RDF2FS – A Unix File System RDF Store

Michael Sintek and Gunnar Aastrand Grimnes

DFKI GmbH, Knowledge Management Department
Kaiserslautern, Germany
firstname.lastname@dfki.de

1 Motivation

RDF2FS is a script that translates an arbitrary RDF graph into a directory structure in a (Unix) file system, which enables Semantic Web applications without the need of having a dedicated RDF store. There are multiple reasons why such an approach makes sense:

1. To access the RDF store (incl. simple queries), one can use the (scripting and any other) language of one's choice, as long as it has proper file system support (incl. handling of symbolic links).
2. The RDF store can be browsed (and edited) with a normal file browser (see screenshot).
3. Subversion showed that people are more comfortable with storing data in a file system than in a dedicated database.
4. Our approach can also have a nice educational effect: "normal" file system users and also "hackers" will be able to understand RDF more easily, without ever seeing an RDF file.
5. If RDF is used as metadata for documents, one usually has two places where metadata is stored: in the hierarchical directory structure where the documents are placed, and in the RDF metadata graph. This means that documents have to be "annotated" twice, by filing the documents into the intended folders, and by creating the RDF metadata with a separate tool. Our approach allows this do be done at one place and with only one tool that everyone is using anyway: a file browser.

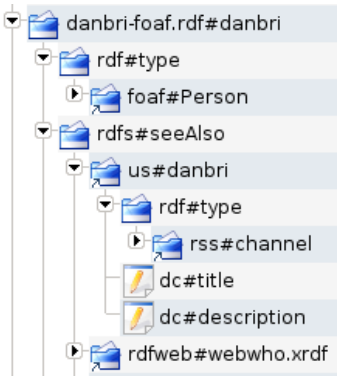
2 Mapping Sketch

RDF2FS maps an RDF graph to a target root directory r as follows:

- All resources s in the graph become sub-directories of the root directory: r/s.
- For each triple $< s, p, o >$:
  - if p is literal-valued, we create the file $r/s/p$ (if it does not yet exist) and append o to this file (i.e., each line of o is a value for the property p for subject s)

- if p is object-valued, we create a directory $r/s/p$, and the object o becomes a symbolic link $r/s/p/o \rightarrow ../../o$, i.e., it links back to that resource on the root level (properties that are both literal- and object-valued are not yet supported; this could easily be done with some naming convention, but would make queries slightly more difficult)

– The names of the files and directories are not the full resource URIs but of the format *namespace-abbrev#localname*. RDF2FS keeps a list of commonly used namespace prefixes, such as *rdfs*, *foaf*, *dc*, etc., and will generate new mappings for unknown namespaces.



3 Querying the Store

The appendix shows that all of the simple queries are easily supported using standard POSIX file system utilities (such as find, grep, ...), including all combinations of statement queries, path expressions, and conjunctive queries.

4 Future Work

In a future version, RDF2FS should support to translate back from a file system RDF store to an RDF file. Real files that correspond to resources and that should be stored in these resource directories (e.g., in some "." file), are not yet supported. Furthermore, mounting an RDF file via FUSE instead of translating to an existing file system would allow more efficient storage (esp. in case of file systems that do not handle many small files well), better handling of queries, and checking for illegal operations.

5 Demo Details

The mapping is implemented twice, as a Java program and a Python script. Either version will translate a set of RDF files into a bash-script that will create the necessary directories, files, and symbolic links. Please download http://www.dfki.uni-kl.de/~sintek/SFSW2008/RDF2FS.tgz and follow the instructions in `README`.

Appendix: Bash Queries

```
# Literal-valued properties

# triple queries with one variable:

# s p ?o  ->  cat s/p
echo '** danbri-foaf.rdf#danbri foaf#name ?o'
cat danbri-foaf.rdf#danbri/foaf#name

# ?s p o  ->  grep -l o */p | $FIRST
# (where $FIRST is an awk call that selects the first part of a path)
echo -e '\n** ?s foaf#name "Dan Brickley"'
grep -l "Dan Brickley" */foaf#name | $FIRST

# s ?p o  ->  grep -l o s/* | $SECOND
# (where $SECOND is an awk call that selects the second part of a path)
echo -e '\n** danbri-foaf.rdf#danbri ?p "Dan Brickley"'
grep -l "Dan Brickley" danbri-foaf.rdf#danbri/* | $SECOND

# triple queries with two variables:

# ?s ?p o  ->  grep -l o */*
echo -e '\n** ?s ?p "Dan Brickley"'
grep -l "Dan Brickley" */*

# ?s p ?o  ->  ls -1 */p
#   plus retrieval of literals (simply with cat)
echo -e '\n** ?s foaf#name ?o'
for f in $(ls -1 */foaf#name ); do
  echo $( echo $f | $FIRST ) \"$( cat $f )\"
done

# s ?p ?o  ->  file pattern s/* plus retrieval of literals (and resources)
echo -e '\n** danbri-foaf.rdf#danbri ?p ?o'
for f in danbri-foaf.rdf#danbri/*; do
  if [ -f $f ]; then
    echo $( echo $f | $SECOND ) \"$( cat $f )\"
  else # resources are just the files (links) in $f
    echo $( echo $f | $SECOND ) $( ls $f )
  fi
done

# Object-valued properties

# triple queries with one variable:
# s p ?o  ->  ls s/p
echo -e '\n** danbri-foaf.rdf#danbri/foaf#knows ?o'
ls danbri-foaf.rdf#danbri/foaf#knows
```

```
... (left as excercise to the reader :-) )

# Path expressions:

# s p1 _ p2 ?o  ->  cat s/p1/*/p2
echo -e '\n** path: danbri-foaf.rdf#danbri foaf#knows ?_ foaf#name ?o'
cat danbri-foaf.rdf#danbri/foaf#knows/*/foaf#name

# ?s p1 _ p2 o  ->  grep -l o */p1/*/p2
echo -e '\n** path: ?s foaf#knows ?_ foaf#name "Libby Miller"'
grep -l "Libby Miller" */foaf#knows/*/foaf#name | $FIRST

# Conjunction:

# ?s p1 o1  AND  ?s p2 o2
#    ->
# using intersection (realized with sort and uniq)
# grep -l o1 */p1 | $FIRST | sort -u > tmp1
# grep -l o2 */p2 | $FIRST | sort -u > tmp2
# sort -m tmp1 tmp2 | uniq -d # = intersection

echo -e '\n** ?s foaf#plan "Save the world" AND ?s uranai#bloodtype "A+"'

grep -l "Save the world" */foaf#plan | $FIRST | sort -u > tmp1
grep -l "A+" */uranai#bloodtype | $FIRST | sort -u > tmp2
sort -m tmp1 tmp2 | uniq -d
rm tmp1 tmp2
```

Output of this script:

```
** danbri-foaf.rdf#danbri foaf#name ?o
Dan Brickley

** ?s foaf#name "Dan Brickley"
anon-64848a97%3A1187e661172%3A-7ffb
danbri-foaf.rdf#danbri

** danbri-foaf.rdf#danbri ?p "Dan Brickley"
foaf#name

** ?s ?p "Dan Brickley"
anon-64848a97%3A1187e661172%3A-7ffb/foaf#name
danbri-foaf.rdf#danbri/foaf#name

** ?s foaf#name ?o
anon-64848a97%3A1187e661172%3A-7fde "Pastor N Pizzor"
...
anon-64848a97%3A1187e661172%3A-7ffb "Dan Brickley"
card#i "Tim Berners-Lee"
```

```
danbri-foaf.rdf#danbri "Dan Brickley"

** danbri-foaf.rdf#danbri ?p ?o
contact#nearestAirport anon-64848a97%3A1187e661172%3A-7ffc
foaf#aimChatID "danbri_2002"
foaf#archnemesis anon-64848a97%3A1187e661172%3A-7ffb
foaf#dateOfBirth "1972-01-09"
foaf#holdsAccount anon-64848a97%3A1187e661172%3A-7ffd anon-64848a97%3A1187e661172%3A-7ffe ...
foaf#homepage danbri.org#
foaf#img ns01#Image1.gif
foaf#jabberID "danbri@jabber.org"
foaf#knows anon-64848a97%3A1187e661172%3A-7fe6 ...
foaf#mbox mailto#danbri%40apocalypse.org mailto#danbri%40danbri.org ...
...
wot#keyid "B573B63A"

** danbri-foaf.rdf#danbri/foaf#knows ?o
anon-64848a97%3A1187e661172%3A-7fe6
...
card#i

** path: danbri-foaf.rdf#danbri foaf#knows ?_ foaf#name ?o
Damian Steer
...
Tim Berners-Lee

** path: ?s foaf#knows ?_ foaf#name "Libby Miller"
danbri-foaf.rdf#danbri

** ?s foaf#plan "Save the world" AND ?s vocab#uranaibloodtype "A+"
danbri-foaf.rdf#danbri
```