SWAD–Europe Deliverable 7.3: Databases, Query, API, Interfaces Public release of reference implementation of an RDF API

Project name:
>   Semantic Web Advanced Development for Europe (SWAD-Europe)

Project Number:
>   IST-2001-34732

Workpackage name:
>   7: Databases, Query, API, Interfaces

Workpackage description:
>   ☞http://www.w3.org/2001/sw/Europe/plan/workpackages/live/esw-wp-7.html

Deliverable title:
>   Public release of reference implementation of an RDF API which may build upon existing implementations.
>   Open Source/free software.

URI:
>   ☞http://www.w3.org/2001/sw/Europe/reports/rdf_api_impl/

Author:
>   Libby Miller

Abstract:
>   Public release of reference implementation of an RDF API which may build upon existing implementations.
>   Open Source/free software.

Status:
>   Document started 2003-08-19, this version 2003-11-05 - completed report.

Introduction

There are two kinds of APIs in RDF: API access to an RDF graph, and API access to RDF queries. The former have been extensively documented and implemented; the latter have this far received less attention.

This document describes a small Java API and covers the first of these aspects. ☞A companion document [1] describes the query language and api used in the same implementation. These implementations are available as a ☞download [2] the ☞W3C license [3]. Other relevant documents include ☞testcases for RDF query [4]; ☞RDF API survey [5]; ☞RDF Query language survey[6].

The aim of this document is to summarise the main access methods for RDF data used and provide a brief description of each. A fuller description of the API is available in the ☞javadoc[7] and in the ☞test and ☞app directories of the software distribution. A ☞download page is available. The software is released under the ☞W3C license.

History - This API and query language implementation is the result of many iterations of development and related discussions. Earlier versions were based on ☞an RDF API by Sergey Melnik[8], which in turn was based on ☞SiRPAC[9]; this version is closer to Dan Brickley's ☞RubyRDF[10]. The query language ('Squish', ☞BNF[11]) is based on R. V. Guha's ☞rdfDB query language[12].

Main features - The aim has been to make a small, simple, and where possible, memorable API, which can be used to create useful applications quickly.

The emphasis is on the query language as the main interface to the data, but a robust API is an essential part of the underlying infrastructure.

Small

☞Jena [13] is an impressively comprehensive Java software toolkit for RDF. The aim of this software is to be smaller although with much less functionality.

Simple

Few methods are required to build applications. See ☞below and in ☞test/ and ☞app/ for sample applications.

Node-centric and triple-centric, but mostly query-centric

It is possible to use node.getProperty(), node.getObject(), etc, but also access the information as a list of triples instead - graph.ask(s,p,o).

However the emphasis is on query as the main interface to the RDF data, for example (from ☞GraphTest.java):

```
String sha1mbox=Util.sha1(mbox);

String
uri="http://swordfish.rdfweb.org/photos/genfiles/ilrt/5018518518518519.rdf";

String query= "SELECT ?pic, ?thumb, ?x  "+
" from "+uri+" "+
" WHERE "+
   "(foaf:depicts ?pic ?x) "+
   "(foaf:mbox_sha1sum ?x '"+sha1mbox+"') "+
   "(foaf:thumbnail ?pic ?thumb) "+
" USING foaf for http://xmlns.com/foaf/0.1/ "+
" dc for http://purl.org/dc/elements/1.1/ ";

java.sql.ResultSet r=gr.askSquish(query);

        while(r.next()){
        System.out.println("pic "+(String)r.getString("pic"));
        System.out.println("thumb "+(String)r.getString("thumb"));
        }
```

produces

```
    [junit] PARSING with Rio with base
http://swordfish.rdfweb.org/photos/genfiles/ilrt/5018518518518519.rdf
    [junit] pic
http://swordfish.rdfweb.org/photos/2003/03/13/2003-03-13-Images/2.jpg
    [junit] thumb
http://swordfish.rdfweb.org/photos/2003/03/13/2003-03-13-Thumbnails/2.jpg
```

Retains provenance

Each RDF Graph has an ID (the url it came from, or some other uri where not present), and each Node object has a reference to its containing Graph (default where not present). SQLGraphs retain the provenance information for updating and removal; however for SQL queries, the provenance is not returned.

SQL and in-memory implementations

Postgresql and MySql implementations. See ☞GraphTest.java, ☞UnionGraphAddRemoveTest.java, ☞UnionGraphQueryProvenanceTest.java, ☞UnionGraphSameStatementTest.java, ☞UnionGraphSquishQueryTest.java, ☞SQLGraphTest.java, ☞Squish2SQLTest.java.
  There are also various ☞demonstrators[14].

Internationalization

Works with UTF-8 character sets for inmemory and Postgres versions; for MySQL, 4.1 is required for UTF-8 support. See
☞QueryI18NTest.java, ☞QueryI18NTestMySQL.java, ☞QueryI18NTestPostgres.java.

There is also an in-memory ☞demo[15].

RDF Features and API Considerations

The following is a brief description of the API functionality using the headings from ☞SWAD-Europe Deliverable 7.1: RDF API requirements and comparison by Jan Grant. A fuller description of the API is available in the ☞javadoc and in the ☞test and ☞app directories of the software distribution.

The RDF Graph - Graph operations

The main Graph classes are Graph, UnionGraph (both in-memory implementations) and SQLGraph. Graph is the simplest case, used for loading RDF from a url or a file as RDF/XML or Ntriples. Graphs may then be added to UnionGraphs and SQLGraphs.

Creating an in-memory graph and querying it - Create a Graph and populate it:

```
Graph gr = new Graph(uri, Util.RDFXML);
gr.load();
```

Ask it for subjects of type rdf:Class, returning a Vector of Nodes

```
Vector classInstances
=gr.askForSubjects(null,"http://www.w3.org/1999/02/22-rdf-syntax-ns#type","http$
```

Ask for everything which has a subclass, returning a vector of Statements

```
Vector superclasses
=gr.ask(null,"http://www.w3.org/2000/01/rdf-schema#subClassOf",null);
```

Ask a Squish query, returning a ResultSet (see ☞Squish BNF).

```
java.sql.ResultSet r=gr.askSquish(query);
```

Tell a graph a triple.

```
gr.tell(new Statement(new Node(subject),new Node(predicate),new
Node(object)));
```

Tell a Graph a vector of Statements

```
gr.tellAll(Vector v);
```

Delete a Statement (s cannot have null Nodes in it as yet)

```
gr.delete(Statement s);
```

Remove all statements

```
gr.removeAll();
```

The results of operations

All API-level reporting operations return a Vector of Statements or Nodes. The former can be quickly converted to a Graph so follows the triples-API format, where queries are directed at a Graph. The latter is useful where a Node-based API format is prefered.

Query-level access produces ResultSet 'rectangular' node bindings, using an implementation of java.sql.ResultSet.

Provenance (extension)

Provenance is tracked through Graphs and in the SQL version. Each graph has a base URI.

Union graphs allow querying several Graphs together.

The Triple

This is termed 'Statement' in this implementation. It consists of access methods for three Nodes. None can be missing (null): for triple-based querying, Query should be used - Query is a subclass of Statement allowing nulls and providing access methods for variable names.

Nodes

use Node.getResource(), Node.getLiteral() and Node.getBlank() to create Nodes; also can have getSubjects(), getObjects(), getPredicates(), with or without an identifier for the Node as a node-centric API.

URIrefs

All Nodes (URIref, blank, Literal) are Nodes, with their features specified by isblank (boolean, false by default) and isresource (boolean, true by default). The source Graph of each Node is avilable using (getGraph()). There is no access to the namespace used. getContent() returns the uri of the Node.

Literals

Literals are Nodes with isresource false. No language tagging is present at the moment. getContent() returns the String associated with the Literal.

Typed literals

No typing is supported at this time.

XML literals

No support is available for XML literals.

Blank Nodes

Blank nodes are Nodes with isblank=true and isresource=false. They have internal identifiers generated from the id from the parser hashed with the name of the file. They are accessible using getID(). getContent will return Util.sha1(graph.getBase()+id. Blank node content is intended to be globally unique and starts with '_:'. Blank node IDS are unique within the document only.

Graph-level operations

Merging is supported using UnionGraph in the in-memory implementation, and SQLGraph in the SQL implementation. Isomorphism is not supported.

RDF Containers

There is no explicit handling of RDF containers.

RDF Collections

There is no explicit handling of RDF collections.

RDF Reification

There is no explicit handling of RDF reification.

RDFS support - RDFS Classes

No support

RDFS Properties

No support

Instance validation

Although there is no support for RDFS classes and properties, an application, ☞Rosco[16], has been written to demonstrate instance validation. The code is available in the distribution under app/.

I/O - The code does not have its own parser, but can use ☞ARP (part of Jena) There is no code for pretty printing, although it will read and write basic Ntriples.

External considerations with RDF API semantics - Concurrent and Overlapping Operations

There is no transaction support or threading.

Selection of Graph Implementation

The software can be used as an in-memory implementation or with MySQL or the Postgresql SQL database.
    Returning provenance in the SQL versions is a significant isssue, discussed in detail in the companion report ☞SWAD-Europe Deliverable 7.3: Databases, Query, API, Interfaces Public release of a "strawman" query language implementation incorporating current best practice.

Query Support

The software uses the ☞Squish RDF query language to query both kinds of store. See ☞SWAD-Europe Deliverable 7.3: Databases, Query, API, Interfaces Public release of a "strawman" query language implementation incorporating current best practice.

Triple Matching

The software uses ask() to query the in-memory and SQL stores.
    Ordering of triples depends on the backend - the Postgresql and MySQL stores return alphabetically sorted data by default.

RDF Path - There has been significant recent interest in RDFPath-like languages, but there is no support as yet for these in this software.

RDF Query - More information is available in a companion document, ☞SWAD-Europe Deliverable 7.3: Databases, Query, API, Interfaces Public release of a "strawman" query language implementation incorporating current best practice. See also ☞testcases for RDF query.

Inference - Inference and closure are not implemented. Certain owl properties are implemented in app/.

References

[1] SWAD-Europe Deliverable 7.4: Public release of a "strawman" query language implementation incorporating current best practice. http://www.w3.org/2001/sw/Europe/reports/rdf_ql_impl/
Libby Miller, November 2003.

[2] SWAD-Europe Workpackage 7: RDF API and query implementation snapshot
http://www.w3.org/2001/sw/Europe/200311/
Libby Miller, November 2003.

[3] W3C SOFTWARE NOTICE AND LICENSE
http://www.w3.org/Consortium/Legal/2002/copyright-software-20021231
Joseph Reagle, January 2001.

[4] RDF Query (and Rule) Testcase Repository
http://www.w3.org/2003/03/rdfqr-tests/
Dan Brickley et al, March 2003.

[5] SWAD-Europe Deliverable 7.1: RDF API requirements and comparison
http://www.w3.org/2001/sw/Europe/reports/rdf_api_comparison_report
Jan Grant, July 2003.

[6] SWAD-Europe Deliverable 7.2: Report on RDF Query Languages
http://www.w3.org/2001/sw/Europe/reports/rdf_ql_comparison_report/
Libby Miller, October 2003.

[7] Strawman RDF API and Query Implementation Javadoc
http://sw1.ilrt.org/rdfquery/javadoc/
Libby Miller, November 2003.

[8] RDF API Draft
http://www-db.stanford.edu/~melnik/rdf/api.html
Sergey Melnik, January 2001.

[9] Sirpac RDF API
http://dev.w3.org/cvsweb/java/classes/org/w3c/rdf/
Janne Saarela, Art Barstow

[10] RubyRDF
http://www.w3.org/2001/12/rubyrdf/intro.html
Dan Brickley, 2001-2003

[11] Squish RDF Query Language BNF
http://sw1.ilrt.org/rdfquery/squish-bnf.html
Libby Miller, November 2003.

[12] rdfDB query language
http://guha.com/rdfdb/query.html
R. V. Guha, 2000.

[13] Jena Semantic Web Toolkit
http://www.hpl.hp.com/semweb/jena.htm
Semantic Web Group, 2001-2003.

[14] RDF query demonstrators
http://sw1.ilrt.org/discovery/
Libby Miller, 2000-2003

[15] Squish internationalization demonstrator
http://sw1.ilrt.org/discovery/2003/10/I18N/

[16] Rosco - a non-judgemental RDF schema and document checker
http://sw1.ilrt.org.org/discovery/2003/08/validation/
Libby Miller, August 2003.