Get RDF data cube example file from RDF data cube specifications

mja@statgroup.dk

2016-01-16

Contents

Get RDF data cube example file from RDF data cube specifications

This script downloads the example from the RDF data cube vocabulary and stores it in the package. The example is normalized.

Setup

```
devtools::load_all(pkg="../..")
```

```
## Loading rrdfqb
```

R-code

```
library(RCurl)
library(devtools)
qbURL<-"https://raw.githubusercontent.com/UKGovLD/publishing-statistical-data/master/specs/src/main/exa
if (! url.exists(qbURL) ) {
  stop(paste0("Can not access URL ",qbURL))
}
examplettl <- getURL(qbURL)
savefile <- file.path(system.file("extdata/sample-rdf", package="rrdfqb"), "example.ttl" )
writeLines( examplettl, savefile)
cat("written to ", normalizePath(savefile) )
```

```
## written to  /home/ma/projects/R-projects/rrdfqbcrnd0/rrdfqb/inst/extdata/sample-rdf/example.ttl
```

Show one observations from the store

```
exfile <- file.path(system.file("extdata/sample-rdf", "example.ttl", package="rrdfqb") )
store <- new.rdf(ontology=FALSE)
load.rdf( exfile, format="TURTLE", store)
```

```
## [1] "Java-Object{<ModelCom   {eg:organization @rdfs:label \"Example org\"@en; eg:organization @rdf:ty
```

```
SPARQLscript<- '
select * where {
 <http://example.org/ns#o62> ?p ?o .
}
'

results <- sparql.rdf(store, SPARQLscript )
knitr::kable(results)
```

p	o
http://example.org/ns#lifeExpectancy	83.4
http://example.org/ns#refArea	http://example.org/geo#cardiff_00pt
http://purl.org/linked-data/cube#dataSet	http://example.org/ns#dataset-le3
http://www.w3.org/1999/02/22-rdf-syntax-ns#type	http://purl.org/linked-data/cube#Observation

```
SPARQLscript<- '
prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
prefix skos: <http://www.w3.org/2004/02/skos/core#>
prefix prov: <http://www.w3.org/ns/prov#>
prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>
prefix dcat: <http://www.w3.org/ns/dcat#>
prefix owl: <http://www.w3.org/2002/07/owl#>
prefix xsd: <http://www.w3.org/2001/XMLSchema#>
prefix qb: <http://purl.org/linked-data/cube#>
prefix pav: <http://purl.org/pav>
prefix dct: <http://purl.org/dc/terms/>
```

```
 select * where {
?s a qb:Observation  ;
?p ?o .
values (?s) {
( <http://example.org/ns#o62> )
}
 }
'
results <- sparql.rdf(store, SPARQLscript )
```

Effect of using the ontology inference

See (https://jena.apache.org/documentation/ontology/#ontology-inference-overview).

```
exfile <- file.path(system.file("extdata/sample-rdf", "example.ttl", package="rrdfqb") )
store.ontology <- load.rdf( exfile, format="TURTLE")

results.ontology <- sparql.rdf(store.ontology, SPARQLscript )
knitr::kable(results.ontology)
```

s	p	o
http://example.org/ns#o62	http://example.org/ns#lifeExpectancy	83.4
http://example.org/ns#o62	http://example.org/ns#refArea	http://example.org/geo#cardi
http://example.org/ns#o62	qb:dataSet	http://example.org/ns#datase
http://example.org/ns#o62	rdf:type	qb:Observation
http://example.org/ns#o62	http://purl.org/linked-data/sdmx/2009/dimension#refArea	http://example.org/geo#cardi
http://example.org/ns#o62	http://purl.org/linked-data/sdmx/2009/measure#obsValue	83.4

The result shows inferred triples, added with respect to the query above.

Load, normalize phase 1, normalize phase 2, and finally dump the graph

RDF data cube normalization algorihtms can be applied (http://www.w3.org/TR/vocab-data-cube/#normalize-algorithm). Note: the `rrdfancillary` package must be installed in R to get this to work.

```
library(rrdfancillary)

exfile <- file.path(system.file("extdata/sample-rdf", "example.ttl", package="rrdfqb") )
store <- new.rdf(ontology=FALSE)
load.rdf( exfile, format="TURTLE", store)
```

```
## [1] "Java-Object{<ModelCom   {eg:organization @rdfs:label \"Example org\"@en; eg:organization @rdf:ty
```

```
normalize.phase.1.file<- file.path(system.file("extdata/cube-vocabulary-rdf", "normalize-algorithm-phase
UpdateNormPhase1 <- paste(readLines(normalize.phase.1.file), collapse="\n")
cat(UpdateNormPhase1,"\n")
```

```
## # Phase 1: Type and property closure
## # http://www.w3.org/TR/vocab-data-cube/#normalize-algorithm
##
## PREFIX rdf:            <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
## PREFIX qb:             <http://purl.org/linked-data/cube#>
##
## INSERT {
##     ?o rdf:type qb:Observation .
## } WHERE {
##     [] qb:observation ?o .
## };
##
## INSERT {
##     ?o  rdf:type qb:Observation .
##     ?ds rdf:type qb:DataSet .
## } WHERE {
##     ?o qb:dataSet ?ds .
## };
##
## INSERT {
##     ?s rdf:type qb:Slice .
## } WHERE {
##     [] qb:slice ?s.
## };
##
## INSERT {
##     ?cs qb:componentProperty ?p .
##     ?p  rdf:type qb:DimensionProperty .
## } WHERE {
##     ?cs qb:dimension ?p .
## };
##
## INSERT {
##     ?cs qb:componentProperty ?p .
##     ?p  rdf:type qb:MeasureProperty .
## } WHERE {
##     ?cs qb:measure ?p .
## };
##
## INSERT {
##     ?cs qb:componentProperty ?p .
##     ?p  rdf:type qb:AttributeProperty .
## } WHERE {
##     ?cs qb:attribute ?p .
## }
```

```
update.rdf( store, UpdateNormPhase1 )
```

```
## [1] TRUE
```

```
normalize.phase.2.file<- file.path(system.file("extdata/cube-vocabulary-rdf", "normalize-algorithm-phas
UpdateNormPhase2 <- paste(readLines(normalize.phase.2.file), collapse="\n")
cat(UpdateNormPhase2,"\n")
```

```
## # Phase 2: Push down attachment levels
## # http://www.w3.org/TR/vocab-data-cube/#normalize-algorithm
##
## PREFIX qb:              <http://purl.org/linked-data/cube#>
##
## # Dataset attachments
## INSERT {
##     ?obs  ?comp ?value
## } WHERE {
##     ?spec    qb:componentProperty ?comp ;
##              qb:componentAttachment qb:DataSet .
##     ?dataset qb:structure [qb:component ?spec];
##              ?comp ?value .
##     ?obs     qb:dataSet ?dataset.
## };
##
## # Slice attachments
## INSERT {
##     ?obs  ?comp ?value
## } WHERE {
##     ?spec    qb:componentProperty ?comp;
##              qb:componentAttachment qb:Slice .
##     ?dataset qb:structure [qb:component ?spec];
##              qb:slice ?slice .
##     ?slice ?comp ?value;
##            qb:observation ?obs .
## };
##
## # Dimension values on slices
## INSERT {
##     ?obs  ?comp ?value
## } WHERE {
##     ?spec    qb:componentProperty ?comp .
##     ?comp a  qb:DimensionProperty .
##     ?dataset qb:structure [qb:component ?spec];
##              qb:slice ?slice .
##     ?slice ?comp ?value;
##            qb:observation ?obs .
## }
```

```
update.rdf( store, UpdateNormPhase2 )
```

```
## [1] TRUE
```

```
normalizedfile<- file.path(system.file("extdata/sample-rdf", package="rrdfqb"), "example-normalized.ttl
save.rdf( store, normalizedfile, format="TURTLE")
```

```
## [1] "/home/ma/projects/R-projects/rrdfqbcrnd0/rrdfqb/inst/extdata/sample-rdf/example-normalized.ttl"
```

Update example cube using Fuseki

Using Fusiki to do the update.

```
FUSEKI_HOME=/opt/apache-jena-fuseki-2.3.1/
(${FUSEKI_HOME}fuseki-server --mem --update /ex2) &
```

The `--men` creates in memory-store, `--update` enables updating operation and `/ex2` is the name of the dataset.

ToDo(MJA): add storing PID in file (`echo $$ > fuseki.pid;`), and redirecting output from fuseki.

Fuseki will re-use configuration files - so be sure of the contents of the run directory.

To load, normalize phase 1, normalize phase 2, and finally dump the graph ${FUSEKI_HOME}bin/s-put http://localhost:3030/ex2/data default ../sample-rdf/example.ttl ${FUSEKI_HOME}bin/s-update –server=http://localhost:3030/ex2/update –update=../cube-vocabulary-rdf/normalize-algorithm-phase-1.ru ${FUSEKI_HOME}bin/s-update –server=http://localhost:3030/ex2/update –update=../cube-vocabulary-rdf/normalize-algorithm-phase-2.ru ${FUSEKI_HOME}bin/s-get http://localhost:3030/ex2/get default > ../sample-rdf/example-normalize-with-fuseki.ttl

Using jena 2.13

This was successfull in december 2015. The approach relies on a not recent version of jena, so it is not investigated further.

```
/opt/apache-jena-2.13.0/arq --desc=jena-assambler.ttl  "select * where {?s ?p ?o} limit 10"
/opt/apache-jena-2.13.0/bin/update --desc=jena-assambler.ttl --update=normalize-algorithm-phase-1.ru --
/opt/apache-jena-2.13.0/update --desc=jena-assambler.ttl --update=normalize-algorithm-phase-2.ru --verb
```

Using jena 3.0.0

This does not work with jena 3.0.0 in december 2015.

```
/opt/apache-jena-3.0.0/bin/tdbloader --loc=DB example.ttl
arq --desc=tdb-assembler.ttl  "select * where {?s ?p ?o} limit 10"
```

Fuseki with customized configuration file

Start fuseki to create the configuration files.

```
(FUSEKI_HOME=/opt/apache-jena-fuseki-2.3.0 /opt/apache-jena-fuseki-2.3.0/fuseki-server )
```

In directory run/configuration add configuration for ex endpoint using the filename run/configuration/ex.ttl as:

```
@prefix :        <#> .
@prefix fuseki:  <http://jena.apache.org/fuseki#> .
@prefix rdf:     <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs:    <http://www.w3.org/2000/01/rdf-schema#> .
@prefix tdb:     <http://jena.hpl.hp.com/2008/tdb#> .
@prefix ja:      <http://jena.hpl.hp.com/2005/11/Assembler#> .
## ---------------------------------------------------------------
## Updatable TDB dataset with all services enabled.
<#service_tdb_all> rdf:type fuseki:Service ;
```

```
rdfs:label                      "TDB ex" ;
fuseki:name                     "ex" ;
fuseki:serviceQuery             "query" ;
fuseki:serviceQuery             "sparql" ;
fuseki:serviceUpdate            "update" ;
fuseki:serviceUpload            "upload" ;
fuseki:serviceReadWriteGraphStore      "data" ;
# A separate read-only graph store endpoint:
fuseki:serviceReadGraphStore       "get" ;
fuseki:dataset           <#tdb_dataset_readwrite> ;
.
<#tdb_dataset_readwrite> rdf:type      tdb:DatasetTDB ;
tdb:location "run/databases/ex" ;
##ja:context [ ja:cxtName "arq:queryTimeout" ;  ja:cxtValue "3000" ] ;
##tdb:unionDefaultGraph true ;
.
```

Note - all files in run/configuration/ are read - so do not leave backup files in the directory.

Start again: (FUSEKI_HOME=/opt/apache-jena-fuseki-2.3.0 /opt/apache-jena-fuseki-2.3.0/fuseki-server)

To run update query

```
(FUSEKI_HOME=/opt/apache-jena-fuseki-2.3.0 /opt/apache-jena-fuseki-2.3.0/bin/s-update --server=http://l
(FUSEKI_HOME=/opt/apache-jena-fuseki-2.3.0 /opt/apache-jena-fuseki-2.3.0/bin/s-update --server=http://l
```

To dump the graph (FUSEKI_HOME=/opt/apache-jena-fuseki-2.3.0 /opt/apache-jena-fuseki-2.3.0/bin/s-get http://localhost:3030/ex/get default)