Create simple RDF data qube

PhuseSubTeamAnalysisResults@example.org

2015-01-11

Contents

Introduction

This vignette shows how to create a very simple RDF data cube from - data - metadata

Setup

```
library (rrdf)
```

```
## Loading required package: rJava
## Loading required package: methods
## Loading required package: rrdflibs
```

```
library(rrdfqbcrnd0)
```

```
## Loading required package: xlsx
## Loading required package: xlsxjars
## Loading required package: RCurl
## Loading required package: bitops
##
## Attaching package: 'RCurl'
##
## The following object is masked from 'package:rJava':
##
##     clone

## Warning: replacing previous import by 'rJava::clone' when loading
## 'rrdfqbcrnd0'
```

Define data

```
obsData<- data.frame(
  category=c("AA-group", "BB-group"),
  procedure=c("count", "count" ),
  factor=c("quantity", "quantity" ),
  unit=c("subject", "subject" ),
  denominator=c(" ", " "),
  measure=c( 123, 456 ),
  stringsAsFactors=FALSE  )
knitr::kable(obsData)
```

category	procedure	factor	unit	denominator	measure
AA-group	count	quantity	subject		123
BB-group	count	quantity	subject		456

Define meta data

```
cubeMetadata<- data.frame(
  compType=c("dimension", "dimension", "dimension", "unit", "denominator", "measure", "metadata"),
  compName=c("category", "procedure", "factor", "attribute", "attribute", "measure", "domainName"),
  codeType=c("DATA", "DATA", "DATA", " ", " ", "<NA>","<NA>"),
  nciDomainValue=c(" "," "," "," "," "," "," "),
  compLabel=c("Category", "Statistical procedure", "Type of procedure", "Result", "Unit", "Denominator"
  Comment=c(" "," "," "," "," "," "," "),
  stringsAsFactors=FALSE  )
knitr::kable(cubeMetadata)
```

compType	compName	codeType	nciDomainValue	compLabel	Comment
dimension	category	DATA		Category	
dimension	procedure	DATA		Statistical procedure	
dimension	factor	DATA		Type of procedure	
unit	attribute			Result	
denominator	attribute			Unit	
measure	measure			Denominator	
metadata	domainName			EXAMPLE	

Create RDF data cube

The RDF data cube for the data above is created using

```
outcube<- BuildCubeFromDataFrames(cubeMetadata, obsData )
```

The RDF data cube is serialized in turtle format and stored as

```
outcube
```

```
## [1] "/tmp/RtmpWS6gNp/DC-EXAMPLE-R-V-0-0-0.TTL"
```

Looking at the RDF data cube

Now look at the generated cubes by loading the turle files. Note: by specifying prefix the output contains is shown using the prefixes. Note for future: This may be a disadvantage if the value of the prefix, say ds, changes.

```
dataCubeFile<- outcube
```

```
# the rest of the code only depends on the value of dataCubeFile
checkCube <- new.rdf(ontology=FALSE)  # Initialize
load.rdf(dataCubeFile, format="TURTLE", appendTo= checkCube)
summarize.rdf(checkCube)
```

```
## [1] "Number of triples: 127"
```

```
# determine the domain name; used for defining prefixes
# TODO: reconsider the use of domain specific prefixes

# TODO: make this simpler - the only purpose is find the dsdName
# TODO: a qb:DataStructureDefinition, and for domainname, say, the DM in ds:dsd-DM
tempstr<- as.character(sparql.rdf(checkCube, "select ?s where { ?s a <http://purl.org/linked-data/cube#
tempstrvec<- unlist(strsplit( tempstr, "/"))
dsdName<- tempstrvec[length(tempstrvec)]
domainName<- strsplit(dsdName,"-")[[1]][[2]]

common.prefixes <- data.frame(
    prefix=gsub("^prefix","",names(Get.default.crnd.prefixes())),
    namespace=as.character(Get.default.crnd.prefixes() )
    )
custom.prefixes <-Get.qb.crnd.prefixes(tolower(domainName))

forsparqlprefix<- Get.rq.prefix.df(rbind(common.prefixes, custom.prefixes))
```

The next statement shows the first 10 triples in the cube.

s	p	o
dccs:measure	qb:measure	prop:measure
dccs:measure	rdfs:label	Denominator
dccs:measure	rdf:type	qb:ComponentSpecification
code:Category	rdfs:subClassOf	skos:Concept

s	p	o
code:Category	rdfs:seeAlso	code:category
code:Category	rdfs:label	Class for code list: category
code:Category	rdfs:comment	Specifies the category for each observation
code:Category	rdf:type	owl:Class
code:Category	rdf:type	rdfs:Class
code:category-AA-group	skos:topConceptOf	code:category

The next statement shows the first 30 triples in the cube, where the subject is a qb:Observation.

```
cube.observations2.rq<-  paste( forsparqlprefix,
'
select *
where { ?s a qb:Observation ; ?p ?o .}
limit 30
',
"\n"
)

cube.observations2<- sparql.rdf(checkCube, cube.observations2.rq)
knitr::kable(head(cube.observations2, 10))
```

s	p	o
ds:obs1	prop:unit	subject
ds:obs1	prop:procedure	code:procedure-count
ds:obs1	prop:measure	123
ds:obs1	prop:factor	code:factor-quantity
ds:obs1	prop:denominator	
ds:obs1	prop:category	code:category-AA-group
ds:obs1	qb:dataSet	ds:dataset-EXAMPLE
ds:obs1	rdfs:label	1
ds:obs1	rdf:type	qb:Observation
ds:obs2	prop:unit	subject

The codelists are shown in the next output.

```
##          vn               clc prefLabel
## 1  category category-AA-group  AA-group
## 2  category category-BB-group  BB-group
## 3    factor   factor-quantity  quantity
## 4 procedure   procedure-count     count
```

vn	clc	prefLabel
category	category-AA-group	AA-group
category	category-BB-group	BB-group
factor	factor-quantity	quantity
procedure	procedure-count	count

The dimensions are shown in the next output.

```
cube.dimensions.rq<- paste(forsparqlprefix,
'
select * where
{ [] qb:dimension ?p .  }
',
"\n"
)
cube.dimensions<- as.data.frame(sparql.rdf(checkCube, cube.dimensions.rq), stringsAsFactors=FALSE)
knitr::kable(print(cube.dimensions))
```

```
##                 p
## 1    prop:factor
## 2 prop:procedure
## 3  prop:category
```

p

prop:factor
prop:procedure prop:category

And finally the SPARQL query for observations.

This is the query for getting the observations

```
cat(cube.observations.rq)
```

```
## 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/>
## prefix mms: <http://rdf.cdisc.org/mms#>
## prefix cts: <http://rdf.cdisc.org/ct/schema#>
## prefix rrdfqbcrnd0: <http://www.example.org/rrdfqbcrnd0/>
## prefix code: <http://www.example.org/dc/code/>
## prefix prop: <http://www.example.org/dc/example/prop/>
```

```
## prefix dccs: <http://www.example.org/dc/example/dccs/>
## prefix ds: <http://www.example.org/dc/example/ds/>
##  select * where { ?s a qb:Observation  ;
##         qb:dataSet ds:dataset-EXAMPLE  ;
##  prop:factor ?factor;
## prop:procedure ?procedure;
## prop:category ?category;
##  prop:measure      ?measure ;
##  optional{ ?factor skos:prefLabel ?factorvalue . }
## optional{ ?procedure skos:prefLabel ?procedurevalue . }
## optional{ ?category skos:prefLabel ?categoryvalue . }
##  }
```

And finally the observations.

```
cube.observations<- as.data.frame(sparql.rdf(checkCube, cube.observations.rq ), stringsAsFactors=FALSE)
knitr::kable(cube.observations[,c(paste0(sub("prop:", "", cube.dimensionsattr), "value"),"measure")])
```

factorvalue	procedurevalue	categoryvalue	measure
quantity	count	AA-group	123
quantity	count	BB-group	456