SWAD-Europe Deliverable 7.4: Databases, Query, API, Interfaces. Public release of a "strawman" query language implementation incorporating current best practice.

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 a "strawman" query language implementation incorporating current best practice, which may use or build upon existing implementations. Open Source/free software.
URI:
☞http://www.w3.org/2001/sw/Europe/reports/rdf_api_ql/
Authors:
Libby Miller
Abstract:
Public release of a "strawman" query language implementation incorporating current best practice, which may use or build upon existing implementations. Open Source/free software.
Status:
Document started 2003-08-19; this revision 2003-11-05. This is a completed report.

Comments on this document are welcome and should be sent 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 document describes some features of a small RDF query implementation ('Tinkling')[1] written in Java. Its API is described in more detail in a companion document[2]. Other work funded by the SWAD-Europe project in this area includes work on RDF Query testcases[6] undertaken in collaboration with members of the Semantic Web Interest Group. A snapshot of Tinkling may be downloaded[7].

Those interested in RDF and Querying should also look at the work of the W3C Data Access Working Group[8], which is standardising work in this area; and those interested in RDf query in Java might consider using the well-supported Jena code from HP Labs[9].

History - The Squish query language is an extended version of R.V.Guha's rdfDB query language[3]. Guha's is a simple, conjunctive SQL-like query language, where you SELECT variables FROM a database WHERE (predicate subject object), where predicate subject and object can be uris or variables, or in the case of object, literals (in inverted commas). Guha also has a syntax for insert information into the database.

Squish differs in three ways:

- 'FROM' is only used when referring to comma-separated RDF files, and is not used to refer to a database.
- where used, uris can be shortened, e.g. (foaf:name ?s ?o) instead of (http;//xmlns.com/foaf/0.1/name ?s ?o). If this form is used, add a clause at the end 'USING foaf for http://xmlns.com/foaf/0.1/'.
- contraint clauses can be used to filter the results. The most useful of these is case-insensitive substring ('~'), for example: ?name ~ 'libby'.

More information is in the ☞BNF. Here is a sample query showing all these features:

```
select ?name
from http://swordfish.rdfweb.org/people/libby/rdfweb/webwho.xrdf
where
(foaf:name ?person ?name)
and ?name ~ 'libby'
using foaf for http://xmlns.com/foaf/0.1/
```

☞try this query

Testing - Tests and applications are available in the distribution. Tinkling is being used to power two demonstration websites[4], [5] backed with MySQL and containing about 350,000 triples.

Provenance - The SQL implementation does not return provenance information via query or API, although it is stored. In the in-memory version provenance can be returned like this:

```
 Graph gr = new Graph(uri, Util.RDFXML);
 gr.load();
java.sql.ResultSet r=gr.askSquish(query);

        while(r.next()){
        Node n = (Node)r.getNode("name");
        String source=n.getGraph().getBase();
        }
```

2. Sample usage

Querying an in-memory Graph with Squish - Create a Graph and populate it:

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

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

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

Querying an in-memory UnionGraph with Squish - Create a UnionGraph and populate it:

```
 Graph gr = new Graph(uri, Util.RDFXML);
 gr.load();
 UnionGraph ug=new UnionGraph();
 ug.add(g);
```

Ask a Squish query.

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

Querying an SQLGraph with Squish - Create an SQLGraph and populate it:

```
 Graph gr = new Graph(uri, Util.RDFXML);
 gr.load();
 SQLGraph sq = new SQLGraph();
 sq.setdb(db);
 sq.setDriver("com.mysql.jdbc.Driver");
 sq.removeAll();
 sq.add(gr);
```

Ask a Squish query.

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

3. Squish BNF

This was originally generated using ☞javacc and jjdoc. I've made some alterations to make it more readable. Here are some ☞example queries.

The query language is similar to and based on ☞R. V. Guha's RDFDB QL

CompilationUnit	::=	☞Query <EOF>
Query	::=	☞SelectClause (☞FromClause)? ☞TriplePatternClause (☞ConstraintClause)? (☞UsingClause)?
SelectClause	::=	'SELECT' ☞VarList
FromClause	::=	'FROM' ☞UriList
TriplePatternClause	::=	'WHERE' ☞TriplePatternList

ConstraintClause	::=	'AND' ☞ConstraintList
UsingClause	::=	'USING' (☞ForList)+
TriplePatternList	::=	☞TriplePattern (☞TriplePattern)*
TriplePattern	::=	'(' ☞VarOrLiteral ☞VarOrLiteral ☞VarOrLiteral ')'
VarOrLiteral	::=	☞Var
	|	☞Literal
Var	::=	'?' ☞Identifier
VarList	::=	☞Var ((',')? ☞Var)*
UriList	::=	☞UriLiteral ((",")? ☞UriLiteral)*
ConstraintList	::=	☞Expression ('AND' ☞Expression)*
ForList	::=	Identifier 'FOR' ☞UriLiteral
Expression	::=	☞Var ☞SomeFunction
SomeFunction	::=	(☞NumExpression | ☞StringExpression)+
NumExpression	::=	('>' | '<' | '==' | '=' | '!=' | '<=' | '>=') ☞NumericLiteral
StringExpression	::=	('like' | 'ne' | 'eq' | '~') ☞Literal
Literal	::=	☞TextLiteral
	|	☞UriLiteral
	|	☞NumericLiteral
NumericLiteral	::=	An integer
	|	A floating point number
UriLiteral	::=	A letter followed by none or more letters, numbers or other characters allowed in ☞RFC 2396
TextLiteral	::=	One or more letters or numbers enclosed in inverted commas
Identifier	::=	A letter followed by optional numbers and letters.

4. MySQL table layout used

```
mysql> describe triples;
+------------+--------------+------+-----+---------+-------+
| Field      | Type         | Null | Key | Default | Extra |
+------------+--------------+------+-----+---------+-------+
| subject    | int(11)      | YES  | MUL | NULL    |       |
| predicate  | int(11)      | YES  | MUL | NULL    |       |
| object     | int(11)      | YES  | MUL | NULL    |       |
| source     | varchar(255) | YES  | MUL | NULL    |       |
| isresource | tinyint(1)   | YES  |     | NULL    |       |
+------------+--------------+------+-----+---------+-------+


mysql> describe resources;
+---------+---------+------+-----+---------+-------+
| Field   | Type    | Null | Key | Default | Extra |
+---------+---------+------+-----+---------+-------+
| keyhash | int(11) | YES  | MUL | NULL    |       |
| value   | text    | YES  | MUL | NULL    |       |
+---------+---------+------+-----+---------+-------+
```

MySQL SQL table creation - `libby@bender:/usr/bin$ ./mysqldump test`

```
-- MySQL dump 8.21
--
-- Host: localhost    Database: test
---------------------------------------------------------
-- Server version       3.23.49-log

--
-- Table structure for table 'resources'
--

CREATE TABLE resources (
  keyhash int(11) default NULL,
  value text,
```

```
  UNIQUE KEY khval (keyhash,value(255)),
  KEY kh (keyhash),
  KEY val (value(255))
) TYPE=MyISAM;

--
-- Dumping data for table 'resources'
--



--
-- Table structure for table 'triples'
--

CREATE TABLE triples (
  subject int(11) default NULL,
  predicate int(11) default NULL,
  object int(11) default NULL,
  source varchar(255) default NULL,
  isresource tinyint(1) default NULL,
  UNIQUE KEY spos (subject,predicate,object,source),
  KEY sub (subject),
  KEY pred (predicate),
  KEY obj (object),
  KEY src (source),
  KEY sp (subject,predicate),
  KEY so (subject,object),
  KEY po (predicate,object)
) TYPE=MyISAM;

--
-- Dumping data for table 'triples'
--
```

5. References

[1] Tinkling, a small RDF API and Query Language implementation
☞http://sw1.ilrt.org/rdfquery/
Libby Miller, November 2003.
[2] SWAD-Europe Deliverable 7.3: Public release of reference implementation of an RDF API
☞http://www.w3.org/2001/sw/Europe/reports/rdf_api_impl/
Libby Miller, November 2003.
[3] rdfDB query language
☞http://guha.com/rdfdb/query.html
R. V. Guha, 2000.
[4] FOAF (Friend of a friend) database interface
☞http://swordfish.rdfweb.org/rweb/who
Libby Miller, 2002-2003.
[5] Codepiction image search demonstrator
☞http://swordfish.rdfweb.org/discovery/2001/08/codepict/
Libby Miller, 2001-2003.
[6] Summary of RDF query tests work, February-May 2003
☞http://www.w3.org/2003/03/rdfqr-tests/summary.html
Libby Miller, 2003.
[7] RDF Query implementation snapshot, November 2003.
☞http://www.w3.org/2001/sw/Europe/200311/readme.html
[8] W3C Data Acces Working Group
☞http://www.w3.org/2001/sw/DataAccess/
[9] Jena 2 - A Semantic Web Framework
☞http://www.hpl.hp.com/semweb/jena2.htm