SWAD-Europe Deliverable 3.11: Developer Workshop Report 4 - Workshop on Semantic Web Storage and Retrieval

Project name:
Semantic Web Advanced Development for Europe (SWAD-Europe)
Project Number:
IST-2001-34732
Workpackage name:
3 Dissemination and Implementation
Workpackage description:
☞http://www.w3.org/2001/sw/Europe/plan/workpackages/live/esw-wp-3
Deliverable title:
3.11 Developer Workshop Report 4
URI:
☞http://www.w3.org/2001/sw/Europe/reports/dev_workshop_report_4/
Author:
Dave Beckett
Abstract:
This report summarises the fourth SWAD-Europe developer ☞Workshop on Semantic Web Storage and Retrieval which was held 13-14 November 2003 at Vrije Universiteit, Amsterdam, Netherlands and attended by 26 semantic web developers from Europe and the USA discussing practical aspects of developing and deploying semantic web storage and retrieval systems.
STATUS:
Completed 2004-01-12, updated 2004-01-13 with workshop evaluation details.

Contents

1. Introduction

The workshop aimed to discuss:

- Implementation techniques and problems met
- Storage models and database schemas
- Aggregation and tracking provenance
- Using RDBMSes for semantic web storage
- Discussion of useful test data and queries for comparisons.
- Implementing RDF datatypes, entailment (from ☞RDF Semantics)

2. Summary

The fourth *SWAD*-Europe workshop was on ☞Semantic Web Storage and Retrieval and held at Vrije Universiteit, Amsterdam, Netherlands organised by ☞Dave Beckett (ILRT) and hosted by ☞Frank van Harmelen from VU.

The workshop participants were mostly technical developers and researchers from a ☞variety of

organisations mostly in industry and education from Greece, UK, Slovenija, The Netherlands and Italy in Europe and from the United States. Most of the participants had practical experience in building and deploying semantic web software and applications. The developers of the two main RDF Java APIs were present, along with the authors of significant APIs in perl, C, python and prolog (all of these were primarily created in Europe).

The two main themes of the workshop were storing semantic web data and retrieving it and the agenda was structured to cover different aspects of both of those items. Other important topics also emerged such as query languages and network access, both related to retrieval and issues with implementing OWL and RDF Schema.

2.1 Storage Issues - The main issues related to storing were with respect to the most common way that the participants described implementing such stores -- with relational databases, especially the freely available MySql and PostgreSQL. The discussion allowed the forming of summary of the general advantages and disadvantages of relational stores compared to non-relational ones that application developers can consider.

These bring up particular issues great care was needed in choosing an appropriate schema, whether to make it of general uses or optimise for the particular application and its requirements. Optimising could also mean not using a relational database at all:

> *"the point at when to optimise from a relational store to a non-relational is when the relationships between the tables get more twisty, more graphy - then a graph solution (triplestore) can work better than a relational."*
> -- ☞*Jack Rusher, Radar Networks*

See the ☞Relational discussion summary for detailed notes of the topics discussed.

Other important storage issues that arose were were with scalability of the current systems available. Stores that can handle 10-20M triples are readily available and the current state of the art is around 40M; the development community is considering the next 10x increase in storage requirements, and their affect on indexing, which has tended to be O(n) for triples. Novel dedicated storage approaches such as in RDFStore were shown to avoid this. The dedicated non-relational stores can outperform the relational ones in such scaling, although the relational databases continue to perform well.

2.2 Retrieval Issues - The main retrieval areas were on the software side with language-specific APIs, querying using a specific query language across multiple implementation languages and network APIs. The authors and experts of several of the most popular RDF software, query languages and network APIs experts were present at the workshop and the initial discussion was on the general aspects of retrieval APIs.

There could be extensions to RDF made via the query language that go beyond the technology built into the RDF model such as providing support for dates, full text, geographical information as well as or instead of RDF graph APIs. This however causes tension over portability, since the query language grows substantially and has to cover application-specific areas outside the standard RDF graph.

The semantic web is all about connecting data, or aggregation and this requires support for taking graphs of triples and unioning them, merging nodes using extra information (also called smushing) such as via OWL inverse functional properties. Not all software has support for managing and performing these operations but it was seen as important to be available in APIs.

The issue of *contexts* in RDF was important (see ☞contexts notes below). It seems that different applications use these as statement identifiers, source document identifiers (where the triples came from), subgraph identifiers or others. There was no consensus on a single definition of these but the two most common useful ones were best defined as *source ID* (where some triples came from, a URI) and *subgraph ID* (identifier for a set of triples in a graph (also called model ID).

The available stores and retrieval mechanisms, along with the query implementations could benefit from ways to perform common benchmarks for testing completeness, scalability and relevance with real data. This could be done with synthetic data but preferably with real (possibly sanitised) data that matches the kind of thing being used today. Some existing work in this area exists, or can be done in a similar way to those for example for full-text query corpuses, testing SQLs.

Query languages at present do not all have good support for the kind of subgraph/model ID contexts described above - some provide one or the other, but not both and it needs to be clear which is available. Query across multiple graphs/multiple subgraphs is also required to aid with the very common semantic web application with data aggregation and management.

2.3 Implementing OWL and RDF Schema - Implementing RDF Schema semantics was reported to be easy except for bnode introduction for datatypes and the infinite container membership properties (`rdf:_1` ... `rdf:_`*infinity*). There has been a little implementation of RDF datatypes - integers, booleans (some), dates but mostly not others. These have at present been added after user demand. For W3C XML Schema Datatypes (XSD) most people use external libraries to handle all the details so rely

on them working to produce correct results - there were some issues of these handling not quite legal data as legal which has caused some testing problems.

There was great interest in understanding OWL layering especially since most of the attendees were more familiar with the lower level RDF and store work. The most common core set of OWL that had the most support for implementation so far was labeled *OWL Tiny* (see the ☞OWL Tiny description below).

> *"OWL Tiny is the subset of OWL-Lite that contains property characteristics and identity relations from OWL Lite. It is the part of OWL that cannot produce conflicts."*
> *-- ☞Steve Harris, University of Southampton.*

Very few people here had OWL DL implementations (this was not the main workshop topic) and mostly were using third-party plugin reasoners or are not covering all of OWL DL at this time.

2.4 Retrieval and Querying - Retrieval by query, that is, RDF query languages has been used successfully by several RDF developers of which several of the authors of the query languages were present at the workshop (SQuishQL, RDQL, SeRQL). The query languages have had a common basis and style (SQL-like). This SQL-style query language familiarity was seen as good for developers, comparing to other query languages nearby such as the XQuery which is more like a programming language for XML.

The group felt that it might be time to feed this into some standardisation work, which the W3C was considering at the time of the workshop and it would be most appropriate to go with a quick and small, well defined activity best described as data access and retrieval covering the basic needs. This would consist of conjunctive queries with constraints. Possibilities beyond this include support for contexts and possibly optionals, although the latter has not been implemented many times.

The query work should exclude updates as that would be beyond retrieval. Several individuals had particular extra requirements but there was no consensus on whether modules or extension methods (like XSLT) were good approaches, especially as this moves implementations apart from the core languages. Returning RDF graphs from the queries (as well as SQL-like result-set tables) was seen as useful and important since this gave RDF in and RDF out which allows pipelining and transformational approaches.

Other RDF query approaches include those based on paths of arcs through the graph, which have been especially found useful when aiming at emitting XML. There have already been interesting work such as OWLP using XQuery, TreeHugger and RDF Twig generating XML from an RDF graph with XPath, and there are other path-like RDF systems also available such as Versa. These were found useful when delivering XML however how much XQuery applies to RDF query languages is not yet clear, although the group noted ongoing work in this area especially looking at reuse of some of the query functions.

The group also discussed the possibilities of querying across graphs and possibly delivering the querying results as a web operation (POST or GET to another URL) which would provide a REST style architectural pipeline of operations. Finally a list of issues for the potential RDF query work was formed, see the section on ☞retrieval by query languages below for the full details.

2.5 Retrieval -- Web APIs and Web Services - Retrieval is also possible by what can be called access protocols, web APIs or web services. These remote APIs can be allowing access to a query language or graph API or both. These remote graph APIs do require a consistent view of the model of store, graph and definition of context to be applied correctly, which might be some indication of a need for some common approaches. The Jena and Sesame groups were both present and noticed they have made pilot experiments in connecting their APIs and will be continuing that.

Web services in the large is much bigger than the *Web Services* brand and includes all services that can be accessed on the web, which have been delivered since the web was built via HTTP and the REST architectural style - this style received a lot of support and using HTTP GET for semantic web data is the cheapest and easiest approach to build a semantic web of data to be stored and retrieved. The Web Service SOAP style RPC or messaging approaches are also possible but are more complex and suitable for larger systems that can hide the interface complexity and networking.

The RDF Net API submission from HP Labs, Joseki were discussed along with DAML Services, now called either OWL Services (OWL-S) or Semantic Web Services (SWS) which are working on semantic web style description of web services allowing both inferencing and services to be connected (this has a background in software agents and planning).

The group also discussed potential useful semantic web services that could be produced, and formed a list of them, which are given below in section ☞Retrieval APIs for software, querying and the web.

2.6 Software APIs - The languages support by the workshop attendees were primarily Java (the main Java RDF API authors were present) and Perl along with some C and a little for python, Tcl, ruby and prolog. There are already lots of software apis (although only some are are mature and complete) and some languages are better than others, although the major web implementation languages are covered well, Java especially. By far the most common question that new RDF developers ask is which bit of

software is best and this has been found tricky to answer, especially as it usually has built-in constraints that do not tend to be expressed until further analysis of the problem has been found (such as requiring one platform or language or access style). SWADE has already looked at this extensively and reported on the issues.

Cross-language APIs could be created, such as DOM for XML but that was seen as a heavyweight design and would be a slow process to produce the RDF equivalent. Interface Definition Languages (IDLs) or equivalent would have to agree on a likely rather small model or face problems matching the existing software APIs. Query languages also provide a defacto cross implementation-language API and this is how relational databases are widely accessed (there are also protocol approaches such as JDBC and ODBC) and indeed this is similar in the RDF developer community, with end-developers generating RDF queries in multiple languages. OWL APIs are emerging slowly although this workshop did not have extensive experience in this area.

The general feeling that there was a state of healthy competition and that it was good, developers were able to switch APIs relatively easily and that the software work in this area in Europe is strong.

3. Workshop Evaluation

This section reports on the the evaluation forms received by the participants of the workshop. 13 forms were handed in.

Breakdown of participants - This is the breakdown of participants by the audience sectors identified in the ☞D3.5 Dissemination and Use Plan, section 1.2.

Sector		
Internet, Web and Open Source developer Community	2	7%
Academic and Research Community	13	50%
Content and Tool Producers	6	24%
Industry and Commerce	5	19%
Total	25	100%

	1 (Agree strongly)	2	3	4	5 (Disagree strongly)	Weighted Average
Store choices for relational/non-relational, schemas and systems	5	5	0	3	0	2.1
Aggregating and scaling semantic web data, benchmarks and test data, tracking provenance	5	5	0	3	0	2.1
Implementing RDF/S semantics and datatypes	5	1	6	1	0	2.8
Supporting retrieval by querying and implementing querying	3	5	3	2	0	2.6
Retrieval APIs for software, network, web services	2	6	5	0	0	2.2
How well did the workshop meet the goals?	5	5	2	0	1	2.0

Other Comments -

- Really enjoyed it, very glad I attended. Inspired to do lots of new work
- Follow-up workshop in 6-12 months?
- Great stuff. All very useful, fruitful discussions
- Nice job
- Found it useful
- Good work convening. Interesting to hear divergent feedback. Interesting for me to hear about

"state of art"

4. Outcomes

The Workshop report will be circulated to the W3C RDF Interest Group and related developer forums. If the W3C charters the anticipated Best Practices Working Group that is currently under discussion, the report will also be offered to that group as materials documenting current practice amongst RDF database developers.

5. Minutes

Thursday 13 November 2003 - 10:00 - Welcome and introduction

☞Dave Beckett (ILRT) and ☞Frank van Harmelen (VU) welcomed the group to VU.
☞Introduction slides from Dave Beckett

☞10:21 - 5 minute introductions by participants/groups

See the ☞list of attendees for the full details of affiliations and projects. This is the order the introductions were made around the table (see the ☞photos from Libby Miller) (☞IRC logs)

- ☞Frank van Harmelen, VU. Working on many projects, with main interest in inference. Just finished an OWL primer book co-written with ☞Grigoris.
- ☞Dave Beckett, University of Bristol. Workshop chair, and main developer of the ☞Redland RDF system and ☞Raptor RDF parser for the ☞SWAD Europe project. Member of W3C ☞RDF Core working group and editor of the ☞RDF/XML Syntax document. Since 1998, maintained an ☞RDF resource guide.
- ☞Mark van Assem, VU. PhD student working on ontology mapping.
- ☞Maarten Menken, VU. PhD student.
- ☞Jack Rusher, Radar Networks. Working on a semantic web browser and LGPL licensed triplestore (not yet released) for a startup company.
- ☞Arjohn Kampman, Aduna. Developer of the Java ☞Sesame RDF system with ☞Jeen
- ☞Jeen Broekstra, Aduna. Developer of the Java ☞Sesame RDF system with ☞Arjohn. Main interest is in querying and inference.
- ☞Andy Seaborne, HP Labs. Working on ☞Jena. and in particular web access via the ☞Joseki Jena RDF server as well as ☞Jena RDQL. Main interest is network retrieval and query and user applications.
- ☞Libby Miller, ILRT, University of Bristol. working on the ☞SWAD Europe project and developed ☞Inkling implementing the SquishQL RDF query language (see ☞Three Implementations of SquishQL, a Simple RDF Query Language), small web service applications, photo metadata and calendaring Co-creator of ☞FOAF with Dan Brickley
- ☞Nick Gibbins, University of Southampton. Working on the ☞AKT project with interest in interactive semantic web applications such as ☞CS AKTiveSpace (which won the ☞2003 Semantic Web Challenge), ☞3store, storage and inference.
- ☞Martin Pike, Stilo. Working with XML and using semantic web technologies especially with the aerospace industry.
- ☞Nick James, Stilo. Working on industrial usage of the technology, a background in databases.
- ☞Heiner Stuckenschmidt, VU. Researcher working on distributed querying.
- ☞Steve Harris, University of Southampton. Working on the ☞AKT project as described above for ☞Nick Gibbins.
- ☞Jo Walsh. Free software developed working with perl and RDF, software 'bots, REST style querying and applications.
- ☞Dave Reynolds, HP Labs. Working on ☞SWAD Europe. and ☞Jena. In particular worked on the Jena1 relational backend and the Jena2 inference, reasoning and OWL support.
- ☞Daniel Krech, University of Maryland College Park. Worked on the ☞rdflib python library which stores via BDB and Zope btrees (before recently joining the mindlab)
- ☞Kevin Wilkinson, HP Labs. Working on ☞Jena and in particular the Jena2 relational store schema.
- ☞Grigoris Antoniou, ICS FORTH (visiting VU). Visiting Frank to work on a textbook for OWL, interested in storage, retrieval, KR and rules.
- ☞Zavisa Bjelogrlic, @semantics. Working on ☞RDFStore and building real commercial semweb

applications.

- ☞Alberto Reggiori, @semantics. Working on ☞RDFStore, querying and RDQL.
- ☞Dirk-Willem van Gulik, @semantics. working on ☞RDFStore with interest in searching support for non-RDF things such as free text querying, queries for geographical information.
- ☞Sander van Splunter, VU. An end-user of sesame and web services based on it.
- ☞Jan Wielemaker, University of Amsterdam (UVA). Main developer of ☞SWI-Prolog and more recently using it for semantic web work; have just added a new triple storage layer. Interested in Prolog based annotation.
- ☞Guus Schreiber, VU. Co-chair of the W3C ☞Web Ontology working group and looking to future work in a best practices WG. Interest is in ontology engineering.
- ☞Lisa Koonts (day 2)

☞10:42 - Identifying key issues from positions, agenda review

(☞IRC)

10:43 Aduna (☞IRC)

Arjohn and Jeen outlined their ☞position paper on work with their ☞sesame Java toolkit. See also their ☞sesame demonstration server

The important issues are query languages, use of path expressions and nesting, dealing with returning the results of querying in a form that can be reapplied to new queries. Schema awareness of stores is seen as very important.

10:47 @semantics (☞IRC)

Alberto presented their ☞position paper on *Indexing and retrieving Semantic Web resources: the RDFStore model*. It is a C/perl toolkit, moving from originally pure perl to have more C. It allows graph manipulation, searching with RDQL, along with free text search of content, under a BSD license. The storage uses BDB with a custom solution with no SQL store used or planned at present. The storage has a very sparse inverted index matrix of indexed triples and properties with optimisations to make the storing and querying be independent of the number of triples. This store has no problems with in-memory joins of millions of triples due to the bitmap compression.

See the slide presentation: ☞Indexing and retrieving Semantic Web resources: the RDFStore model

11:08 UV Amsterdam (☞IRC)

Jan Wielemaker described his ☞position paper *Prolog-based RDF storage and retrieval* on storage in ☞SWI-Prolog. It provides a complete in memory storage solution using prolog atoms directly for speed using with all possible indexing patterns (which is efficient). There is special support provided for `rdfs:subPropertyOf` and transitive closures since prolog is not very good at that. The store uses approximately ~80 bytes perl triples and 3M triples loads into the store in ~10 seconds from an optimised format, rdf/xml is around 10x slower. The next planned future work is looking at OWL which is a challenge.

11:17 University of Bristol (☞IRC)

Dave Beckett presented his ☞position paper *Redland RDF Storage* and the tradeoffs chose from the original development in 2000 to the present. The original choices assumed that disks being slow as the key problem to optimise, other choices are now being considered. Issues include no freetext or schema support at present which is a problem for digital library centred problems.

11:27 HP (☞IRC)

Kevin Wilkinson described the first of the ☞HP position papers outlining how in jena2 there was a complete redesign of the internals adding an enhanced node and graph SPI, changing the graph hierarchies to add for inference and ontology apis. The relational storage schema was moved to a slightly denormalised approach adding the property tables and better support for reification. Jena2 storage uses short prefixes for long URIs, which are always made by the system. Shorter URIs are stored directly in the table. Property tables did add some complexity, in particular to the querying. The result is faster than Jena1 but at 2x the storage cost.

11:37 HP (☞IRC)

Kevin Wilkinson outline the second of the ☞HP position papers *ReMark*, a strawman proposal for benchmarks. It is early days for performance tools and currently this is done by mining RDF graphs and logs of API calls looking for patterns. For example this was done for Haystack which had lots of sequences of getProperty calls for some resource.

11:43 University of Maryland College Park (☞IRC)

Daniel Krech described his ☞MINDlab position paper *RDF Storage and Retrieval with rdflib* which is a Python API providing a persistent stores via BDB and using the (python) Zope object store.

11:47 Radar networks (☞IRC)

Jack Rusher described the ☞Radar Network position paper *Triple Store*. An experienced DBA and

database implementor he saw how the existing database backend for the application wasn't working too well. A new Java triplestore (will be LGPL) was thus developed from scratch based on b-trees, optimising for fast reads, and longer writes. The keys are long integers using composite keys for indexes. The payloads are sorted sets of long integers. The schema also uses string tables however uses compressed tries for short literals. It will be converting to Java New IO (NIO) mainly in order to reduce the seek time in retrieval. Caching helps speed things up such as hot pages of short strings. As a generic triplestore (not RDF specific) it presently does not distinguish between literals and URI references. It scales and performs well.

11:53 University of Southampton (☞IRC)

Steve Harris presented the ☞Southampton position paper *Semantic Web Storage with 3store*. A store was required to deal with about 15m triples and has been developed based on MySQL. The library provides both an OKBC API and RDQL API but the former is deprecated and will be removed. It was originally targeted at was RDF but is now aiming at OWL.

The RDFS support is sound but not complete and there are problems with some RDFS corner cases which are not likely to to be fixed for implementation reasons; see the ☞3store RDF implementation report sent to RDF Core and the (corrected link) ☞3store RDF testcase results linking to problems with test rdfms-seq-representation/Manifest.rdf#test002 and rdfms-seq-representation/Manifest.rdf#test004 as described in ☞[1] and ☞[2]. The Jena team agreed with this problem with the empty document entailing various properties of rdf:_1.

One problem is that the database insert time gets slower over time very rapidly due to the indexing which is possibly O(n). It now handles about 26M triples at ~152 bytes per triple for an on disk size of 5G. (See the ☞live AKTors data). The store also provides free text support and a few datatypes (integer), just those that were needed. It has a remote query API that looks very like Joseki and has a context extension to RDQL as a 4th part of a triple. It is intended to support "OWL Tiny" later and other future work will be on distributed storage and query to allow scaling further. Support for context-based truth maintenance is also an important planned development.

"OWL Tiny" here means the subset of OWL-Lite that contains property characteristics and equality/inequality relations from OWL Lite. It is the part of OWL that cannot produce conflicts.

12:00-13:00 Lunch

☞13:12 - Identifying key issues from positions (continued)

13:12 Stilo (☞IRC)

Nick James and Martin Pike presented the ☞Stilo position paper and noted their interest on using maths with RDF (MathML) and simple rdf storage for presentation. Martin outlined one real problem they have with capturing information about the process of making a plane so it lives beyond the original engineers. Scaling and version control are important and interesting issues here.

13:12 Jo Walsh (☞IRC)

Jo presented her ☞position paper *A Restful RDF/XML Application Server Written In Perl, Its Interfaces, And Its Considerate Approach To Web-Based RDF Query* with interest in APIs using the REST model such as Joseki and querying - "querying returns more than you ask for" using things such as SeRQL. Sees the need for better ways to represent context (see ☞RDF Query and Rules: A Framework and Survey by Eric Prud'hommeaux).

☞13:15 - Store choices: relational/non-relational, schemas and systems

(☞IRC)

Relational

A discussion of the various aspects of using a relational database store to store triples contrasting the old and new Jena approaches. Jena2 uses a specialised support for certain properties and an overflow structure for remaining triples.

3store does not presently have a query optimiser. Steve Harris noted that many databases do not like multiple joins. However this might be an application or modeling problem that could be fixed by changing the graph (the ☞@semantics presentation on RDFStore covers this more). In particular, Dirk-Willem noted that if there are many blank nodes in the data, queries are difficult and it would be better to optimise the rdf to remove the blank nodes. SQL optimisers can't gather statistics effectively from triples bucket structure, so need the extra tables.

The interface layer (JDBC/ODBC) can be a bottleneck to performance (Sesame) and from experience with Joseki (Jena), Andy noted that the JDBC drivers return whole query result set to the client before the first result to the application, not stream as required. The latter is bad for memory usage with big results, no cursors. Steve Harris noted that ☞MySQL at least has a

streaming result C API (that they plan to change 3store to use).

For schema support, 3store uses partly backchaining (at query time), partially forward-chaining (at store) time whereas sesame uses exhaustive forward chaining at store time. Jena2 also supports an efficient reification method allowing multiple reifications of a triple.

Regarding relational database and triple stores, it does not always mean just using a RDB for a triple store, but can mean mapping from an existing RDB with no RDF background.

Related work

- Chris Bizer's ☞D2RMAP that allows "describing mappings between relational database schemata and OWL ontologies".
- ☞SWADE D10.2 Mapping data from RDBMS, Dave Beckett and Jan Grant, ILRT, University of Bristol, 2003-02-18.

Free text and other searching methods

RDF stores (and query languages) do not typically support free text searching or specialised ways of searching the information such as for geographical information. This is critical for some application areas, mixed in with triple-style RDF querying. ☞Lucene has been used along with Sesame to do text indexing, but on top of Sesame, and is not presently part of the system. RDFStore provides free text search with support for multiple languages.

Non Relational

Building a triple store based on non-relational technology was represented by several participants such as those using BDB (not in Jena2 at present) and more sophisticated indexing such as @semantics. These have the advantages of smaller system dependencies than RDBs (slightly different SQLs, optimising needs, features) but are more "bare metal". The indexing is done using hashes (content digests) or using triple identifiers. A brief discussion showed that there were a variety of content digests used across relational and non-relational, MD5, SHA1 and using the top/lower 32/64 bits. As disks are still much slower than processors, there is little difference on modern systems (but MD5 is seen as more common). The non-relational triplestores tend to have better intimate knowledge and use of the RDF details such as schema information but still need to have query optimising, text searching and so on added by hand rather than reusing relational work.

The other kind of non-relational store represented was the SWI Prolog which takes advantage of Prolog's internal atoms to provide full indexing of the triples. This is restricted by memory use, but is efficient and very fast (<1ms) for most queries.

Jack Rusher summarised that to him, the point at when to optimise from a relational store to a non-relational is when the relationships between the tables get more twisty, more graphy - then a graph solution (triplestore) can work better than a relational.

The discussion led into the pros and cons of extensions to RDF querying, for example for operations on dates, geographical queries. Should a query language (in a store) make extra functionality available to querying, should the query language have extensions? (Extensions, modules, profiles). This led to tension in that it makes one query language difficult to be portable to all systems against the requirement for extensions for specific applications. XML Query is also active and important to think about in relation to storing; databases are available now that do XML, XPath and XQuery, can they be used for RDF, can the operators in XQuery be used and reused? (Querying was discussed more later).

Some possible query language extensions that were mentioned:

1. Dates
2. Geographic
3. Full text searching, stemming
4. Internationalisation (multi-lingual)
5. XML XQuery
6. Inferential (RDF, RDFS, the various OWLen)

A partial summary of some of the advantages and disadvantages of RDBs for storing.

Why use relational stores?

- built-in query optimisers
- transactions
- backups
- database management
- plenty of support for relational technology
- not throwing away decades of database work
- very good when the data schema is known in advance and the store can be optimised

why not

- does not work well with rdf schemas
- the query model is very different
- problems with handling general RDF
- optimising the store requires a specialised skill set (DBA)
- JDBC (or is it JDBC+Java) / ODBC is a bottleneck

15:00 - Coffee

☞15:33 - Aggregating and scaling semantic web data, benchmarks, test data.

(☞IRC)

☞15:33 scaling

Scaling means not just scaling the size of the data in terms of the number of triples, but the network issues that come along with this, such as connecting together a lot of stores and dealing with the distributed systems issues. Jeen pointed out the ☞SWAP - Semantic Web and Peer 2 Peer project that is looking into P2P with semantic web data, scaling and sharing them.

Big triplestores are getting sizes of 20M+ today and are much more advanced than even last year. The main problems are in making them 10x larger again, hitting current database limits in terms of support. The next steps up might involve using serious commercial RDBs, for the relational stores. @semantics reports no problems in this range with RDFStore due to the sparse matrix inverted indexing.

☞15:50 aggregation

Aggregation of data separated from large scale data. That can mean the issues related to collecting (semantic web crawling) the data which is similar to standard web crawling in that regard. The special case that more often matters is when the collected RDF triples from multiple documents (graphs) are merged. A simple graph union is relatively straightforward (however see the ☞notes on contexts below) but typically an operation that merges blank nodes according to some extra information is wanted to be performed. This operation has been called *smushing*, where a property of a resource is used as an inverse functional property, with the same meaning as `owl:inverseFunctionalProperty` (IFP). For example, the FOAF community uses `foaf:mbox_sha1sum`, `foaf:workplaceHomePage` and others as IFPs.

This can be implemented in a simple case by rewriting the node identifiers for all triples that mention one of the merged node identifiers, or inventing an (internal) URI such as via a UUID. This rewrite operation is O(n) to the number of triples and making it scale is hard. Another approach that has been tried in an implementation of a FOAF crawler/aggregator by Matt Biddulph makes `owl:SameAs` between the merged nodes and relies on a Jena2 inferencing model to handle the associations. Dave Reynolds reported that there are scalability problems with this due to the implementation which wasn't optimised to handle lots of these equality operations. Others agreed that this is hard and seems an ongoing research problem to make it efficient.

Pointers to some related work and papers:

- ☞tool by ☞Borys Omalyaenko (VU) that analyzes RDF data and finds 'same InvidualAs'
- The ☞SIMILE project is currently looking at this problem in matching `dc:creators` in image catalogues.
- ☞Managing Reference: Ensuring Referential Integrity of Ontologies for the Semantic Web In Proceedings 13th International Conference on Knowledge Engineering and Knowledge Management (EKAW'02), Siguenza (Spain).

Related problems to aggregation are de-merging graphs if they are updated or withdrawn. This operation of tracking the source of triples has been approached in a variety of different ways, but seems a critical need, at least to know the original (document URI) source of a triple. This is also related to contexts, see the ☞contexts notes below.

☞16:00 benchmarks and test data

Kevin introduced the background for the second of the ☞HP position papers *ReMark* which was to have a standard set of datasets to do benchmark and comparison testing between storage and query systems. The discussion then covered what kinds of data would be needed to match real life examples. Several people offered to make datasets available but there were issues with ownership and privacy / data protection with some data. The test datasets would need to be representative as well as freely usable, in order that true comparisons could be made with all. There was concern over synthetic datasets versus real ones since the characteristics required were not entirely clear.

Some typical types were identified:

- Ontologies and RDF schemas (lots of classes and properties, few assertions)
- Dublin Core records, commonly seen from digital library projects; few numbers of properties, lots of flat records, little or no schema information
- Large literal records - from working with free text sources, having larger literals than are typically seen either full document text or several kbytes. These work poorly in stores that do not have special support, and are generally accessed with free-text style queries (compare Z39.50). Synthetic data generated here would have to match the natural language(s) so that the queries could remain realistic.
- Rich and linked data - descriptive, geographic, syndication, relationships. Such as provided by FOAF or the data models behind the @semantics applications.

Some other data requirements were needed such as large files. There are already existing single RDF/XML files over 700Mbytes (Musicbrainz) on the web and off the web, into gigabytes. A snapshot of one of these large files would make good test data for dealing with such sizes. Otherwise, the test data files should be moderate size (low Mbytes) to be sufficient to give a good test.

In addition to test data for benchmarking, there were also needs for test data for queries - standard queries against standard test data. This should also include logs of typical (real) queries that are made against semantic web stores in certain applications. Jeen Broekstra and Nick Gibbins already have pending actions from the PSSC workshop at ISWC 2003 to make a small set of benchmark queries to give a low-level starting point for comparison of systems. Jeen would provide them over Wordnet and Nick over a subset of the CS AKTors dataset, to be announced on the ☞www-rdf-rules list.

Kevin noted that the Jena2 team has been working with query logs from the ☞Haystack application to identify patterns. Presently they have noted that it is used in an object style; a pattern of queries for property/values of the same resource.

Pointers to related work on data and test cases:

- ☞LeHigh University Benchmark (LUBM) described in *Benchmarking DAML+OIL Repositories, Yuanbo Guo, Jeff Heflin and Zhengxiang Pan*, proceedings of ISWC 2003. ☞PDF
- ☞RDF Test Cases Jan Grant, Dave Beckett (editors), W3C Working Draft 10 October 2003
- ☞OWL Web Ontology Language Test Cases, Jeremy J. Carroll and Jos De Roo (editors), W3C Candidate Recommendation 18 August 2003
- ☞RDF Query Test Cases, Alberto Reggiori describing an RDF vocabulary for RDF query test cases inputs, outputs and manifest.
- ☞RDF Query and Rule testcase repository, Alberto Reggiori and Andy Seaborne.
- ☞Data Sets Section 7 in ☞SWADE D10.1 Scalability and Storage: Survey of Free Software / Open Source RDF storage systems, Dave Beckett, ILRT, University of Bristol, 2003-02-17.
- ☞DAML Data Sources, Mike Dean, DAML

☞16:18 - Contexts, provenance and tracking

This session was on dealing above the triple level using some other grouping or identification method. Several of the implementations allow recording of additional addifiers when individual triple or groups of triples are added to a store. These include allowing a *model ID* to be given; which amounts to allowing multiple graphs inside a single database (typically). The ID can be an internal identifier or a URI. Another identifier that can be assigned is the *source* of the triples content, usually the URI that the content was retrieved from, for stores that are dealing with aggregation. The source URI can be used as a model identifier but these should not be confused. These are the typical choices for the "4th part of the triple" (quad) and usually called context. This term was deprecated by the participants in favour of either model IDs, source IDs or grouping IDs where the identifier is only used internally for identifying subgraphs.

Summarising what some of the applications represented used:

- 3store - Source URIs used as Model(subgraph) IDs
- Jena2 relational store - Model(subgraph) IDs
- RDFStore - Model(subgraph) IDs (and Statement IDs separately)
- Redland - either at user's choice

These identifiers are useful to be able to assign when adding triples into a store and also to be able to get

via APIs including via queries. Applications of the IDs are for tracking where the triples originally came from, distinguishing triples in subgraphs inside single stores and for truth maintenance. The latter was discussed and solutions varied from fine-grained triple recording (sesame) to source-based (3store).

Some applications have made extensions to the existing triple-style query languages such as RDQL to add this fourth argument as a context identifier. Given the inconsistent use of this in the backend stores, it was seen as essential to be able to distinguish between a model(sub graph) ID, source ID in standardising a query language API. It was felt that both were necessary by most, but not all participants. Querying across contexts (subgraphs) was also seen as required.

Pointers to related work:

- ☞ Representing Contextualized Data using Semantic Web Tools in ☞proceedings of the *Practical and Scalable Semantic Systems workshop* held at the ☞ISWC conference.

16:51 - End

Friday 14 November 2003 - ☞09:25 - RDF/S Semantics, datatypes, OWL

(☞IRC)

This session discussed implementing RDF schema, RDF datatypes and OWL. There seemed to be agreement that there was no problem implementing RDF schema except for two points - the bnode introduction for datatypes and the infinite container membership properties (Jena2, 3Store). Sesame implements all the rules in the RDF model theory. RDF datatypes is not required to be implemented, since it is a general mechanism that several datatypes can fit into. Several applications have at least some support for integers, some for booleans, some for dates. Jena2 uses the Xerces W3C XML schema datatypes library so has a richer support and finds that users want datatypes (typed objects) in and out at the Java level but do not want to do datatype subclassing at present. Sesame handles some the datatype entailment in the parser (such as rule rdfD2) and comparison operators in the SeRQL query language and has received requests for integer and date comparison operations support. RDFStore needs datatypes for searching and some inferencing, and especially for geographic information or complex operations on dates.

There was little user need at present seen for user defined datatypes, this is related to the problem that there is no way to get a URI for a user defined datatype, the XML Schema WG/TAG has not decided how to do this yet. The minimum set of the XSD builtins plus a few specialised datatypes covered most datatype feature requests. XSD datatypes and others can have validation defined; at this point there can be a datatype violation in RDFS - the only place such a thing can occur. Jena2 implements this with an API level validate() call, 3store tries to be generous in accepting bad datatypes, but does fail some. This is related to the RDF test on rejecting " `10.0` " as an invalid xsd:integer (no whitespace). Sesame would reject that however does trim spaces on output, and may add a three-way ignore/normalize/validate switch in future. This area has tension between between wanting to do the right thing with bad data and handling precise semantics of what is and is not a legal datatype lexical form.

There is now emerging support for the XML query XQuery in standard databases and XML systems and some experimentation in applying the XQuery functionality to semantic web data. In May 2003 an ☞RDF and XMLQuery BOF was held at WWW2003 in Budapest discussing how to use and re-use the XQuery work such as the functions and operators. Since then there have been some systems announced building RDF and OWL work on top of XML systems:

- *Ontology Web Language Program (OWLP)* parser for OWL DL based on the ☞GALAX XQuery system, both from Bell Labs (☞ OWLP 0.5 announcement)
- ☞TreeHugger by Damian Steer making an RDF graph look like an XML document allowing XQuery, XPath, XSLT etc. to be applied on it. There was some interest in the room to investigating this area in terms of reusing as much of the XQuery work, but it may not be all applicable since XML trees are quite different from RDF graphs and OWL ontologies.

☞09:53 - OWL and layering

(☞IRC)

Frank van Harmelen asked about how much of OWL was being implemented and gave a general introduction to the OWL layers - OWL Lite, DL and Full. This participants were mostly dealing with the RDF and storage layer but had some experience with OWL work. They are not all reasoning experts and typically provide support for plugin reasoners. Those present described that there was a core set of OWL that was had the most support for implementation, labeled here *OWL Tiny* (the 4th OWL).

"OWL Tiny"

Using the terminology of the OWL constructs from the ☞OWL Lite Synopsis in ☞OWL Reference), OWL Tiny was described approximately as the following subset of OWL Lite:

RDF Schema Features
Class, rdf:Property, rdfs:subClassOf, rdfs:subPropertyOf, rdfs:domain, rdfs:range, Individual
(In)Equality
owl:sameAs, owl:differentFrom
Property Characteristics
owl:inverseOf, owl:TransitiveProperty, owl:SymmetricProperty, owl:FunctionalProperty, owl:InverseFunctionalProperty
Annotation Properties
rdfs:label, rdfs:comment, rdfs:seeAlso, rdfs:isDefinedBy
Header Information
owl:ontology, owl:imports (**unsure**)
Versioning
versionInfo, priorVersion, backwardCompatibleWith, inCompatibleWith, DeprecatedClass, DeprecatedProperty. (**unsure**)

With the following constructs not included from ☞OWL Lite (for comparison):

(In)Equality
owl:equivalentClass, owl:equivalentProperty, owl:allDifferent.
Property Type Restrictions
owl:allValuesFrom, owl:someValuesFrom.
Restricted Cardinality
owl:minCardinality, owl:maxCardinality, owl:cardinality.
Class Intersection
owl:intersectionOf
Datatypes
DatatypeProperty

(And none of the extra ☞OWL DL and Full constructs)
The following experiences were recorded from implementation:

- The main advantages are that there are no class definitions and no restrictions
- The trickiest part to implement is in/equality
- Jena2/HP, 3Store/Southampton plan to or have implemented this subset

☞09:54 OWL Lite

(☞IRC)
The major additions of OWL Lite over tiny were described as the owl:minCardinality, owl:maxCardinality, owl:cardinality (all 0 or 1 in OWL Lite) and owl:allValuesFrom, owl:someValuesFrom. The latter two were seen as the tricky parts again due to issues with implementing equality.

Dave Reynolds reported on the Jena2 OWL work. He described what Jena2 implemented as the "OWL Lite fragment of OWL Full" - most of OWL Lite but not restricting, for example, that classes cannot be instances as required by OWL DL. Jena2 also adds two OWL DL/Full terms after user request - owl:hasValue although it has bad interactions with inverseOf and cardinality constraints beyond 1 (OWL Lite having only 0, 1). This was starting to get hard while having equality support in combination. The request that required the cardinality restriction was with respect to validation of datatype properties (in OWL Lite). There seem to be two sets of users - logicians and those using OWL as a near-schema validation language. So far, there has been no user presure to implement owl:allValuesFrom, owl:someValuesFrom (OWL Lite).

10:05 OWL DL

DL and Full add more constructs above OWL Lite -- see ☞Incremental Language Description of OWL DL and OWL Full). DL also restricts how the constructs are used; in particular the type separation of classes, individuals and properties. In this group there was not significant experience with these at this time (this was not an OWL implementors workshop). As mentioned above, plugin reasoners were being used to take advantage of existing reasoner work by others and using off-the-shelf systems. It was estimated that it takes three months to build a complete reasoner suitable for OWL DL using current literature, when familiar with the field. See also the ☞OWL Implementations recorded by ☞WebONT.

10:15 OWL Full

The remaining OWL layer does not impose the restrictions of OWL DL on the ontologies, with impact on the reasoning tools that can be applied to it. In informal terms, you are "allowed to mess with the furniture" (Dave Beckett) such as using classes as instances.

10:25 Extensions to OWL

The workshop briefly looked at the ☞WebOnt Issues List to consider postponed items. The two major ones that seemed significant were that OWL cannot describe cousins (although there may be a solution that via some other method not captured in the notes) (☞ Compound Keys) and cross properties although the other postponed issues such as justifications and quoting are likely to return as querying and rules work continues.

10:30 - Coffee

☞10:50 - Retrieval by query languages.

(☞IRC)

SWAD Europe has several activities in this area. In a survey ☞7.2 Databases, Query, API, Interfaces: report on Query languages, Libby Miller found more than 12 RDF query languages available. SWADE has also been developing use cases and implementations, examples from this work. See also Libby's ☞RDF query tests.

At the ISWC2002 in Sardinia after a query BOF, a page of ☞query use cases and examples was developed by Alberto Reggiori and Andy Seaborne, allowing additions by anyone including a comparison of query languages. Andy developed a format for ☞Recording Query Results that are the result of an SQL-like RDF query languages, that is, variable bindings. Nearby are the raw ☞RDF query test data.

At WWW2003 in Budapest, an ☞RDF and XMLQuery BOF was held considering how XML Query could be used for some future work on standard RDF query languages. The trend remains that most RDF query languages so far are SQL-like/familiar or frendly rather than looking like XQuery.

Alberto argued that for querying, it would be best to concentrate on the basics - conjunctive queries, constraints and support for context or provenance. @semantics also had requirements for querying to support very good text searching, stemming, multilingual support as well as specialised searching for some of their geographical information. This would possibly be more suitable for an extension to a core query language.

Jeen Broekstra and Arjohn Kampman described ☞SeRQL and that it was based mostly on other RDF query languages rather than SQL, although OQL was an influence. One key design feature of SeRQL is that as well as variable bindings as a result, an RDF graph can be `CONSTRUCT`ed. Thus the output of SeRQL queries can be composed with other queries rather than going from RDF graph to variable bindings and losing the graphs. SeRQL also always has built-in RDF/S entailment, and it is seen as important to Aduna that an RDF query language should support this (required, not optional).

Dirk-Willem noted that RDQL (and similar languages) have advantages of familiarity to existing SQL developers, and the reuse of the ODBC and JDBC infrastructure. It can also expose and use lots of the other parts of the SQL work such as transactions, views etc. The disadvantage of a new RDF query language is that it would be unfamiliar to the developer or customer.

In addition to the SQL-style, there have been experiments with path-based query languages, most recently ☞TreeHugger providing XPath over RDF graphs. There has not been much other work recently, although paths and conjunctive queries/joins have a clear mapping. XPath can probably fit on top of RDQL.

Several of these languages have also been used to transform to make XHTML using XSLT. Other approaches in this area include ☞Graph Stylesheets (GSS) in IsaViz by Emmanuel Pietriga. CWI has also worked on this in the ☞Cuypers Multimedia Transformation Engine over Sesame to transform the content into web presentations via XSL and Prolog rules.

The workshop discussed the core ideas for the smallest useful work on a query language that could be done in this area. This might mean that some of the needs of the applications represented may not be met and so an extension mechanism, modules were considered. This does bring up interoperability problems but might be required as an escape hatch to allow application customisation.

It was mentioned how queries should be able to use multiple graphs/models, sometimes called contexts. This could be mapped to a fourth part of a triple in a match, which is outside the RDF model (but to compare, SQL views are outside the original SQL model). 3store allows some access to this kind of information in RDQL queries but this has to be made consistent across all applications. Others may want source URIs, context URIs or other identifiers to be either specified in querying and/or returned

with the result set.

A summary of issues that were mentioned in the discussion

- It must be simple format, short timescale and have clear goals (no research)
- Not calling it a query WG would help; prefer access and retrieval protocol
- Simple conjunctive query (join) of triple patterns
- Simple constraints of those queries
- Use case: generating webpages from RDF
- RDF/S entailment required
- Full text searching - substrings, stemming, I18N
- Take care with syntax - <, ?, # are hard to escape in URIs and/or carry semantics outside the semweb world
- Composition of query output (rdf in/out like tables in/out for SQL)
- Need compactness of syntax when using URIs
- Possibly could define a small set of (namespace) prefixes
- Different usecase may require different query languages
- An XML query syntax is not a primary requirement (like XPath/XQuery)
- Return set format - RDF triples, a table, defined at query time or negotiable. Joseki allows a bunch of alternatives:
  - single subgraph
  - results
  - result set in RDF
  - template (like SeRQL)
- Asking to evaluate some query with RDFS entailment (requiring that is available) or other semantics.
- A trigger action such as a URI GET when a query finishes/succeeds. Allows chaining queries between independent services (Alberto)

☞11:36 access protocols / web apis, web services

(☞IRC)

Andy Seaborne introduced the ☞RDF Net API submission to the W3C with Graham Moore. It is a SOAP interface to an RDF store and addresses how to access large RDF stores remotely. Query seems a natural paradigm for accessing RDF graphs; the query language is a triple with holes in it that maps quite easily onto existing toolkits. The service description does not preclude a REST architectural style, and indeed Joseki provides that. There was general interest in small and easy to implement access protocols.

Related work

- Semantic Web Services Initiative (from DAML Services, OWL-S)
- ☞paper by Marta Sabou from ISWC 2003 on web service description of services such as those provided by Sesame.

12:00-13:00 Lunch

☞13:08 - Retrieval APIs for software, querying and the web (such as web services)

(☞IRC logs)

The group considered what kind of semantic web services might be wanted dealing with or for semantic web data, queries or access. What kinds of protocols should be used, does it need a semantic web specific service technology such as DAML-S/OWL-S? The discussion started from the use cases to direct the kinds of services that might be needed. Most of the discussion was starting from the potential services and their implications on the access model.

Most of the services were not a multi-stage series of protocol requests or complex interactions. Several of the services could be implemented by sending a query in some concrete query language, which is how many of the participants currently express their web service APIs. This has been the current trend, away from remote API level services in an RPC style. AKT has effectively dropped OKBC/WS work in favour of sending RDQL although Sesame still uses the sesame client library.

Services ranged from the Web Service (SOAP, RPC style) to the REST architectural model which is the core of the Web design. Many people supported REST as the preferred approach and have made several REST based RDF API web services (Jo Walsh reports on this in her ☞position paper) and Libby Miller has made several as demonstrators for SWADE.

The services need to have a consistent view on certain things such as description of a store, a graph or a context (such as discussed earlier) in order that these can be used in the services. This might be partially a call for a common underlying API or even a higher level one, although as in the relational

database world, a query language can do this (as well as having software APIs such as ODBC and JDBC).

On the common API front, the Jena and Sesame projects noted they have experimented with connecting their work at the triple store and inferencing levels (Graph API for Jena2, SAIL level for Sesame). Most other interconnections have been made using web services over SOAP, Java RMI or alternately using the web with REST HTTP interfaces. The latter (RMI and HTTP) are the most lightweight. HTTP was particularly favoured since it is the web and has the best support, ease of use, lowest cost and interacts well with other parts of the web such as proxies and firewalls without "tunneling" them.

The services also benefit from description and although WSDL existed and can be used, there was felt to be a need for much simpler one that connects well to the semantic web.

Some potential semantic web services:

- ontology mapping
- mapping the terms in the query to terms in your own framework
- transforming data
- transforming a query
- thesaurus services
- multilingual mappings of concepts (thesauruses)
- query expanders - returning subtypes, more properties
- request-reply interaction such as an interactive reasoning process
- a reasoning service like Jena2's plugin reasoning API, exposed as a web service
- rules engine services (higher up the semantic web stack/wave)
- a trust expander service
- operating business rules
- wandering round a remote conceptual map service
- concept explosion service
- pathfinders
- visualisation services
- browsing (examples given included AKT frame-based HTML browser ☞brownsauce ☞foafnaut)
- queries with triggers to take another request
- interconnected services in semiautomatic chains
- publish and subscribe services for a queryable aggregator (send a query, get new result sets when they drop into the slots)
- asynchronous messaging
- monitoring frequently updating sources

☞13:47 APIS for software

(☞IRC logs)

The workshop participants contained a core set of the developers of the main programming APIs for RDF and in particular for the two major Java APIs, Java and Sesame. Dave Beckett described that as the maintainer of a big RDF guide, he receives lots of questions about RDF and what APIs are best. The questions typically can be best answered by first finding out the target development language, usually the major consideration. This leads to the question of whether there should be one recommended (by somebody) API per language, or even one API cross language, such as is done with DOM for XML (although DOM is only one of several XML APIs). Some in the workshop suggested that too much cross-language consistency was not a fruitful expectation - such as between declarative, object based or functional where the programming styles might be just too disparate. Redland however has provided the same API across a variety of web languages for several years without problems.

The DOM API was seen as possible evidence of over-engineering in an API and thus a simpler approach might be the best minimal starting point. APIs for RDF at the simplest level match the generality of the RDF triple model, but when particular applications and data sets are targeted, different API support may be needed. In the relational database world, this led to the construction of standard query languages so that the separation of the database schema and the code was done via a third language. In addition protocols for database access were developed (ODBC, JDBC) that allowed more direct access and easier remote access, leveraging on top of the query language. See also ☞SWADE D7.2 Databases, Query, API, Interfaces: report on Query languages, Libby Miller, ILRT, University of Bristol, 2003-04-01

The Sesame and Jena groups have at various times discussed their connecting their Java APIs, and have tested implementing some of this, connecting Jena and Sesame at the store or inferencing API level. However it is not so easy to emulate the one on top of the other and requires not just API mapping but translation between object types in the respective APIs.

Most of the RDF APIs work with the RDF triple as the fundamental concept of the API but there is

also the graph style, called the node or resource-centric approach where the application can walk or follow arcs, rather than in terms of triples. The downside of this is that RDF cannot describe a node or arc alone, only as part of triples so this could lead to illegal or partial graphs if used to construct them in this manner. The triple API may be intuitive but can be easy to get lost in, when there are lots of triples. Several systems provide both the triple and node-centric API (Jena, Redland), but the vast majority have the triple API, since that is the defined RDF model.

The Sesame group expressed a desire to reuse the graph path finding traversal code in Jena if it was separately packaged and if so could be used as a common API.

Beyond APIs for RDF alone, there are emerging OWL APIs such as the WonderWeb OWL API and the OWL API in Jena allowing manipulations in terms of the OWL concepts, and checking the constraints of the appropriate OWL layers, possibly performing inference. Both of these are using a triple-based RDF API below. The sesame group is also planning an OWL API very soon. This workshop was not specifically for OWL implementors so there are other OWL APIs or systems available also that provide yet again different APIs in other languages; there is no common API emerging here, apart from being based in terms of the OWL abstract syntax concepts.

The group discussed the pros and cons of the API approaches (triples, resource centric, query languages, common APIs) and whether having multiple APIs is a big problem. This is something already reported on in SWADE in ☞D7.1 RDF API requirements and comparison, by Jan Grant, ILRT, University of Bristol, 2003-02-27. It was noted that at least between Jena and Sesame, people have found migration in either direction pretty straightforward so was not as hard as first may seem. There are concerns at least with some APIs about the level of maturity of the software, support and longer term maintenance.

Other possibilities rather than using one API would be to provide portability at the store level; standardising on the relational database schemas for particular stores. The position papers, the suggested readings for the workshop and the discussions showed some commonalities in schemas but a single complete and general one has not yet emerged. It might be that such a schema would only exist for such purpose rather than being of good applicability for multiple applications.

Current RDF and semantic web standards work has been in describing the documents or data rather than standardising the processes or APIs that deal with them, although this is starting to change with work like Joseki, in semantic web services and related areas. Work on an API activity such as a web service interface or neutral language independent interface description language (an IDL) would have to be very tightly specified and small to be of general use.

The current state is mostly a healthy ecosystem of competitive APIs encouraging each other to get better or address different concerns. This was a concern of some participants -- should they wait for the eventual winner or are the multiple choices stable and safe to develop on.

To help discuss comparing APIs, ☞Marta Sabou presented her work on identifying common operations in several RDF tools using four basic types of operations - query, remove, retrieve and add. This was used to form a visualization of the overlap between Jena, Sesame and KAON showing a fair amount of common functionality and little unique to one. See ☞Like API Like Web-Service: Building Domain Ontologies from API (PDF) by Marta Sabou, Department of Artificial Intelligence, Vrije Universiteit Amsterdam, The Netherlands.

14:10 - Coffee

14:25 - Summary. What is missing and what still needs standardising.

A general discussion of the current state of the standardising work at the W3C. The RDF and OWL documents were both nearing proposed recommendation at this time. In terms of future work, this is still under consideration but after earlier discussions, the lightweight query or data access seemed to have some support but not a lot of new work needed right now. The research continues on other parts of the semantic web picture but most of the participants were working on applications and practical use of the technology.

Finally a summary was given of the upcoming SWADE work that is planned including workshops on internationalization and accessibility with one already planned for XML Europe in Amsterdam, April 2004 on interoperation with Topic Maps, possibly with one on calendaring after that.

15:10 - End of Workshop

6. Attendees

(Order by family name)

1. Grigoris Antoniou, ☞ICS FORTH (visiting Vrije Universiteit Amsterdam), Heraklion, Crete, ☞antoniou@ics.forth.gr, Greece working on a textbook for OWL with Frank van Harmelen.
2. Dave Beckett, ☞Institute for Learning and Research Technology (ILRT), ☞University of Bristol, ☞dave.beckett@bristol.ac.uk,UK working on ☞Redland RDF system and and ☞Raptor RDF parser for the ☞SWAD Europe project.
3. Zavisa Bjelogrlic, ☞ASemantics, ☞z@asemantics.com, Slovenija working on ☞RDFStore (☞FOAF)
4. Jeen Broekstra, ☞Aduna (formerly Aidministrator b.v.) Julianaplein 14b, 3817 CS Amersfoort, ☞jeen@aduna.biz The Netherlands working on ☞Sesame
5. Nick Gibbins, ☞Department of Electronics and Computer Science, University of Southampton, ☞nmg@ecs.soton.ac.uk UK working on ☞3store for the ☞AKT project.
6. Steve Harris, ☞AKT project, ☞Department of Electronics and Computer Science, University of Southampton, ☞swh@ecs.soton.ac.uk, UK working on ☞3store for the ☞AKT project.
7. Nick James, ☞Stilo, North Quay, Temple Back, Bristol, BS1 6FL, ☞Nick.James@stilo.com, UK
8. Arjohn Kampman, ☞Aduna (formerly Aidministrator b.v.) Julianaplein 14b, 3817 CS Amersfoort, ☞arjohn.kampman@aduna.biz, The Netherlands working on ☞Sesame
9. ☞Lisa Koonts
10. Daniel Krech, ☞Mindlab Semantic Web Research Group (MINDswap), University of Maryland College Park, ☞eikeon@eikeon.com, USA working on ☞rdflib.
11. ☞Maarten Menken, ☞Vrije Universiteit Amsterdam (VU), ☞mrmenken@cs.vu.nl, The Netherlands
12. Libby Miller, ☞Institute for Learning and Research Technology (ILRT), ☞University of Bristol, ☞libby.miller@bristol.ac.uk, UK working on many things for ☞SWAD Europe. Co-creator of ☞FOAF with Dan Brickley
13. Martin Pike, ☞Stilo, North Quay, Temple Back, Bristol, BS1 6FL, ☞mp@stilo.com, UK
14. Alberto Reggiori, ☞ASemantics, @semantics S.r.l, Milan Office, via Monteggia 98 (internal 7), 21014 Laveno Mombello (Varese), ☞alberto@asemantics.com, Italy, working on ☞RDFStore (☞FOAF)
15. Dave Reynolds, ☞HP Laboratories, Bristol, ☞der@hplb.hpl.hp.com, UK working on ☞Jena and ☞SWAD Europe.
16. Jack Rusher, Radar Networks, ☞jack@rusher.com, working on an LGPL triplestore (not yet released).
17. Andy Seaborne, ☞HP Laboratories, Bristol, ☞andy_seaborne@hplb.hpl.hp.com, UK working on ☞Jena.
18. ☞Guus Schreiber, ☞Vrije Universiteit Amsterdam (VU), Co-chair of the W3C ☞Web Ontology working group.
19. Heiner Stuckenschmidt, PhD student, ☞Vrije Universiteit Amsterdam (VU), ☞heiner@cs.vu.nl, The Netherlands
20. Mark van Assem, PhD student, ☞Vrije Universiteit Amsterdam (VU)
21. Dirk-Willem van Gulik, ☞ASemantics, @Semantics s.r.l., Leiden Office, Janvossensteeg 37, 2312 WC Leiden, ☞dirkx@asemantics.com The Netherlands, working on ☞RDFStore (☞FOAF)
22. ☞Frank van Harmelen, Professor of Knowledge Representation and Reasoning in the AI Department, ☞Vrije Universiteit Amsterdam (VU), ☞Frank.van.Harmelen@cs.vu.nl, The Netherlands working on many projects. Member of the W3C ☞Web Ontology working group and co-editor of ☞OWL Web Ontology Language Overview
23. ☞Sander van Splunter, ☞Vrije Universiteit Amsterdam (VU), ☞sander@cs.vu.nl, The Netherlands
24. Jo Walsh, ☞University of Openness, Limehouse, London, ☞jo@abduction.org, UK
25. Jan Wielemaker, ☞Social Science Informatics (SWI), University of Amsterdam (UVA), ☞jan@swi.psy.uva.nl, Roetersstraat 15, 1018 WB Amsterdam The Netherlands working on ☞SWI-Prolog.
26. Kevin Wilkinson, ☞HP Laboratories, Palo Alto, California, ☞kw@hplwkw.hpl.hp.com, USA working on ☞Jena.