

Tutorial on RDF Stream Processing

M. Balduini, J-P Calbimonte, O. Corcho,
D. Dell'Aglio, E. Della Valle

<http://streamreasoning.org/rsp2014>



SPARQLstream and Morph- streams: Hands on Session

Jean-Paul Calbimonte & Oscar Corcho



Share, Remix, Reuse — Legally

- This work is licensed under the Creative Commons Attribution 3.0 Unported License.

- **Your are free:**



to Share — to copy, distribute and transmit the work



to Remix — to adapt the work

- **Under the following conditions**



Attribution — You must attribute the work by inserting

- “[source <http://streamreasoning.org/sr4ld2013>]” at the end of each reused slide
- a credits slide stating
 - These slides are partially based on “Streaming Reasoning for Linked Data 2013” by M. Balduini, J-P Calbimonte, O. Corcho, D. Dell'Aglio, E. Della Valle, and J.Z. Pan <http://streamreasoning.org/sr4ld2013>

- To view a copy of this license, visit <http://creativecommons.org/licenses/by/3.0/>

- What we will cover:
 - SPARQLstream queries
 - Register queries
 - Pull data
 - Push data

- Morph-web: a demo web application for Morph-streams
 - <https://github.com/jpcik/morph-web>
 - Install it yourself (follow the instructions in github)

- The instructions are on the github wiki:
 - <https://github.com/jpcik/morph-web/wiki/Tutorial:-Morph-streams>

- We'll be using this server for the hands-on:
 - <http://linkeddata2.dia.fi.upm.es:9000>
 - If port 9000 is blocked:
 - <http://streams.linkeddata.es>

- You can choose one of the use cases in the Demo home:



The screenshot shows the Morph-streams Web demo home page. At the top, there is a navigation bar with the Morph-streams logo on the left and 'Home' and 'About' links on the right. Below the navigation bar, the main heading reads 'Morph-streams Web demo'. Underneath, it says 'Choose a demo System:' followed by a bulleted list of five demo systems, each with a blue underlined link and a note about the system it runs on.

morph
streams

Home About

Morph-streams Web demo

Choose a demo System:

- [Social Sensor Demo \(running Esper\)](#)
- [EMT Bus stations Madrid \(running GSN\)](#)
- [Citybikes urabn Bike sensors \(running GSN\)](#)
- [Swiss Experiment environmental data \(running GSN\)](#)
- [HL7 synthetic patient data \(running Esper\)](#)

- In short: People detected in rooms
- Use Esper as datasource

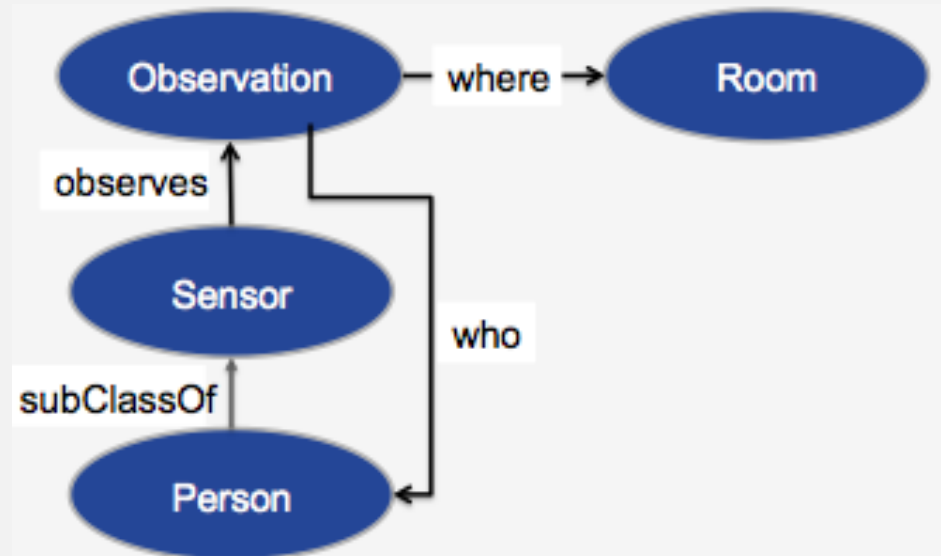
detections {roomid:string, person:string, time:string}

For example, this stream may contain tuples as the following:

r1,alice,2013-10-10T10:00

But of course we want to query this data through an ontology...

Let's use this ontology:



Oversimplified ontology: an observation encapsulates something that a sensor has observed.

who was observed (a person), and **where** (in a room).

- Go to MORPH_HOST/query/social.
- Write a query or choose one
- e.g. all observations when carl was detected in the last 30 seconds:

```
PREFIX sr41d: <http://streamreasoning.org/ontologies/social#>
PREFIX pers: <http://streamreasoning.org/data/person/id/>
SELECT ?obs
FROM NAMED STREAM <http://streamreasoning.org/data/social.srdf>
[NOW - 30 S]
WHERE {
  ?obs sr41d:who pers:carl.
}
```

- Only registered the query. to see some data pull results.

- The query has been given an identifier
- Can be used to retrieve results by pulling.



The screenshot shows the 'morph streams' web interface. At the top, there is a navigation bar with the 'morph streams' logo and links for 'Home' and 'About'. Below the navigation bar, the system ID is displayed as 'System: fe966dc1-4b70-4fdf-81a8-a8ac7a76acdb'. Underneath, the label 'qid' is followed by a text input field containing the value 'fe966dc1-4b70-4fdf-'. Below the input field, the word 'Required' is written. At the bottom of the form, there are two buttons: 'Pull' and 'Remove'.

- You can also remove the query when you no longer need it.

- Receive results as soon as they are available
- Using a WebSocket.
- WebSockets implement full-duplex communication via TCP, and are supported by most browsers.



The screenshot shows the 'morph-streams' web application. The navigation bar includes 'Home', 'Register Query', 'Listen Push', and 'About'. The main content area is titled 'Push Query' and shows the system 'social'. A message states 'This browser supports WebSocket.' The 'Location' field contains 'ws://linkeddata2.dia.fi.upm.es:9000/pu'. Below this are 'Connect', 'Disconnect', and 'Set Query' buttons. A text area contains a SPARQLStream query:

```
PREFIX sr4Id: <http://streamreasoning.org/ontologies/social#>
SELECT ?obs
FROM NAMED STREAM
<http://streamreasoning.org/data/social.srdf> [NOW - 30 S]
WHERE {
  ?obs a sr4Id:Observation.
}
```

 To the right, a 'Log' section shows a 'CONNECTED' status followed by two error messages: 'RESPONSE: Fatal Error, please disconnect and try again.' and 'RESPONSE: Error: could not create: com.hp.hpl.jena.query.QueryException: Encountered "" at line 1, column 0. Was expecting one of: "base" ... "prefix" ... "select" ... "describe" ... "construct" ... "ask" ...'. The log ends with a 'DISCONNECTED' status.

`ws://linkeddata2.dia.fi.upm.es:9000/push?query=PREFIX%20sr41....`

For example you can change the URI template for a Person, instead of this predicate map:

```
rr:predicateObjectMap [  
  rr:predicate sr41d:who;  
  rr:objectMap [rr:template  
"http://streamreasoning.org/data/person/id/{person}"]];
```

You can define the following:

```
rr:predicateObjectMap [  
  rr:predicate sr41d:who;  
  rr:objectMap [rr:template  
"http://someotherplace.org/persons/Person/{person}"]];
```

- Underlying queries checkbox
- To see what is being sent to the DSMS or CEP

- Using GSN
- Instantaneous one-off queries
- get all bus stop observations in the last 5 mins:

```
PREFIX ssn: <http://purl.oclc.org/NET/ssnx/ssn#>
PREFIX qudt: <http://data.nasa.gov/qudt/owl/qudt#>
PREFIX emt: <http://emt.linkeddata.es/data#>
SELECT ?timeto ?obs ?av
FROM NAMED STREAM <http://emt.linkeddata.es/data#busstops.srdf>
[NOW - 300 S]
WHERE {
    ?obs a emt:BusObservation.
    ?obs ssn:observationResult ?output.
    ?output emt:timeToBusValue ?av.
    ?av qudt:numericValue ?timeto.
}
```

- Fire and forget

[Home](#)[One-Off Query](#)[About](#)

System: emt

Results

timeto	obs
999999^^http://www.w3.org/2001/XMLSchema#string	http://transporte.linkeddata.es/emt/busstop/id/66/busline/14/observation/20/10/2013%2014:53:42%20%2B0200
999999^^http://www.w3.org/2001/XMLSchema#string	http://transporte.linkeddata.es/emt/busstop/id/66/busline/150/observation/20/10/2013%2014:53:42%20%2B0200
999999^^http://www.w3.org/2001/XMLSchema#string	http://transporte.linkeddata.es/emt/busstop/id/66/busline/5/observation/20/10/2013%2014:53:42%20%2B0200
999999^^http://www.w3.org/2001/XMLSchema#string	http://transporte.linkeddata.es/emt/busstop/id/66/busline/45/observation/20/10/2013%2014:53:42%20%2B0200
999999^^http://www.w3.org/2001/XMLSchema#string	http://transporte.linkeddata.es/emt/busstop/id/66/busline/14/observation/20/10/2013%2014:53:42%20%2B0200
994^^http://www.w3.org/2001/XMLSchema#string	http://transporte.linkeddata.es/emt/busstop/id/66/busline/27/observation/20/10/2013%2014:53:42%20%2B0200
394^^http://www.w3.org/2001/XMLSchema#string	http://transporte.linkeddata.es/emt/busstop/id/66/busline/150/observation/20/10/2013%2014:53:42%20%2B0200
367^^http://www.w3.org/2001/XMLSchema#string	http://transporte.linkeddata.es/emt/busstop/id/66/busline/45/observation/20/10/2013%2014:53:42%20%2B0200
320^^http://www.w3.org/2001/XMLSchema#string	http://transporte.linkeddata.es/emt/busstop/id/66/busline/5/observation/20/10/2013%2014:53:42%20%2B0200

- MORPH_HOST/emt/sparqlstream?query=ENCODEDQUERY
- `ENCODEDQUERY` is the SPARQLStream encoded for a URL. E.g.:

```
http://linkeddata2.dia.fi.upm.es:9000/emt/sparqlstream?query=PREFIX%20ssn%3A%20%3Chttp%3A//purl.oclc.org/NET/ssnx/ssn%23%3E%0APREFIX%20qudt%3A%20%3Chttp%3A//data.nasa.gov/qudt/owl/qudt%23%3E%0APREFIX%20emt%3A%20%3Chttp%3A//emt.linkeddata.es/data%23%3E%0ASELECT%20%3Ftimeto%20%3Fobs%20%3Fav%20%0AFROM%20NAMED%20STREAM%20%3Chttp%3A//emt.linkeddata.es/data%23busstops.srdf%3E%20%5BNOW%20-%20300%20S%5D%0AWHERE%20%7B%0A%20%20%3Fobs%20a%20emt%3ABusObservation.%0A%20%20%3Fobs%20ssn%3AobservationResult%20%3Foutput.%0A%20%20%20%3Foutput%20emt%3AtimeToBusValue%20%3Fav.%0A%20%20%20%3Fav%20qudt%3AnumericValue%20%3Ftimeto.%0A%7D
```

A bit ugly but it's a kind of SPARQLstream endpoint

```
{
  "head": {
    "vars": [ "timeto" , "obs" , "av" ]
  } ,
  "results": {
    "bindings": [
      {
        "timeto": { "datatype":
"http://www.w3.org/2001/XMLSchema#string" , "type": "typed-
literal" , "value": "999999" } ,
        "obs": { "type": "uri" , "value":
"http://transporte.linkeddata.es/emt/busstop/id/44/busline/147/
observation/20/10/2013%2010:35:38%20%2B0200" } ,
        "av": { "type": "uri" , "value":
"http://transporte.linkeddata.es/emt/busstop/id/44/busline/147/
timeToBusValue/20/10/2013%2010:35:38%20%2B0200" }
      } ,
    ]
  }
}
```

Add a predicate object map

```
rr:predicateObjectMap [  
  rr:predicate sr41d:when;  
  rr:objectMap [rr:column "time"]];
```

Tutorial on RDF Stream Processing

M. Balduini, J-P Calbimonte, O. Corcho,
D. Dell'Aglio, E. Della Valle

<http://streamreasoning.org/rsp2014>



Morph-streams: Hands on Session

