RDF/ttl Queries tested on the CDS triplestore for images
M.Louys 2018, Oct 26

```
PREFIX : <http://www.semanticweb.org/holzmann/ontologies/2018/5/Prov-test/>


#A
# Select all activities which implement the ActivityDescription of name 'zzz' (
in all case)
SELECT ?activity ?name WHERE {
  ?activity a :Activity .
  ?activity :describedBy/:name ?name .
  FILTER regex(?name,"*zzz*","i")
}


#B
#Select all entities generated by an activity of type "*zzz*"
SELECT ?entity ?type WHERE {
  ?entity :wasGeneratedBy/:refersTo/:describedBy/:type ?type .
  FILTER regex(?type,"*zzz*","i") .
}


#C
# Give me all activities launched with a Parameter value > *x* and
# which belong to the  *zzz* activity class.
SELECT ?activity WHERE {
  ?activity a :Activity .
  ?activity :describedBy/:type ?type .
  ?activity :hasParameter/:value ?value .
  FILTER regex(?type,"*zzz*","i") .
  FILTER (xsd:integer(?value) >= *x*) .
}


#D
# Select all activities attributed to an Agent whose role was *R*.
SELECT ?entity WHERE {
  ?entity :wasAttributedTo/:holdsRoleInTime/:withRole :*R* .
}


#E
# Select all activities attributed to an Agent whose name is *x*.
SELECT ?entity WHERE {
  ?entity :wasAttributedTo/:refersTo/:name ?name .
  FILTER regex(?name,"*x*","i") .
}


#F
# Select all entities used with the activity *A*.
SELECT ?entity WHERE {
  :*A* :used/:refersTo ?entity .
}
```

```
#G
#Select all entities together with the activities which generated them and the
parameters involved
SELECT ?entity ?activity ?ad ?adname ?adtype ?adsubtype ?addocu ?adnot ?param
?value WHERE {
  ?entity :wasGeneratedBy/:refersTo ?activity .
  OPTIONAL {
  ?activity :hasParameter ?param .
  ?param :value ?value .
    }
  ?activity :describedBy ?ad .
  ?ad :name ?adname .
  ?ad :type ?adtype .
  ?ad :subtype ?adsubtype .
  ?ad :doculink ?addocu .
  ?ad :annotation ?adnot .
}


#H#1.
#Select the number of roles an agent took in a wasAttributedTo relation
(resp. wasAssociatedWith) with respect to an Entity ( resp. activity)
SELECT ?entity ?agent (COUNT(?mid) AS ?numberOfRoles) WHERE{
  ?entity :wasAttributedTo ?node .
  ?node :refersTo ?agent .
  ?node :holdsRoleInTime/:withRole ?mid .
}
GROUP BY ?entity ?agent
ORDER BY DESC(?numberOfRoles)

#H#2.
# Select the number of roles an agent took , all together for all relations
SELECT ?agent (COUNT(?mid) AS ?numberOfRoles) WHERE{
  #?entity :wasAttributedTo ?node2 .
  #?node :refersTo ?agent .
  #?node :holdsRoleInTime/:withRole ?mid .
  ?activity :wasAssociatedWith ?node .
  ?node2 :refersTo ?agent .
  ?node2 :holdsRoleInTime/:withRole ?mid .
}
GROUP BY ?agent
ORDER BY DESC(?numberOfRoles)


#I
# Select the software name used for an activity A.
SELECT ?software WHERE {
  :*A* :describedBy/:name ?software .
  }


#J
# Give me the location of the information page (doculink) about one specific
activity A.
SELECT ?link WHERE {
 :*A* :describedBy/:doculink ?link .
}
```

```
#K
#Select the activity that generated Entity E.
SELECT ?activity WHERE {
  :*E* :wasGeneratedBy ?activity .
}


#M
#Select all activities that used Entity E.
SELECT ?activity WHERE {
 ?activity :used/:refersTo :*E* .
}


#N
#Select all agents which have used an Entity from which derived-- a link with
entity E.
SELECT ?name ?role ?relation WHERE {
 { :*E* ?relation ?x .
   ?x :refersTo ?name .
   ?x :holdsRoleInTime/:withRole ?role .
   FILTER regex(str(?relation),"wasAttributedTo","i") .
 }
 UNION
 { ?activity :used/:refersTo :*E* .
   ?activity ?relation ?y .
   ?y :refersTo ?name .
   ?y :holdsRoleInTime/:withRole ?role .
   FILTER regex(str(?relation),"wasAssociatedwith","i") .}
}


#O
# Select all agents which have used an Entity from which Entity E is derived.
(ancestors)
SELECT DISTINCT ?name WHERE {
 :*E* :wasDerivedFrom+ ?entity .
 ?activity a :Activity .
 ?activity :used/:refersTo ?entity .
 ?activity :wasAssociatedWith/:refersTo ?name .
}


#P
# select all ancestors of Entity E and give the generation level (on which
filters can apply)
SELECT ?ancester (COUNT(?mid) AS ?level) {
  :*E* :wasDerivedFrom* ?mid .
  ?mid :wasDerivedFrom+ ?ancester .
}
GROUP BY ?ancester
HAVING (COUNT(?mid) = 2)


#R
# Select all activities used in the pipeline producing Entity E
#two possible ways:
```

```
SELECT DISTINCT ?name ?activity ?entity WHERE {
 <ivo://cds/P/DSS2color#RGB_M87> :wasDerivedFrom+ ?entity .
 ?activity a :Activity .
 ?entity :wasGeneratedBy/:refersTo ?activity .
 ?activity :name ?name .
}


SELECT DISTINCT ?name ?activity ?entity WHERE {
    ?entity :wasGeneratedBy/:refersTo ?activity .
    ?activity :name ?name .
    {SELECT ?entity {
      <ivo://cds/P/DSS2color#RGB_M87> :wasDerivedFrom+ ?entity .
      }
            }
}

#Queries in ttl / complements

#1
# Give me all agents and activities for which theses agents had the role  R.
SELECT ?activity ?name WHERE {
 ?activity :wasAssociatedWith ?s .
 ?s :holdsRoleInTime/:withRole :*R* .
 ?s :refersTo ?name .
}


#2
# Give me all entities and all agents to which they have been attributed to with
the corresponding role of each agent
SELECT ?entity ?name ?role WHERE {
 ?entity :wasAttributedTo ?s .
 ?s :holdsRoleInTime/:withRole ?role .
 ?s :refersTo ?name .
}


#3
#select all direct ancestors of an entity E, i.e. all  entities used by some
activity to produce E
SELECT ?entity WHERE {
 :*E* :wasDerivedFrom ?entity .
}


#4
# Select all agents involved in an activity that used entity E.
SELECT ?name WHERE {
 ?activity :used/:refersTo :*E* .
 ?activity :wasAssociatedWith/:refersTo ?name .
}


#5
# Give me all agents to whom entity E has been attributed
SELECT ?name WHERE {
```

```
  :*E* :wasAttributedTo/:refersTo ?name .
}


#6
# Select all Activities which have used entity E and its descendant entities.
SELECT ?activity WHERE {
 ?entity :wasDerivedFrom+ :*E* .
 ?activity a :Activity .
 ?activity :used/:refersTo ?entity .
}


#7
# Select all agents associated to an Activity A and list the Role it has taken.
SELECT ?agent ?role WHERE {
 :*A* :wasAssociatedWith ?temp .
 ?temp :refersTo ?agent .
 ?temp :holdsRoleInTime/:withRole ?role .
}


#8
#Select the Activity which generated entity E and list its description  and its
parameters
SELECT ?activity ?ad ?adname ?adtype ?adsubtype ?addocu ?adnot ?param ?value
WHERE {
  :*E* :wasGeneratedBy/:refersTo ?activity .
  ?activity :describedBy ?ad .
  ?ad :name ?adname .
  ?ad :type ?adtype .
  ?ad :subtype ?adsubtype .
  ?ad :doculink ?addocu .
  ?ad :annotation ?adnot .
  OPTIONAL {
  ?activity :hasParameter ?param .
  ?param :value ?value .
    }
}


#9
# Select all activity descriptions which generated at least  *x* entities
SELECT ?activityDescription (COUNT(?entity) AS ?numberOfGeneratedEntities)
WHERE{
  ?entity :wasGeneratedBy/:describedBy ?activityDescription .
}
GROUP BY ?activityDescription
HAVING (?numberOfGeneratedEntities>:*x*)
ORDER BY DESC(?numberOfGeneratedEntities)


#10
# Select all activities which last less than one minute
SELECT ?activity ?end ?duration WHERE {
  ?activity a :Activity .
  ?activity :name ?name .
  ?activity :endTime ?end .
```

```
  ?activity :startTime ?start .
  BIND(if(str(?end) = "","no end time entered",(
      ((YEAR(?end) - YEAR(?start))*31540000)
      + ((MONTH(?end) - MONTH(?start))*2628000)
      + ((DAY(?end) - DAY(?start))*86400)
      + ((HOURS(?end) - HOURS(?start))*3600)
      + (xsd:double(MINUTES(?end) - MINUTES(?start))*60)
      + (xsd:double(SECONDS(?end) - SECONDS(?start)))
  ))
   AS ?duration)
   FILTER(?duration<60)
}
ORDER BY DESC(?duration)
```