SPARQLStream: Ontology-based access to data streams

Jean-Paul Calbimonte, Oscar Corcho
jp.calbimonte@upm.es, ocorcho@fi.upm.es
http://www.oeg-upm.net/
Share, Remix, Reuse — Legally

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

- **You 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/
SPARQLStream

- Virtual RDF views over data streams
- Ontology-based access to data streams
  - Examples
  - Architecture
  - Underlying query processors
- SPARQLStream language
- Query rewriting
  - R2RML mappings
- Resources
Querying RDF Streams

Where is the data coming from?

Existing streaming data sources:
- DSMS
- CEP
- Sensor middleware
- ...
Virtual RDF views over data streams

users, applications

query processing

data layer

DSMS

CEP

Sensor middleware

...
Already seen somewhere...?

- RDF DBMS
- Load, import
- SPARQL Query Processor
- R2RML Mappings
- R2RML Mappings

http://streamreasoning.org/rsp2014
Stream Processor Implementations

Data Stream Management Systems (DSMS)
- CQL/Stream
- Borealis
- StreamMill
- NiagaraCQ
- TelegraphCQ

Complex Event Processors (CEP)
- GEM
- Rapide
- Cayuga
- CEDR
- Esper
- Oracle CEP
- IBM InfoSphere
- Microsoft StreamInsight
- Sybase CEP
- StreamBase

Stream Data Middleware
- Hourglass
- Cosm
- SStreamWare
- GSN

Diverse query languages
Different query capabilities
Different query models
Morph-streams: Overview

SELECT ?proximity
FROM STREAM
WHERE {
  ?obs a ssn:ObservationValue;
  qudt:numericalValue ?proximity;
  FILTER (?proximity > 10)
}

$\pi_{\text{timed}, \text{prox}}$

$\sigma_{\text{prox} > 10}$

$\text{5 Seconds}$

$\text{sens}$

SELECT prox
FROM sens.win:time(5 sec)
WHERE prox > 10

Morph-streams processing SPARQLStream queries

http://streamreasoning.org/rsp2014

https://github.com/jpcik/morph-streams
SPARQLStream Language

Query Form

CONSTRUCT  DESCRIBE  SELECT

Dataset Clause
FROM NAMED

Where Clause (Graph Pattern)

Triple pattern
FILTER
OPTIONAL AND UNION

FROM NAMED STREAM
WINDOW

ISTREAM
DSTREAM
RSTREAM

Underlying data source restrictions

http://streamreasoning.org/rsp2014
SPARQLStream: examples

**SPARQLStream**

PREFIX sr4ld: <http://www.streamreasoning.org/ontologies/socialsensor,owl#>
SELECT ?room
FROM NAMED STREAM <http://www.streamreasoning.org/streams/socialsensor.srdf> [NOW-10 S]
WHERE {
  ?obs sr4ld:observedBy ?sensor.
}

All rooms where something was observed in the last 10s

PREFIX sr4ld: <http://www.streamreasoning.org/ontologies/socialsensor,owl#>
SELECT (COUNT(?person) AS ?nmb) ?room
FROM NAMED STREAM <http://www.streamreasoning.org/streams/socialsensor.srdf> [NOW-10 S]
WHERE {
}
GROUP BY ?room

Number of persons observed in each room in the last 10s
SPARQLStream Language

- **NamedStream** → ‘FROM’ [‘NAMED’] ‘STREAM’ StreamIRI ‘[’ Window ‘]’
- **Window** → ‘NOW-’ Integer TimeUnit [UpperBound] [Slide]
- **UpperBound** → ‘TO NOW-’ Integer TimeUnit
- **Slide** → ‘SLIDE’ Integer TimeUnit
- **TimeUnit** → ‘MS’ | ‘S’ | ‘MINUTES’ | ‘HOURS’ | ‘DAY’

SELECT ISTREAM ?room
FROM NAMED STREAM <http://www.streamreasoning.orgstreams/socialsensor.srdff> [NOW-10 S]
WHERE {...

- **Select** → ‘SELECT’ [Xstream] [Distinct | Reduced] ...
- **Xstream** → ‘RSTREAM’ | ‘ISTREAM’ | ‘DSTREAM’
Morph-streams: Overview

SELECT ?proximity
FROM STREAM
<http://streamreasoning.org/SensorReadings.srdf> [NOW−5 S]
WHERE {
  ?obs a ssn:ObservationValue;
  qudt:numericalValue ?proximity;
  FILTER (?proximity > 10)}

π timed,prox
φ prox>10
Ω 5 Seconds
sens

SELECT prox
FROM sens.win:time(5 sec)
WHERE prox > 10

Client

Query rewriting

SPARQL-Stream

R2RML Mappings

Data translation

[tuples/bin dings]

Query Processing

Algebra expression

[tuples]

pull/push

SNEE

Esper

GSN

Cosm

Other

Morph-streams processing SPARQL-Stream queries

https://github.com/jpcik/morph-streams

http://streamreasoning.org/rsp2014
Now, where is the data?

sensor

\begin{align*}
\tau_i \quad (\text{alice, room1}) \\
\tau_{i-1} \quad (\text{alice, room1}) \\
\tau_{i+1} \quad (\text{carl, room1}) \\
\tau_{i+1} \quad (\text{luke, room1}) \\
\tau_{i+1} \quad (\text{bob, room2}) \\
\tau_i \quad (\text{alice, room3}) \\
\tau_{i+1} \quad (\text{luke, room1}) \\
\end{align*}

Stream Schema

DSMS, CEP, middleware can evaluate queries over this model.
<table>
<thead>
<tr>
<th>Underlying Query Processors</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Esper</strong></td>
</tr>
<tr>
<td>• CEP/DSMS</td>
</tr>
<tr>
<td>• EPL language</td>
</tr>
<tr>
<td>SELECT prox FROM sensors.win:time(5 minute) WHERE prox &gt;10</td>
</tr>
<tr>
<td><strong>SNEE</strong></td>
</tr>
<tr>
<td>• DSMS/Sensor Network Query Evaluator</td>
</tr>
<tr>
<td>• Compile queries to sensor code</td>
</tr>
<tr>
<td>SELECT prox FROM sensors [FROM NOW-5 MINUTES TO NOW] WHERE prox &gt;10</td>
</tr>
<tr>
<td><strong>GSN</strong></td>
</tr>
<tr>
<td>• Sensor middleware</td>
</tr>
<tr>
<td>• REST API</td>
</tr>
<tr>
<td><a href="http://montblanc.sfl.ch:22001/multidata?vs%5B0%5D=sensors&amp;field%5B0%5D=proximity_field&amp;c_min%5B0%5D=10&amp;from=15/05/2012+05:00:00&amp;to=15/05/2012+10:00:00">http://montblanc.sfl.ch:22001/multidata?vs[0]=sensors&amp;field[0]=proximity_field&amp;c_min[0]=10&amp;from=15/05/2012+05:00:00&amp;to=15/05/2012+10:00:00</a></td>
</tr>
<tr>
<td><strong>Cosm/Xively</strong></td>
</tr>
<tr>
<td>• Sensor middleware</td>
</tr>
<tr>
<td>• Open platform</td>
</tr>
<tr>
<td>• REST API</td>
</tr>
<tr>
<td><a href="http://api.cosm.com/v2/feeds/14321/datastreams/4?start=2012-05-15T05:00:00Z&amp;end=2012-05-15T10:00:00Z">http://api.cosm.com/v2/feeds/14321/datastreams/4?start=2012-05-15T05:00:00Z&amp;end=2012-05-15T10:00:00Z</a></td>
</tr>
</tbody>
</table>
SELECT ?proximity
FROM STREAM <http://streamreasoning.org/SensorReadings.srdf>
[NOW–5 S]
WHERE {
  ?obs a ssn:ObservationValue;
  qudt:numericalValue ?proximity;
  FILTER (?proximity>10) }

SELECT prox FROM sensors [FROM NOW-5 MINUTES TO NOW]
WHERE prox >10

SELECT prox FROM sensors.win:time(5 minute)
WHERE prox >10

http://montblanc.slf.ch:22001/multidata?vs[0]=sensors&field[0]=proximity_field&c_min[0]=10&from=15/05/2012+05:00:00&to=15/05/2012+10:00:00

http://api.cosm.com/v2/feeds/14321/datastreams/4?start=2012-05-15T05:00:00Z&end=2012-05-15T10:00:00Z

SNEE (DSMS)

Esper (CEP)

GSN (middlwr)

Cosm Xively
## Underlying query processors

<table>
<thead>
<tr>
<th>Features</th>
<th>Esper</th>
<th>SNEE</th>
<th>GSN</th>
<th>Cosm/Xively</th>
</tr>
</thead>
<tbody>
<tr>
<td>Projection</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>Fixed</td>
</tr>
<tr>
<td>Proj expression</td>
<td>✔</td>
<td>✔</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>Joins</td>
<td>✔</td>
<td>✔ ✗</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>Union</td>
<td>✗</td>
<td>✔ ✗</td>
<td>✔</td>
<td>✗</td>
</tr>
<tr>
<td>Selection</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✗ ✗ limited</td>
</tr>
<tr>
<td>Aggregates</td>
<td>✔</td>
<td>✔</td>
<td>✔ ✗</td>
<td>✗</td>
</tr>
<tr>
<td>Time window</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Tuple window</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✗</td>
</tr>
<tr>
<td>R2S</td>
<td>✔</td>
<td>✔</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>Conjunction, Disj</td>
<td>✔</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>Repetition pattern</td>
<td>✔</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>Sequence</td>
<td>✔</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
</tr>
</tbody>
</table>
Configuring Morph-streams

- Main ingredients:

1. Data streams
2. Ontology (network)
3. R2RML mappings

Link both models
SSN Ontology with other ontologies

W3C SSN Ontology

modeling our streaming data
combine with domain ontologies
Our simpler ontology...

We can use different ontologies for the same data
Our simpler ontology...

Observation \( \xrightarrow{\text{where}} \) Room

observes

Sensor \( \xrightarrow{\text{subClassOf}} \) Person

Define mappings

detections

(person, room,...)
R2RML - Overview
R2RML - Overview
Encoding in R2RML

Mapping definition

:triplesMap a rr:TriplesMap;
   rr:logicalTable [ rr:tableName "sensors" ];

   rr:subjectMap [ 
      rr:template "http://streamreasoning.org/data/Observation/{person}{timed}";
      rr:class sr4ld:Observation; rr:graph sr4ld:socialstream.srdf ];

   rr:predicateObjectMap [ 
      rr:predicate sr4ld:who ;
      rr:objectMap [ rr:template "http://streamreasoning.org/data/Person/{person}" ];
   ];

the stream name

subject URI

stream attributes

triple predicate + object

the object (a URI in this case)
Now some code

Morph-streams:
- Coded in **Scala**
- JAR bundle, use it from Scala or **Java** code
- Maven, Sbt
- Examples
  - One off query
  - Register continuous query
  - Pull data
  - Push
  - Basic REST
- **https://github.com/jpcik/morph-streams**
### Code examples

- **Parse SPARQLStream**

  ```scala
  val query = "PREFIX sr4ld: <...>. SELECT ?a ..."
  val syntax = StreamQueryFactory.create(query);
  ```

- **Execute One-off query**

  ```scala
  val query = "PREFIX sr4ld: <...>. SELECT ?a ..."
  mapping = Mapping(new URI(mappings/social.ttl))
  val adapter: QueryEvaluator = Application.adapter(system)
  val results = adapter.executeQuery(query, mapping)
  ```
## Code examples

### Register and Pull

```scala
val queryid = adapter.registerQuery(query, mapping)
val results1 = adapter.pull(queryid)
val results2 = adapter.pull(queryid)
```

### Register and Push

```scala
class ExampleReceiver extends StreamReceiver{
  override def receiveData(s:SparqlResults):Unit =
    Logger.debug("got: "+res)
}
val receiver = new ExampleReceiver
val queryid = adapter.listenToQuery(query, mapping, receiver)
```
### SPARQLStream from command line


  Just encoding query

- `curl "http://streams.linkeddata.es/emt/sparqlstream?query=$encoded_value"

**Disclaimer:** Simplistic, not implementing all of the SPARQL protocol
{  
  "head": {  
    "vars": [ "timeto" , "obs" ]  
  } ,  
  "results": {  
    "bindings": [  
      {  
        "timeto": { "datatype": "http://www.w3.org/2001/XMLSchema#string" , "type": "typed-literal" , "value": "0" } ,  
      }  
    ]  
  }  
}
Resources

- Morph-Streams
  - https://github.com/jpcik/morph-streams

- See demos
  - http://transporte.linkeddata.es/ (SPARQL-Stream tab)
  - Check our Madrid buses demo at SSN2013 workshop tomorrow

- Read out more

- Contact point
  - jp.calbimonte@upm.es
  - ocorcho@fi.upm.es
SPARQLStream: Ontology-based access to data streams

Jean Paul Calbimonte, Oscar Corcho
jp.calbimonte@upm.es, ocorcho@fi.upm.es
http://www.oeg-upm.net/
For streams?

\[(s,p,o, \tau)\]

(time-stamped triples)

- Gutierrez et al. (2007) Introducing time into RDF. IEEE TKDE
- Rodríguez et al. (2009) Semantic management of streaming data. SSN