Create Integrity Contraints SPARQL Queries from RDF data cube definition

mja@statgroup.dk

2016-01-23

Contents

Create Integrity Contraints SPARQL Queries from RDF data cube definition

This script retrieves the RDF data cube vocabulary from (http://www.w3.org/TR/2014/REC-vocab-data-cube-20140116/).

The Integrity Constraints are stored as file in the package

Setup

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

```
## Loading rrdfqb
```

R-code

IC-19 is two queries, so it is split into IC-19a and IC-19b: For IC-20 and IC-21 special handling are needed. The queries are templates and the value of p should be inserted as $p in the template.

```
library(RCurl)
library(XML)
library(devtools)

qbURL<-"http://www.w3.org/TR/2014/REC-vocab-data-cube-20140116/"
if (! url.exists(qbURL) ) {
  stop(paste0("Can not access URL ",qbURL))
}


## Acknowledgement: I got the approach from
## http://stackoverflow.com/questions/1395528/scraping-html-tables-into-r-data-frames-using-the-xml-pack
```

```
webpage <- getURL(qbURL)
## The following two lines is suggested in the stackoverflow post
## Apparantly not needed here
## Process escape characters
## webpage <- readLines(tc <- textConnection(webpage)); close(tc)

## Parse the html tree, ignoring errors on the page
pagetree <- htmlTreeParse(webpage, error=function(...){}, useInternalNodes = TRUE)

## appears that integrity checks starts with h3 and then a table with class bordered-table
## so that's what we look for
both<- getNodeSet(pagetree,"//*/h3[@id]|//*/table[@class='bordered-table']/tbody/tr/td/pre")


irq20<- "
SELECT ?p WHERE {
    ?hierarchy a qb:HierarchicalCodeList ;
                qb:parentChildProperty ?p .
    FILTER ( isIRI(?p) )
}
"

irq21<-"
SELECT ?p WHERE {
    ?hierarchy a qb:HierarchicalCodeList;
                qb:parentChildProperty ?pcp .
    FILTER( isBlank(?pcp) )
    ?pcp  owl:inverseOf ?p .
    FILTER( isIRI(?p) )
}
"

storeIC<-function(ictitle,instantiationRq,rq) {
   return( list(
     ictitel= ictitle,
     HasInstantiation= nchar(instantiationRq)>0,
     instantiationRq= instantiationRq,
     rq= rq) )
  }

qbIClist<- list()
for (i in 1:(length(both)-1)) {
  icname<- xmlGetAttr(both[[i]],"id",default="none")
  if (grepl('ic-[1-9]([0-9])*', icname ) ) {
   ictitle<- unlist(xmlValue(xmlChildren(both[[i]])$text ))
   rq<- xmlValue(xmlChildren(both[[i+1]])$text)
      ##   print(paste0( "Node ", i, ", IC name ", icname, " - ", ictitle ))
   if (icname %in% "ic-19") {
       ## XXX change list to vection - use unlist ??
       ##     print(i)
      rq<- paste0(unlist(strsplit(xmlValue(xmlChildren(both[[i+1]])$text),"\n"))[1:8], collapse="\n")
      qbIClist[["ic-19a"]]<- storeIC(gsub("IC-19", "IC-19a", ictitle), "", rq)
      rq<- paste0(unlist(strsplit(xmlValue(xmlChildren(both[[i+1]])$text),"\n"))[10:17], collapse="\n")
```

```
      qbIClist[["ic-19b"]]<- storeIC(gsub("IC-19", "IC-19b", ictitle), "", rq)
    } else if ( icname == "ic-20" ) {
      qbIClist[[icname]]<- storeIC( ictitle, irq20, rq)
    } else if ( icname == "ic-21" ) {
      qbIClist[[icname]]<- storeIC( ictitle, irq21, rq)
    } else {
      qbIClist[[icname]]<- storeIC( ictitle, "", rq)
   }
  }
  }
```

Here are the integrity constraints:

```
for (icname in names(qbIClist)) {
  ##    fileConn<-file(paste0(icname, ".rq"))
  icall<- qbIClist[[icname]]
  cat( paste(names(icall),icall,collapse="\n",sep="\n"),"\n")
##    close(fileConn)
  }
```

```
## ictitel
## IC-1.  Unique DataSet
## HasInstantiation
## FALSE
## instantiationRq
##
## rq
## ASK {
##   {
##     # Check observation has a data set
##     ?obs a qb:Observation .
##     FILTER NOT EXISTS { ?obs qb:dataSet ?dataset1 . }
##   } UNION {
##     # Check has just one data set
##     ?obs a qb:Observation ;
##        qb:dataSet ?dataset1, ?dataset2 .
##     FILTER (?dataset1 != ?dataset2)
##   }
## }
##
## ictitel
## IC-2. Unique DSD
## HasInstantiation
## FALSE
## instantiationRq
##
## rq
## ASK {
##   {
##     # Check dataset has a dsd
##     ?dataset a qb:DataSet .
##     FILTER NOT EXISTS { ?dataset qb:structure ?dsd . }
##   } UNION {
```

```
##     # Check has just one dsd
##     ?dataset a qb:DataSet ;
##        qb:structure ?dsd1, ?dsd2 .
##     FILTER (?dsd1 != ?dsd2)
##   }
## }
##
## ictitel
## IC-3. DSD includes measure
## HasInstantiation
## FALSE
## instantiationRq
##
## rq
## ASK {
##   ?dsd a qb:DataStructureDefinition .
##   FILTER NOT EXISTS { ?dsd qb:component [qb:componentProperty [a qb:MeasureProperty]] }
## }
##
## ictitel
## IC-4. Dimensions have range
## HasInstantiation
## FALSE
## instantiationRq
##
## rq
## ASK {
##   ?dim a qb:DimensionProperty .
##   FILTER NOT EXISTS { ?dim rdfs:range [] }
## }
##
## ictitel
## IC-5. Concept dimensions have code lists
## HasInstantiation
## FALSE
## instantiationRq
##
## rq
## ASK {
##   ?dim a qb:DimensionProperty ;
##        rdfs:range skos:Concept .
##   FILTER NOT EXISTS { ?dim qb:codeList [] }
## }
##
## ictitel
## IC-6. Only attributes may be optional
## HasInstantiation
## FALSE
## instantiationRq
##
## rq
## ASK {
##   ?dsd qb:component ?componentSpec .
##   ?componentSpec qb:componentRequired "false"^^xsd:boolean ;
```

```
##                   qb:componentProperty ?component .
##   FILTER NOT EXISTS { ?component a qb:AttributeProperty }
## }
##
## ictitel
## IC-7. Slice Keys must be declared
## HasInstantiation
## FALSE
## instantiationRq
##
## rq
## ASK {
##     ?sliceKey a qb:SliceKey .
##     FILTER NOT EXISTS { [a qb:DataStructureDefinition] qb:sliceKey ?sliceKey }
## }
##
## ictitel
## IC-8. Slice Keys consistent with DSD
## HasInstantiation
## FALSE
## instantiationRq
##
## rq
## ASK {
##   ?slicekey a qb:SliceKey;
##       qb:componentProperty ?prop .
##   ?dsd qb:sliceKey ?slicekey .
##   FILTER NOT EXISTS { ?dsd qb:component [qb:componentProperty ?prop] }
## }
##
## ictitel
## IC-9. Unique slice structure
## HasInstantiation
## FALSE
## instantiationRq
##
## rq
## ASK {
##   {
##     # Slice has a key
##     ?slice a qb:Slice .
##     FILTER NOT EXISTS { ?slice qb:sliceStructure ?key }
##   } UNION {
##     # Slice has just one key
##     ?slice a qb:Slice ;
##            qb:sliceStructure ?key1, ?key2;
##     FILTER (?key1 != ?key2)
##   }
## }
##
## ictitel
## IC-10. Slice dimensions complete
## HasInstantiation
## FALSE
```

```
## instantiationRq
##
## rq
## ASK {
##   ?slice qb:sliceStructure [qb:componentProperty ?dim] .
##   FILTER NOT EXISTS { ?slice ?dim [] }
## }
##
## ictitel
## IC-11. All dimensions required
## HasInstantiation
## FALSE
## instantiationRq
##
## rq
## ASK {
##     ?obs qb:dataSet/qb:structure/qb:component/qb:componentProperty ?dim .
##     ?dim a qb:DimensionProperty;
##     FILTER NOT EXISTS { ?obs ?dim [] }
## }
##
## ictitel
## IC-12. No duplicate observations
## HasInstantiation
## FALSE
## instantiationRq
##
## rq
## ASK {
##   FILTER( ?allEqual )
##   {
##     # For each pair of observations test if all the dimension values are the same
##     SELECT (MIN(?equal) AS ?allEqual) WHERE {
##         ?obs1 qb:dataSet ?dataset .
##         ?obs2 qb:dataSet ?dataset .
##         FILTER (?obs1 != ?obs2)
##         ?dataset qb:structure/qb:component/qb:componentProperty ?dim .
##         ?dim a qb:DimensionProperty .
##         ?obs1 ?dim ?value1 .
##         ?obs2 ?dim ?value2 .
##         BIND( ?value1 = ?value2 AS ?equal)
##     } GROUP BY ?obs1 ?obs2
##   }
## }
##
## ictitel
## IC-13. Required attributes
## HasInstantiation
## FALSE
## instantiationRq
##
## rq
## ASK {
##     ?obs qb:dataSet/qb:structure/qb:component ?component .
```

```
##     ?component qb:componentRequired "true"^^xsd:boolean ;
##                qb:componentProperty ?attr .
##     FILTER NOT EXISTS { ?obs ?attr [] }
## }
##
## ictitel
## IC-14. All measures present
## HasInstantiation
## FALSE
## instantiationRq
##
## rq
## ASK {
##     # Observation in a non-measureType cube
##     ?obs qb:dataSet/qb:structure ?dsd .
##     FILTER NOT EXISTS { ?dsd qb:component/qb:componentProperty qb:measureType }
##
##     # verify every measure is present
##     ?dsd qb:component/qb:componentProperty ?measure .
##     ?measure a qb:MeasureProperty;
##     FILTER NOT EXISTS { ?obs ?measure [] }
## }
##
## ictitel
## IC-15. Measure dimension consistent
## HasInstantiation
## FALSE
## instantiationRq
##
## rq
## ASK {
##     # Observation in a measureType-cube
##     ?obs qb:dataSet/qb:structure ?dsd ;
##          qb:measureType ?measure .
##     ?dsd qb:component/qb:componentProperty qb:measureType .
##     # Must have value for its measureType
##     FILTER NOT EXISTS { ?obs ?measure [] }
## }
##
## ictitel
## IC-16. Single measure on measure dimension observation
## HasInstantiation
## FALSE
## instantiationRq
##
## rq
## ASK {
##     # Observation with measureType
##     ?obs qb:dataSet/qb:structure ?dsd ;
##          qb:measureType ?measure ;
##          ?omeasure [] .
##     # Any measure on the observation
##     ?dsd qb:component/qb:componentProperty qb:measureType ;
##          qb:component/qb:componentProperty ?omeasure .
```

```
##     ?omeasure a qb:MeasureProperty .
##     # Must be the same as the measureType
##     FILTER (?omeasure != ?measure)
## }
##
## ictitel
## IC-17. All measures present in measures dimension cube
## HasInstantiation
## FALSE
## instantiationRq
##
## rq
## ASK {
##   {
##       # Count number of other measures found at each point
##       SELECT ?numMeasures (COUNT(?obs2) AS ?count) WHERE {
##           {
##               # Find the DSDs and check how many measures they have
##               SELECT ?dsd (COUNT(?m) AS ?numMeasures) WHERE {
##                   ?dsd qb:component/qb:componentProperty ?m.
##                   ?m a qb:MeasureProperty .
##               } GROUP BY ?dsd
##           }
##
##           # Observation in measureType cube
##           ?obs1 qb:dataSet/qb:structure ?dsd;
##                 qb:dataSet ?dataset ;
##                 qb:measureType ?m1 .
##
##           # Other observation at same dimension value
##           ?obs2 qb:dataSet ?dataset ;
##                 qb:measureType ?m2 .
##           FILTER NOT EXISTS {
##               ?dsd qb:component/qb:componentProperty ?dim .
##               FILTER (?dim != qb:measureType)
##               ?dim a qb:DimensionProperty .
##               ?obs1 ?dim ?v1 .
##               ?obs2 ?dim ?v2.
##               FILTER (?v1 != ?v2)
##           }
##
##       } GROUP BY ?obs1 ?numMeasures
##         HAVING (?count != ?numMeasures)
##   }
## }
##
## ictitel
## IC-18. Consistent data set links
## HasInstantiation
## FALSE
## instantiationRq
##
## rq
## ASK {
```

```
##     ?dataset qb:slice       ?slice .
##     ?slice   qb:observation ?obs .
##     FILTER NOT EXISTS { ?obs qb:dataSet ?dataset . }
## }
##
## ictitel
## IC-19a. Codes from code list
## HasInstantiation
## FALSE
## instantiationRq
##
## rq
## ASK {
##     ?obs qb:dataSet/qb:structure/qb:component/qb:componentProperty ?dim .
##     ?dim a qb:DimensionProperty ;
##         qb:codeList ?list .
##     ?list a skos:ConceptScheme .
##     ?obs ?dim ?v .
##     FILTER NOT EXISTS { ?v a skos:Concept ; skos:inScheme ?list }
## }
## ictitel
## IC-19b. Codes from code list
## HasInstantiation
## FALSE
## instantiationRq
##
## rq
## ASK {
##     ?obs qb:dataSet/qb:structure/qb:component/qb:componentProperty ?dim .
##     ?dim a qb:DimensionProperty ;
##         qb:codeList ?list .
##     ?list a skos:Collection .
##     ?obs ?dim ?v .
##     FILTER NOT EXISTS { ?v a skos:Concept . ?list skos:member+ ?v }
## }
## ictitel
## IC-20. Codes from hierarchy
## HasInstantiation
## TRUE
## instantiationRq
##
## SELECT ?p WHERE {
##     ?hierarchy a qb:HierarchicalCodeList ;
##                  qb:parentChildProperty ?p .
##     FILTER ( isIRI(?p) )
## }
##
## rq
## ASK {
##     ?obs qb:dataSet/qb:structure/qb:component/qb:componentProperty ?dim .
##     ?dim a qb:DimensionProperty ;
##         qb:codeList ?list .
##     ?list a qb:HierarchicalCodeList .
##     ?obs ?dim ?v .
```

```
##     FILTER NOT EXISTS { ?list qb:hierarchyRoot/<$p>* ?v }
## }
##
## ictitel
## IC-21. Codes from hierarchy (inverse)
## HasInstantiation
## TRUE
## instantiationRq
##
## SELECT ?p WHERE {
##     ?hierarchy a qb:HierarchicalCodeList;
##                  qb:parentChildProperty ?pcp .
##     FILTER( isBlank(?pcp) )
##     ?pcp  owl:inverseOf ?p .
##     FILTER( isIRI(?p) )
## }
##
## rq
## ASK {
##     ?obs qb:dataSet/qb:structure/qb:component/qb:componentProperty ?dim .
##     ?dim a qb:DimensionProperty ;
##          qb:codeList ?list .
##     ?list a qb:HierarchicalCodeList .
##     ?obs ?dim ?v .
##     FILTER NOT EXISTS { ?list qb:hierarchyRoot/(^<$p>)* ?v }
## }
##
```

TODO(mja): show the SPARQL code in a highlight environment.

This stores the qbIClist in the data directory.

TODO(mja): Consider making qbIClist an internal data set.

TODO(mja): Extract nomalization scripts and store also.

```
devtools::use_data(qbIClist,overwrite=TRUE)
```

```
## Saving qbIClist to data/qbIClist.rda
```

```
print("Done")
```

```
## [1] "Done"
```