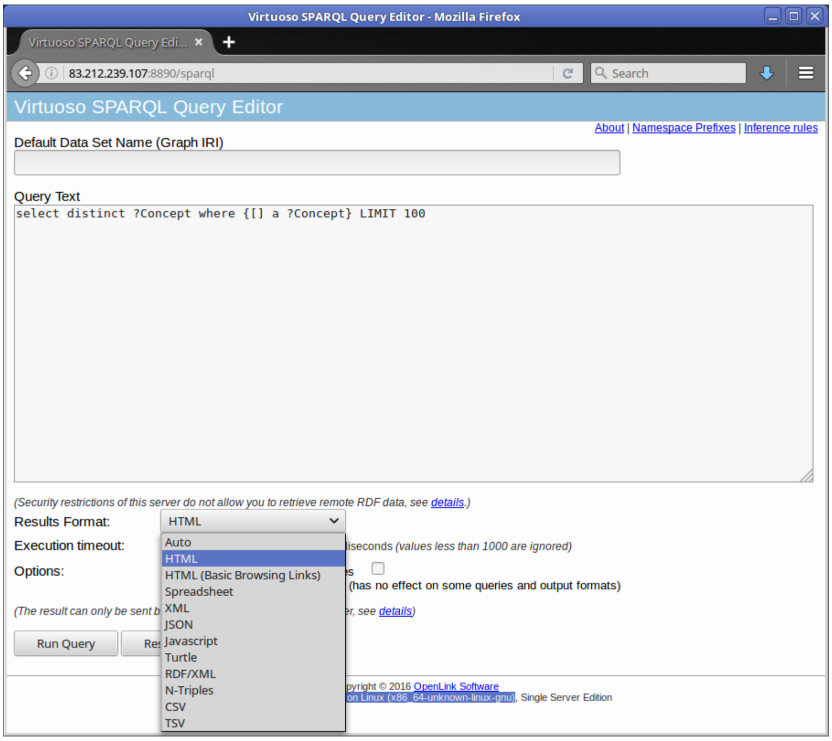


Virtuoso SPARQL Query Editor - Mozilla Firefox
Virtuoso SPARQL Query Edi...
83.212.239.107:8890/sparql
Search
Virtuoso SPARQL Query Editor
About | Namespace Prefixes | Inference rules
Auto
HTML
HTML (Basic Browsing Links)
Spreadsheet
XML
JSON
Javascript
Turtle
RDF/XML
N-Triples
CSV
TSV
Run Query
Res

Default Data Set Name (Graph IRI)

Query Text

```
select distinct ?Concept where {[] a ?Concept} LIMIT 100
```

(Security restrictions of this server do not allow you to retrieve remote RDF data, see details.)

Results Format: HTML

Execution timeout: iseconds (values less than 1000 are ignored)

Options: s □ (has no effect on some queries and output formats)

(The result can only be sent b r, see details)

pyright © 2016 OpenLink Software
on Linux (x86_64-unknown-linux-gnu), Single Server Edition

Figure 1: The web page interface of the SPARQL endpoint

Aim and Scope

The purpose of this document is to briefly demonstrate the SPARQL endpoint of the Triple Store hosting the generated RDF triples from the datasets of the initiation phase.

The Triple Store

The converted to RDF datasets are hosted on Virtuoso triple store version 07.20.3217, installed on a Linux server. This version of Virtuoso supports spatial functions, that are demonstrated in the next section.

For security reasons, access will be permitted only from authorized IP addresses. Users can access the data from `http://83.212.239.107:8890/sparql` using any web browser. The next section presents a sample of queries currently supported.

Sample Queries

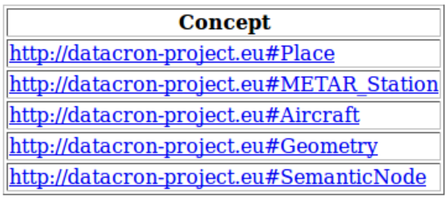
Users can write SPARQL queries in the Query text box of the endpoint web page. The SPARQL endpoint web page also allows the users to select the file format for the result set of the query to be submitted, as shown in Figure 1.

Sample query #1

The following query retrieves the concepts from the datAcron ontology that have at least one assertion in the store:

```
PREFIX datp: <http://datacron-project.eu#>
SELECT DISTINCT ?Concept WHERE
{
   GRAPH ?g { [] a ?Concept}
   FILTER(?g=<http://localhost:8890/DAV>)
}
```

The result set in HTML table format for the above query is:



Concept
http://datacron-project.eu#Place
http://datacron-project.eu#METAR_Station
http://datacron-project.eu#Aircraft
http://datacron-project.eu#Geometry
http://datacron-project.eu#SemanticNode

Sample query #2

For the datAcron ontology, a trajectory is formed by a sequence of Semantic Nodes, i.e. nodes with spatio-temporal dimensions, linked to contextual and other information. The next query, retrieves 100 URIs of the semantic nodes currently asserted in the store:

```
PREFIX datp: <http://datacron-project.eu#>
SELECT DISTINCT ?s WHERE
{
   ?s a datp:SemanticNode .
}
LIMIT 100
```

Please notice the use of LIMIT keyword, which limits the result set to be retrieved. The query:

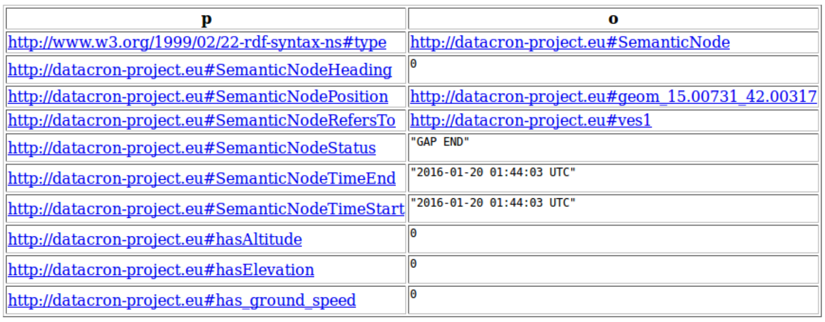
```
PREFIX datp: <http://datacron-project.eu#>
SELECT COUNT(DISTINCT ?s) WHERE
   ?s a datp:SemanticNode .
}
```

indicates that currently there are 206987 Semantic nodes in the store.

Recalling the example query for the Semantic Nodes, we can retrieve any available information linked to a specific Semantic Node, e.g. for the semantic node `datp:node_1_1453254243_1453254243_42.00317_15.00731_3`, using a query like:

```
PREFIX datp: <http://datacron-project.eu#>
SELECT * WHERE {
         datp:node_1_1453254243_1453254243_42.00317_15.00731_3 ?p ?o .
}
```

The result set in HTML table format for the above query is:



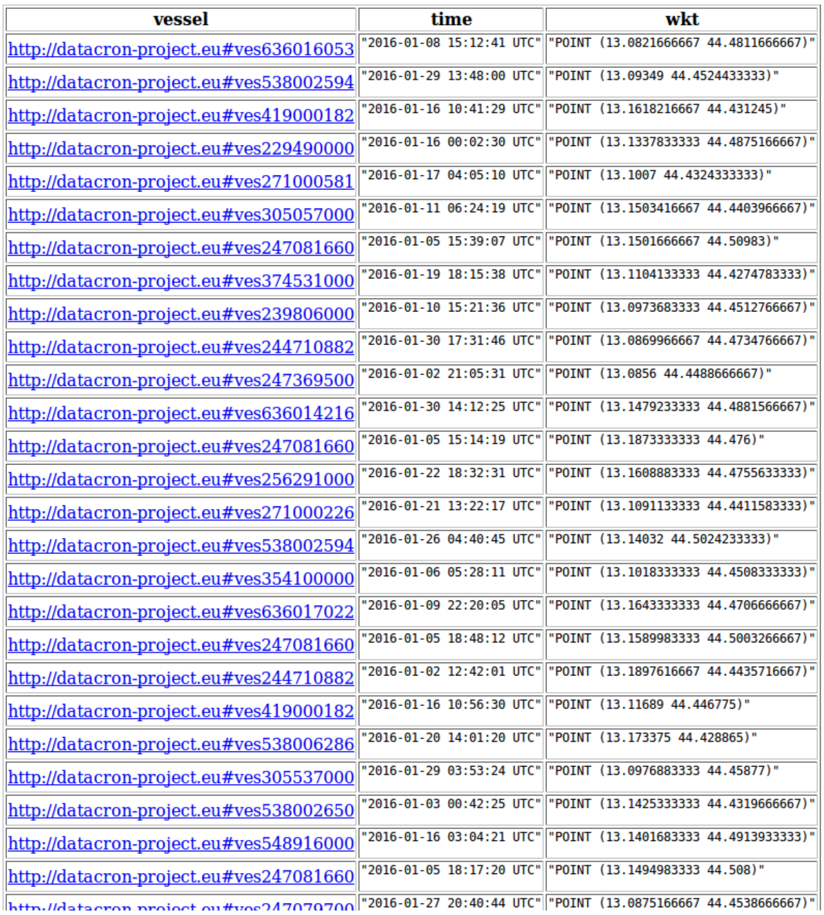
p	o
http://www.w3.org/1999/02/22-rdf-syntax-ns#type	http://datacron-project.eu#SemanticNode
http://datacron-project.eu#SemanticNodeHeading	0
http://datacron-project.eu#SemanticNodePosition	http://datacron-project.eu#geom_15.00731_42.00317
http://datacron-project.eu#SemanticNodeRefersTo	http://datacron-project.eu#ves1
http://datacron-project.eu#SemanticNodeStatus	"GAP END"
http://datacron-project.eu#SemanticNodeTimeEnd	"2016-01-20 01:44:03 UTC"
http://datacron-project.eu#SemanticNodeTimeStart	"2016-01-20 01:44:03 UTC"
http://datacron-project.eu#hasAltitude	0
http://datacron-project.eu#hasElevation	0
http://datacron-project.eu#has_ground_speed	0

Sample query #3

We can submit spatial queries to the query engine, using the predefined functions. For example, to retrieve all the vessels that had been (at some time) within a range of 5Km from the point 13.13904999999999923 44.46613333330000017, we can submit the following query:

```
PREFIX datp: <http://datacron-project.eu#>
SELECT ?vessel ?time ?wkt WHERE {
        ?s1 a datp:SemanticNode ;
        datp:SemanticNodeRefersTo ?vessel ;
        datp:has_ground_speed ?speed ;
        datp:SemanticNodeTimeStart ?time ;
        datp:SemanticNodePosition ?g .
        ?g datp:hasMBR_WKT ?wkt .
        FILTER(bif:st_distance(
            bif:st_geomfromtext (
              ”POINT(13.13904999999999923 44.46613333330000017)”),
            bif:st_geomfromtext(?wkt))<=5)
}
```

Part of the result set in HTML table format for the above query is shown in the following figure:

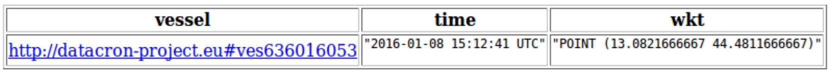


vessel	time	wkt
http://datacron-project.eu#ves636016053	"2016-01-08 15:12:41 UTC"	"POINT (13.0821666667 44.4811666667)"
http://datacron-project.eu#ves538002594	"2016-01-29 13:48:00 UTC"	"POINT (13.09349 44.4524433333)"
http://datacron-project.eu#ves419000182	"2016-01-16 10:41:29 UTC"	"POINT (13.1618216667 44.431245)"
http://datacron-project.eu#ves229490000	"2016-01-16 00:02:30 UTC"	"POINT (13.1337833333 44.4875166667)"
http://datacron-project.eu#ves271000581	"2016-01-17 04:05:10 UTC"	"POINT (13.1007 44.4324333333)"
http://datacron-project.eu#ves305057000	"2016-01-11 06:24:19 UTC"	"POINT (13.1503416667 44.4403966667)"
http://datacron-project.eu#ves247081660	"2016-01-05 15:39:07 UTC"	"POINT (13.1501666667 44.50983)"
http://datacron-project.eu#ves374531000	"2016-01-19 18:15:38 UTC"	"POINT (13.1104133333 44.4274783333)"
http://datacron-project.eu#ves239806000	"2016-01-10 15:21:36 UTC"	"POINT (13.0973683333 44.4512766667)"
http://datacron-project.eu#ves244710882	"2016-01-30 17:31:46 UTC"	"POINT (13.0869966667 44.4734766667)"
http://datacron-project.eu#ves247369500	"2016-01-02 21:05:31 UTC"	"POINT (13.0856 44.4488666667)"
http://datacron-project.eu#ves636014216	"2016-01-30 14:12:25 UTC"	"POINT (13.1479233333 44.4881566667)"
http://datacron-project.eu#ves247081660	"2016-01-05 15:14:19 UTC"	"POINT (13.1873333333 44.476)"
http://datacron-project.eu#ves256291000	"2016-01-22 18:32:31 UTC"	"POINT (13.1608883333 44.4755633333)"
http://datacron-project.eu#ves271000226	"2016-01-21 13:22:17 UTC"	"POINT (13.1091133333 44.4411583333)"
http://datacron-project.eu#ves538002594	"2016-01-26 04:40:45 UTC"	"POINT (13.14032 44.5024233333)"
http://datacron-project.eu#ves354100000	"2016-01-06 05:28:11 UTC"	"POINT (13.1018333333 44.4508333333)"
http://datacron-project.eu#ves636017022	"2016-01-09 22:20:05 UTC"	"POINT (13.1643333333 44.4706666667)"
http://datacron-project.eu#ves247081660	"2016-01-05 18:48:12 UTC"	"POINT (13.1589983333 44.5003266667)"
http://datacron-project.eu#ves244710882	"2016-01-02 12:42:01 UTC"	"POINT (13.1897616667 44.4435716667)"
http://datacron-project.eu#ves419000182	"2016-01-16 10:56:30 UTC"	"POINT (13.11689 44.446775)"
http://datacron-project.eu#ves538006286	"2016-01-20 14:01:20 UTC"	"POINT (13.173375 44.428865)"
http://datacron-project.eu#ves305537000	"2016-01-29 03:53:24 UTC"	"POINT (13.0976883333 44.45877)"
http://datacron-project.eu#ves538002650	"2016-01-03 00:42:25 UTC"	"POINT (13.1425333333 44.4319666667)"
http://datacron-project.eu#ves548916000	"2016-01-16 03:04:21 UTC"	"POINT (13.1401683333 44.4913933333)"
http://datacron-project.eu#ves247081660	"2016-01-05 18:17:20 UTC"	"POINT (13.1494983333 44.508)"
http://datacron-project.eu#ves247079700	"2016-01-27 20:40:44 UTC"	"POINT (13.0875166667 44.4538666667)"

Obviously, we can filter also the temporal dimension, i.e. on the variable `?time`. For example, to retrieve only those vessels that had been around the point 13.139049999999999923 44.466133333330000017 and between `"2016-01-08 14:12:41"` and `"2016-01-08 16:12:41"`, the query will be:

```
PREFIX datp: <http://datacron-project.eu#>
SELECT ?vessel ?time ?wkt WHERE {
    ?s1 a datp:SemanticNode ;
    datp:SemanticNodeRefersTo ?vessel ;
    datp:has_ground_speed ?speed ;
    datp:SemanticNodeTimeStart ?time ;
    datp:SemanticNodePosition ?g .
    ?g datp:hasMBR_WKT ?wkt .
    FILTER(
        bif:st_distance(
            bif:st_geomfromtext (
              "POINT(13.139049999999999923 44.466133333330000017)"),
            bif:st_geomfromtext(?wkt))<=5 &&
        xsd:dateTime(substr(?time,1,19))<xsd:dateTime("2016-01-08 16:12:41") &&
        xsd:dateTime(substr(?time,1,19))>xsd:dateTime("2016-01-08 14:12:41")
    )
}
```

The result set in HTML table format for the above query is:

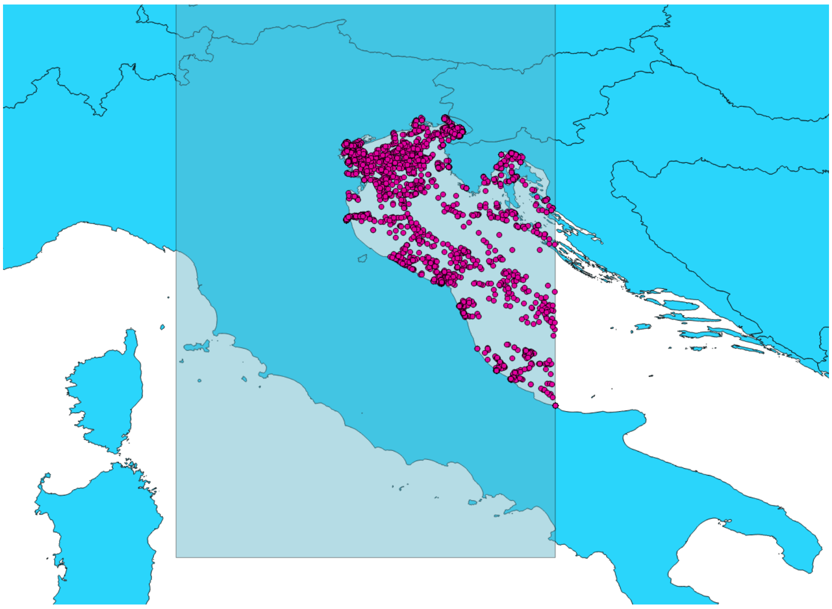


vessel	time	wkt
http://datacron-project.eu#ves636016053	"2016-01-08 15:12:41 UTC"	"POINT (13.0821666667 44.4811666667)"

A similar spatial query, is to retrieve all the semantic nodes of vessels that have crossed a specific area. In this case, the geometry of the area of interest is required, i.e. it can be any geometry related to a resource in the store (e.g. a Natura2000 region), or it can be manually defined as a Well-Known-Text. For example, the following query will retrieve all the vessels within an area defined by `"POLYGON((10 50,15 50,15 40,10 40,10 50))"`, for `"2016-01-08"`:

```
PREFIX datp: <http://datacron-project.eu#>
SELECT ?vessel ?time ?wkt WHERE {
        ?s a datp:SemanticNode ;
        datp:SemanticNodeRefersTo ?vessel ;
        datp:has_ground_speed ?speed ;
        datp:SemanticNodeHeading ?heading ;
        datp:SemanticNodeTimeStart ?time ;
        datp:SemanticNodePosition ?g .
        ?g datp:hasMBR_WKT ?wkt .
        FILTER( bif:st_intersects (
           bif:st_geomfromtext ( "POLYGON((10 50,15 50,15 40,10 40,10 50))" ),
           bif:st_geomfromtext(?wkt)) &&
        xsd:dateTime(substr(?time,1,10))=xsd:dateTime("2016-01-08")
        )
}
```

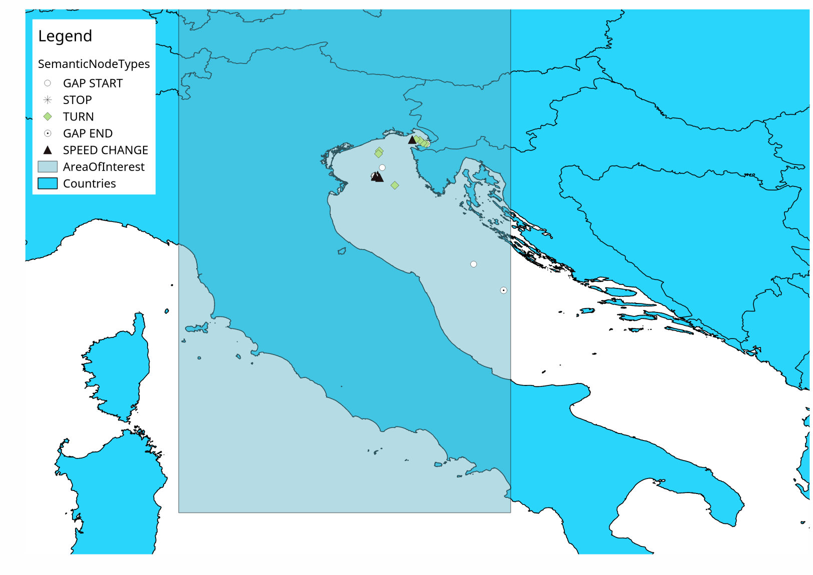
The result set exported to CSV and visualized is shown in the following figure:



A refinement of the above query, will be the retrieval of all the semantic nodes in the area of interest, for a specific day/time (or interval), and for a specific vessel. For example, we can request the semantic nodes of vessel `datp:ves209996000`, and the status of the vessel at each semantic node, for the above area and any time after `"2016-01-08"`, ordered by `?time`:

```
PREFIX datp: <http://datacron-project.eu#>
SELECT ?time ?status ?wkt WHERE {
    ?s a datp:SemanticNode ;
    datp:SemanticNodeRefersTo datp:ves209996000 ;
    datp:has_ground_speed ?speed ;
    datp:SemanticNodeStatus ?status ;
    datp:SemanticNodeHeading ?heading ;
    datp:SemanticNodeTimeStart ?time ;
    datp:SemanticNodePosition ?g .
    ?g datp:hasMBR_WKT ?wkt .
    FILTER(bif:st_intersects (
        bif:st_geomfromtext (
          ”POLYGON((10 50,15 50,15 40,10 40,10 50))” ),
          bif:st_geomfromtext(?wkt)) &&
        xsd:dateTime(substr(?time,1,10))>=xsd:dateTime(”2016-01-08”)
    )
} ORDER BY ?time
```

The result set exported to CSV and visualized is shown in the following figure:


Legend
SemanticNodeTypes
○ GAP START
✳ STOP
◆ TURN
⊙ GAP END
▲ SPEED CHANGE
▨ AreaOfInterest
▨ Countries