMM] to provide SOAP and Java RMI network interfaces for Sesame.

Towards a Modification Exchange Language for Distributed RDF Repositories☞[MODEXDIST] describes EDUTELLA's☞[EDUTELLAPROJ] storage API which consists of the MEL language that transfers sub-graphs around rather than triples and includes support for replication.

The TAP Project☞[TAPPROJ] has recently defined an *GetData Interface*☞[GETDATA] for machine access to querying the TAP Knowledge Base. This interface is still being deployed and not yet entirely public at this date (July 2002).

7 Data Sets

There are several large semantic web data sets that can be used with stores for testing their scalability and performance. There are also other datasets which address other typical features of RDF data such as use of common properties, schema information and large and complex ontologies. This section contains pointers to some of these data sets and will be updated as they are found and/or collected.

- Open Directory RDF Dump (DMoz) / chef DMoz - a very large dataset that tests the scalability and performance of stores and APIs.
- Adobe's XMP tool can extract from recent PDFs (of which there are probably millions containing RDF) - tests with real RDF data that is being generated by end users.
- Dublin Core Metadata Element Set as used in Internet catalogues such as RDN (in the UK) - more real data in a usually quite simple flat metadata format, of the scale of 10,000s of records with small numbers of predicates with requirements for good text indexing, searching and result ranking.
- DAML Data as described in ☞DAML Data Sources and ☞Possible DAML Use Cases which are a variety of richer data with more schema and other restrictions that can exercise the schema processing parts of tools.
- Wordnet - several RDFized versions exist such as ☞[WORDNETSW] and ☞[WORDNETWEB] - these can

test large, deep and very interconnected ontologies.
- Thesauri in RDF - rich structures, sometimes mapped from other thesauri formats in the digital library world.
- RSS 1.0☞[RSS10] or converted from the various 0.9x formats as gathered from aggregators such as Syndic8 or Meerkat - testing more typical data from users, with a possibility of a lot of variety if using RSS 1.0 modules.
- W3C Tech Reports in RDF page - a representation pointing to documents, people and organisational parts of the W3C.
- ☞"Friend of a Friend" (FOAF) data which includes image / co-depiction data and can test distributed data handling, retrieval abilities stores.
- ☞FOAFCorp (data from theyrule.net) - describing relationships between people, companies.
- Automatically generated random RDF graphs - this has been used in the past to generate a multiple legal RDF/XML forms from walking the grammar and can generate unusual graphs that can be used to exercise all parts of a store.
- The TAP☞[TAPPROJ] Knowledge Base, a "shallow but broad knowledge base containing basic lexical and taxonomic information about a wide range of popular objects" - a large dataset for testing
- RDFIG chat logs - which have a large number of predicates as well as millions of triples
- RDF Core Working Group test cases data - covering and testing various aspects of the RDF model and syntax
- buggy / problematic / corner case data - to be developed
- PARKA-KB☞[PARKADB] data - this has been tested with some of the other data sets mentioned here plus some others that might usefully be added to the general tests.
- Universal Product Codes (UPC) such as seen in bar codes on many products - some of it is seen at ☞UPC Database but unofficial.
- Mail archives and threads mapped into RDF like has been done in ☞maillog2rdf by Dan Connolly, W3C - can give many large datasets with interesting and useful relationships from threading, message IDs.
- Musicbrainz metadata for music may be expressed in DC and RDF - a potential large dataset of useful information.
- ☞RPMfind which uses RDF/XML to describe RedHat packages (RPMs) - would benefit from dynamic updating of the large data set as well as good text searching and organising of search results.
- Ontologies as data such as those in the ☞DAML ontologies library
- ☞[BenchRDFS] found large schemas available for "Real Estate Data Consortium, Basic Semantic registry, UNSPC, Gene Ontology" that could be used.

Future work in this area could be to collect examples of these centrally for easy use by application writers, and develop test data that aren't covered by the above. In particular it has been found that the support for literal data varies a lot, some stores may break with lots or large literals or handle them badly and data with large numbers of predicates (10000+) isn't commonly supported - typically, there are low numbers of predicate URIs compared to resource URIs

8 Frequently Asked Questions (FAQs)

This section is intended to answer Frequently Asked Questions (FAQs) from users and developers about storing triples based on experience in maintaining the ☞RDF Resource Guide☞[RDFGUIDE], answering questions on RDF mailing lists and from IRC.

I have an RDBMS *name*, how do I implement an RDF store on top? - Most of the main RDF tools provide some way to map an existing RDBMS as a low-level store. The particulare DBMS available may not, however, be supported by all systems. The most common are the freely available PostgreSQL and MySQL which are used by several tools but some such as Inkling, 4suite and Jena provide access via JDBC to a wide range of SQL backends and have been tested on them. The ☞SWAD-Europe Deliverable 10.2 RDBMSs and Semantic Web Data will be investigating further the ways that this can be done and evaluating existing schemas and applications of this. *Storing RDF in a relational database*☞[RDFRELATIONAL1] by Melnik points at some schemas that have been used in the past, although it isn't very up-to-date. The Jena☞[JENA], RSSDB☞[RSSDB] and other tool documentation list the schemas used there and provide a more up-to-date guide to database schema design for triplestores.

I am using *language*, how can I use an RDF store in it? - If you want an interface in a particular language there are tools that provide it either by being purely written in the language without require no extra configuring and others that provide *access* to the tool which is written in another language.

C
: Redland, rdfDB

C++
: DAML DB

Java
: Edutella, EOR, Haystack, Inkling, Jena, KAON, RDFSuite, Sesame, DAML DB (access), Redland (access)

Perl
: RDFStore, rdfDB (access), Redland (access)

Python
: 4Suite, RDFLib / Redfoot, Redland (access)

Tcl

Redland (access)

Ruby

Redland (access)

Which shows how much Java is prefered over other languages, if you are flexible. There is also a lot of Python development on smaller scale, not shown here.

Note: This list is not complete for general RDF tools and there are new tools emerging regularly. See ☞RDF Resource Guide☞[RDFGUIDE] for full details.

I want to build an RDF application for *this application*. What features should I look for in an RDF store? - This is an open ended question that depends greatly on the subject of the application. Some things are more like data aggregation and need help with contexts, which can be a key feature; others are heading towards needing advanced query systems and manipulation of complex ontologies, inferencing and support for higher level logics.

Some examples:

- Annotation, Personal Information - provenance is crucial, support for distribution and ability to gather information with network support.
- Thesauri, Ontologies - schema support, schema validation, querying and inferencing
- Data Aggregation, Syndication - provenance, large data support, network access
- Digital Libraries - text indexing and searching, querying, result ranking
- Digital Media Description - provenance, large data support, dynamic networking
- Web Knowledge Bases - large data support, querying, network APIs

Which store features are widely implemented? - In particular the most implemented are:

- A general triple store
- RDBMS-support
- General RDF model access
- Query language support in the store such as RQL, RDQL

Some stores provide the following:

- Provenance - tracking of who-said-whaat
- Schema support - RDF schema, DAML+OIL or other
- Network retrieval and APIs

Very few stores provide:

- Full text search
- Inference and rule languages

What free RDF storage tools are there that I can use or adapt? - These are the main subject of this report and are outlined in ☞section 5 - Current Systems.

I need to store *this many* statements, what is best for that? - Some of this depends on the scale of the underlying store, since many tools use RDBMSs to handle the raw data management. BDB based systems such as rdfDB have handled 20M+ triples and scale relatively well due to the code being lightweight. If access is required via a particular language, that restricts the options and the potential for scalibility.

How well does this scale? Cannot be answered too easily at present without figures. *The RDFSuite: Managing Voluminous RDF Description Bases*☞[MANVOLUME] described in the presentation results for scaling with their particular approach and found that as the triples increased, the custom database schema for the particular application scaled better than a generic store.

How many triples can these systems store? - *(Is this even a sensible question to ask?)* Yes, although the stores may not necessarily express the data in triples. There have been reports in *Survey of RDF/Triple Data Stores*☞[TRIPSURVEY] of several million 1.5M (RDFStore), 6M (RDF Suite), 20M (rdfDB), 1.5M (Redland), 300K+ (Sesame), 800K+ (Jena).

The sesame project reports in personal communication that the in-memory store has been tested with 500K triples and the RDBMS storage with 5M triples on standard desktop hardware. The respnse time for simple queries to the in-memory store was sub-50ms.

How do I transfer RDF to/from stores on the web via SOAP / XML-RPC / HTTP? - This isn't well supported at present although some work is being done in evaluation SOAP for transfering RDF in this project in ☞SWAD-Europe WP 5: Integration with XML technology. *Joseki*☞[JOSEKIPROJECT] and its *RDF NetAPI*☞[RDFNETAPI] also have been addressing this problem but no standard has yet been accepted. The best method to use so far is to use HTTP GET and PUT(POST) of RDF/XML content representing the graph that is transfered. Other methods using SOAP or XML/RPC will be at the application level until more standard approaches evolve.

The TAP Project☞[TAPPROJ]'s *GetData Interface*☞[GETDATA] looks plausible as some standard interface for the future but is still early in its public visibility.

Is there an RDF storage object model such as DOM, CORBA, DCOM? - Not at present but the models generally look very similar - a set of triples and most of them follow similar APIs as discussed in ☞section 6.

This project will be investigating RDF APIs in more detail in ☞SWAD-Europe WP 7 Deliverable 7.1 - Report

comparing existing RDF API functionality

Are there common implementation-independent interfaces to RDF stores? - Standard interface languages like IDL (in general), CORBA IDL, UML, .Net are not mature yet for RDF apart from the Mozilla IDL☞[MOZILLARDF] which has been deployed in the browser since 1999.

Are there any test suites or datasets for use by RDF store implementors? - See the list of possible data sets in ☞section 7.

How can I export (or expose) my RDBMS data as RDF? - There has been some research work on automatically mapping RDFBMS data into RDF but not very widely considered. The next ☞SWAD-Europe Deliverable 10.2 RDBMSs and Semantic Web Data will be addressing this in terms of what tools help with this and what problems there are such as scalability.

What tips are there on writing an RDF store? - The first answer is to not reinvent something but investigate using one of the existing stores in ☞section 5 - Current Systems.

If you want to write one from scratch then that requires a detailed analysis of the problem such as was done for Haystack in *Haystack: A Platform for Creating, Organizing and Visualizing Information Using RDF*☞[HAYSTACK] where they found their requirements exceeded the stores that were available. This cannot be resolved in a quick FAQ answer.

An intermediate step is to use BDB to provide the low level disk access, storage and indexing for the triple store and then write above that. This has been used by several systems with success and can give a good cost/benefit ratio compared to writing an entirely new system. The BDB indexing are mostly only described in the code and haven't at present been investigated deeply. See ☞[REDLANDDES] for one method.

What tips are there on writing a RDF store over a RDBMS? - The first answer is to not reinvent something but investigate using one of the existing stores, most of which have RDBMS backends.

If this isn't possible, then there are several design choices that can be made:. Interning the URIs into local IDS, whether to track provenance on each statement, optimising particular predicates for the application, specially handling literal content if it needs certain searches. It may also be that the data naturally is modelled in a different way in which case a mapping to/from the RDF data would be the best approach with the application ensuring that the model is correct at all times by validating the set of triples that are added. There is lots more flexibility in designing your own schema but optimisation strategies for schema processing, handling unknown properties etc. may require careful optimisation.

See also the next ☞SWAD-Europe Deliverable 10.2 RDBMSs and Semantic Web Data which will address this further.

A References - Publications

[RDFMS]
☞*Resource Description Framework (RDF) Model and Syntax Specification*, O. Lassies and R. Swick, Editors. World Wide Web Consortium. 22 February 1999. This version is http://www.w3.org/TR/1999/REC-rdf-syntax-19990222. The ☞latest version of RDF M&S is available at http://www.w3.org/TR/REC-rdf-syntax.

[RDFSCHEMA]
☞*RDF Vocabulary Description Language 1.0: RDF Schema*, D. Brickley, E.V. Guha, Editors, World Wide Web Consortium W3C Working Draft, work in progress, 19 March 2002. This version of the RDF Primer is http://www.w3.org/TR/2002/WD-rdf-schema-20020430/. The ☞latest version of the RDF Primer is at http://www.w3.org/TR/rdf-schema/.

[RDFRELATIONAL1]
☞*Storing RDF in a relational database*, Sergey Melnik, Stanford University, 2000-2001

[ONTOSURVEYTR]
☞*Ontology Storage and Querying* - Survey on Query Languages/Tools for RDF/S, DAML+OIL, TopicMap, Techical Report 308, ICS-FORTH, April 2002

[TRIPSURVEY]
☞*Survey of RDF/Triple Data Stores*, Art Barstow, W3C, April 2001

[RDFDATATR1]
☞*Survey of RDF data on the Web*, A. Eberhart, Technical Report, International University in Germany, 20 December 2001. ☞details

[RDFDATATR2]
Survey of RDF data on the Web - ☞PDF, A. Eberhart, Technical Report, International University in Germany, 15 August 2002. ☞details

[RDFDATA]
Survey of RDF data on the Web, A. Eberhart, Proceedings of 6th World Multiconference on Systemics, Cybernetics and Informatics (SCI 2002), Orlando, USA

[RDFQUERYEP]
☞*RDF Query and Rules Status*, Eric Prud'hommeaux, W3C, November 2001.

[RDFQUERYAR]
☞*RDF Query and Rule languages Use Cases and Examples survey*, Alberto Reggiori, Joint Research Centre (JRC), Italy, 2002

[SEMWEBXMLRDF]

☞*The Semantic Web - on the respective Roles of XML and RDF*, Decker at al, IEEE Internet Computing, vol 4, number 5, pp 63-74, 2000.

[MANVOLUME]
The RDFSuite: Managing Voluminous RDF Description Bases (☞HTML ☞PDF), S. Alexaki and V. C hristophides and G. Karvounarakis and D. Plexousakis and K. Tolle, Technical report, ICS-FORTH, Heraklion, Greece, 2000.

[QuerySemi]
Querying Semistructured (Meta)Data and Schemas on the Web: The case of RDF RDFS, Greg Karvounarakis, Vassilis, ICS-FORTH, Heraklion, Greece, 2000

[BenchRDFS]
Benchmarking RDF Schemas for the Semantic Web (☞PDF), A. Maganaraki, S. Alexaki, V. Christophides, and Dimitris Plexousakis, ICS-Forth, Heraklion, Greece in proceedi ngs of ☞First International Semantic Web Conference (ISWC'02), Sardinia, Italy, June 9-12, 2002.

[SESAMETR]
☞*Sesame: A Generic Architecture for Storing and Querying RDF and RDF Schema*, OTK-del-10, ☞On-To-Knowledge deliverable 10.

[SESAME]
Sesame: A Generic Architecture for Storing and Querying RDF and RDF Schema (☞Springer subscriber-only link) , Jeen Broekstra, Arjohn Kampman, and Frank van Harmelen in proceedings of ☞First International Semantic Web Conference (ISWC'02), Sardinia, Italy, June 9-12, 2002.

[ONTOSURVEY]
☞*Ontoweb Deliverable 1.3: A survey on ontology tools*, ☞OntoWeb (IST-2001-29243), 31 May 2002

[PARKADBIK]
☞*Parka-DB: Integrating knowledge- and data- based technologies*, Kilian Stoffel, Merwyn Taylor, James Hendler, University of Maryland at College Park, USA.

[XMLQUERYUSECASES]
☞*XML Query Use Cases* , Don Chamberlin, Peter Fankhauser, Daniela Florescu, Massimo Marchiori, Jonathan Robie, W3C Working Draft, work in progress, 22 August 2003. This version of the XML Query Use Cases is http://www.w3.org/TR/2003/WD-xquery-use-cases-20030822/. The ☞latest version of the XML Query Uses Cases is at http://www.w3.org/TR/xquery-use-cases/.

[SYNTACTICWEB]
☞*The Syntactic Web* - Syntax and Semantics on the Web, Jonathan Robie, Software AG, USA in proceedings XML Conference, December 9-14 2001, Orlando Florida , USA.

[MODEXDIST]
Towards a Modification Exchange Language for Distributed RDF Repositories (☞Springer subscriber-only link), Wolfgang Nejdl, Wolf Siberski, Bernd Si mon, Julien Tane in proceedings of ☞First International Semantic Web Conference (ISWC'02), Sardinia, Italy, June 9-12, 2002.

[EDUTELLA]
☞*EDUTELLA: A P2P Networking Infrastructure Based on RDF*, Wolfgang Nejdl et al. in proceedings of ☞WWW2002, May 7-11, 2002, Honolulu, Hawaii, USA.

[KAONWP]
☞*The **KA**rlsruhe **ON**tology and Semantic Web Infrastructure*, Siegfried Handshuh, Alexander Maedche, Ljiljana Stojanovi c, Raphael Volz, ☞KAON project.

[PARKALOAD]
☞*Loading RDF into the Parka database*, Ronald P. Reck, ☞MINDSWAP, University of Maryland at College Park, USA

[HAYSTACK]
☞*Haystack: A Platform for Creating, Organizing and Visualizing Information Using RDF*, David Huynh, David Karger, Dennis Quan in Proceedings of ☞International Workshop on the Semantic Web, ☞WWW2002, Hawaii, 7 May 2002.

[RDFNETAPI]
Accessing RDF remotely - An RDF NetAPI using HTTP, Andy Seaborne, HP Labs in proceedings of ☞First International Semantic Web Conference (ISWC'02), Sardinia, Italy, June 9-12, 2002. (Also HPL-2002-109 technical report from HP Labs).

[EDUTELLA2]
☞*EDUTELLA: Searching and Annotating Resources within an RDF-based P2P Network*, Wolfgang Nejdl, Boris Wolf, Steffen Staab, Julien Tane in proceedi ngs of ☞International Workshop on the Semantic Web, ☞WWW2002, Hawaii, 7 May 2002.

[FREESOFTWARE]
☞*The Free Software Definition*, GNU Project, 1996-2001

[OPENSOURCE]
☞*The Open Source Definition*, Open Source Initiative, 2002

[RDFGUIDE]
☞*Resource Description Framework (RDF) Resource Guide*, Dave Beckett, ILRT, University of Bristol, 2002

[SYNTACTICWEB2]
☞*The syntactic Web: Syntax and semantics on the Web*, Jonathan Robie, Software AG in proceedings of ☞X ML 2001, Montreal, Canada, August 2001

[JENATR]
☞*Jena: Implementing the RDF Model and Syntax Specification*, Brian McBride, HP Labs in proceedings of the ☞S econd International Workshop on the Semantic Web, ☞WWW10, Hong Kong, 1 May 2001.

[REDLANDDES]

☞*The Design and Implementation of the Redland RDF Application Framework*, Dave Beckett, ILRT, University of Bristol in proceedings of ☞WWW10, Hong Kong, May 2001.

[RDFAPIS]
☞*Summary of Recent Discussions about an Application Programming Interface for RDF*, Peter Hannappel, University of Essen, Germany, March 2000.

[STANFORDRDFAPI]
☞*(Stanford Java) RDF API Draft*, Sergey Melnik, Stanford University, 1999-2001.

[RADIX]
☞*RADIX - A proposal for an RDF API*, Ron Daniel Jr., June 1999

[SESAMECOMM]
☞*Communication protocols* in ☞User Guide for Sesame, Aidministrator Nederland b.v. and Sirma A I Ltd., 2002

[GETDATA]
☞*GetData Data Query Interface*, TAP Project, Stanford University, 2002

[RSS10]
☞*RDF Site Summary (RSS) 1.0*, RSS-DEV Working Group, 2001-05-30

B References - Tools and Projects

[RDFCRAWLER]
☞*RDF Crawler*, an extension of ☞RDFCRAWL

[RDFCRAWL]
☞*RDF Crawl*

[DAMLDATA]
☞*DAML Data Sources*, Mike Dean, ☞BBN / Verizon

[DAMLCRAWLER]
☞*DAML Crawler*, Mike Dean and Kelly Barber, ☞BBN / Verizon

[PARKADB]
☞*PARKA-DB: A Scalable Knowledge Representation System>*

[PARKASW]
☞*ParkaSW - Parka inferencing database* (open source version), ☞MINDswap group, University of Maryland, College Park, Maryland, USA.

[ONTOMERGE]
☞*OntoMerge*, tool for performing ontology translation by merging ontologies.

[RSSDB]
☞*RSSDB - RDF Schema Specific DataBase (RSSDB)*, ICS-Forth, 2002

[RDFDB]
☞*rdfDB RDF Database*, R.V. Guha, 2000.

[RDFSTORE]
☞*RDFStore Perl API fgor RDF Storage*, Alberto Reggiori, 2002

[LIBGNURDF]
☞*libgnurdf - GNUpdate RDF library*

[TEKASCS]
☞*Agent Semantic Communication Service (ASCS)*, ☞Teknowledge. GPLed DAML search software.

[DAML]
☞*DAML: The DARPA Agent Markup Language*, program of US ☞DARPA.

[DAMLOIL]
☞*DAML+OIL Language*, ☞DAML project.

[DMOZ]
☞*Open Directory Project RDF content*

[WORDNETSW]
☞*WordNet in RDF Schema* by Sergey Melnik, Stefan Decker, Stanford

[WORDNETWEB]
☞*Wordnet for the Web*, Dan Brickley - an RDF schema based on Wordnet 1.6

[VRP]
☞*Validating RDF Parser (VRP)*, ICS-FORTH, Heraklion, Greece

[Redland]
☞*Redland*, Dave Beckett, ILRT, University of Bristol

[EOR]
☞*EOR (Extensible Open RDF) Toolkit*, Harry Wagner et al, ☞Dublin Core Metadata Initiative (DCMI), 2001

[EDUTELLAPROJ]
☞*Edutella Project*

[KAON]
☞*The Karlsruhe Ontology and Semantic Web Tool Suite (KAON)*

[HAYSTACKPROJ]
☞*Haystack project*

[RDFHOME]
☞*RDF Home Page*, W3C

[INKLING]

☞*Inkling: RDF query using SquishQL*, Libby Miller, ILRT, University of Bristol

[JENA]
☞*Jena Semantic Web Toolkit*, ☞HP Labs Semantic Web Activity

[MOZILLARDF]
☞*Mozilla rdf: back-end architecture*, Christ Waterson, Mozilla Project

[SESAMESAIL]
☞*Sesame RDF SAIL (Storage And Inference Layer) API*

[SESAMEPROJ]
☞*Sesame Project*

[TAPPROJ]
☞*TAP Project*, Stanford University, 2002

[RDFSUITE]
☞*RDFSutie Project*

[JOSEKIPROJECT]
☞*Joseki*, Andy Seabourne, HP Labs

[PYRDFLIB]
☞*Python RDFLib*, ☞Daniel Krech

[RDFLIBTS]
☞*Python RDFLib Triple Store Interface*, ☞Daniel Krech - part of *RDFLib*☞[PYRDFLIB]

[REDFOOT]
☞*Redfoot*, ☞Daniel Krech and James Tauber.

[FOURSUITE]
☞*4Suite*, ☞Fourthought Inc.

[DAMLDB]
☞*DAML DB*, Mike Dean and Paul Neves, BBN.

[OMM]
☞*Ontology Middleware Module (OMM)*, Knowledge & Language Engineering Lab of Sirma, Sofia, Bulgaria.

[ONTOTEXT]
☞*OntoText*, Knowledge & Language Engineering Lab of Sirma, Sofia, Bulgaria.