Stream Reasoning for Linked Data
M. Balduini, J-P Calbimonte, O. Corcho, D. Dell’Aglio, E. Della Valle
http://streamreasoning.org/events/sr4ld2014

Morph-streams: SPARQLStream OBDA in action
Jean-Paul Calbimonte
Jean-paul.calbimonte@epfl.ch
Lsir.epfl.ch
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/events/sr4ld2014]” at the end of each reused slide
    - a credits slide stating
      - These slides are partially based on “Streaming Reasoning for Linked Data 2014” by M. Balduini, J-P Calbimonte, O. Corcho, D. Dell'Aglio, and E. Della Valle, http://streamreasoning.org/events/sr4ld2014

- To view a copy of this license, visit
  http://creativecommons.org/licenses/by/3.0/
Querying RDF Streams

Where is the data coming from?

Existing streaming data sources:
- DSMS
- CEP
- Sensor middleware
- ...

Streams already exist in non-RDF form
Feed the RDF Stream: Scenarios

Stream generated by a Sensor

RDF Stream Processor

Different RDF representations
Dynamic structural changes
Multiple RDF views over the same stream

bob, room1, t1
Carl, room1, t2
Carl, room2, t3

person/bob, room1, t1

person/bob, observedBy: sensor1, t1

obst1, a, ssn:Observation, t1
obst1, ssn:observedBy: sensor1, t1
obst1, ssn:featureOfInterest: bob, t1
obst1, ssn:observedValue: room1, t1

http://streamreasoning.org/events/sr4ld2014
Virtual RDF views over data streams

Users, applications

SPARQLStream

RDF Stream Processor

Virtual RDF Stream

Rewritten queries

DSMS

CEP

Sensor middleware

Morph-streams

Query processing

Data layer
Already seen somewhere…?

Similar to RDB2RDF

Stream DBMS?

Stream SQL?

Load, import

R2RML Mappings

DBMS

SPARQL Query Processor

RDF

Load, import

RDB2RDF

RDF

SPARQL

W3C

W3C

W3C
Stream Processor Implementations

Data Stream Management Systems (DSMS)

Complex Event Processors (CEP)

Stream Data Middleware

Diverse query languages
Different query capabilities
Different query models
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)
}

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

Morph-streams processing SPARQL Stream queries

http://streamreasoning.org/events/sr4ld2014
What do I need for Morph-streams

- Main ingredients:
  1. Data streams
  2. Ontology
  3. R2RML mappings

Link both models
1st: the data streams

\[
\begin{align*}
\tau_i & : (\text{bob, room2}) \\
\tau_{i+1} & : (\text{alice, room3}) \\
\tau_i & : (\text{karl, room1}) \\
\tau_{i+1} & : (\text{luke, room1}) 
\end{align*}
\]

Stream Schema

DSMS, CEP, middleware can evaluate queries over this model.
2nd: An Ontology!

W3C SSN Ontology

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

- Observation subClassOf Room
- Post subPropOf Sensor
- Sensor observes Person
- Person subClassOf Room
- Post discusses Sensor
- Person authorOf Observation

We can use different ontologies for the same data.
Define mappings

observation

Sensor

subClassOf

Person

where

Room

detections

(person, room, ...)

3rd: Mapping the two models
R2RML – There is a recommendation!

We can use the W3C recommendation
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

stream
attributes

subject URI

triple predicate + object

the object (a URI in this case)
Morph-streams: Overview

**Query Rewriting**

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

**Data Translation**

```
π timed.prox
Ω prox>10
ω 5 Seconds
sens
```

**Query Processing**

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

**Morph-streams processing SPARQL Stream queries**

[http://streamreasoning.org/events/sr4ld2014](http://streamreasoning.org/events/sr4ld2014)
SPARQLStream Language

Query Form

CONSTRUCT  DESCRIBE  SELECT

FROM NAMED STREAM
ISTREAM DSTREAM RSTREAM

WHERE Clause (Graph Pattern)

Underlying data source restrictions

FROM NAMED STREAM WINDOW

FROM

FILTER
OPTIONAL
AND
UNION

 Triple pattern

Dataset

TRUE - FALSE

http://streamreasoning.org/events/sr4ld2014
SPARQLStream Language

- **NamedStream**  \(\rightarrow\) `FROM ['NAMED'] 'STREAM' StreamIRI ['Window']`
- **Window**  \(\rightarrow\) `NOW-' Integer TimeUnit [UpperBound] [Slide]`
- **UpperBound**  \(\rightarrow\) `TO NOW-' Integer TimeUnit`
- **Slide**  \(\rightarrow\) `SLIDE' Integer TimeUnit`
- **TimeUnit**  \(\rightarrow\) `MS' | 'S' | 'MINUTES'| 'HOURS' | 'DAY'`

```
SELECT ISTREAM room
FROM NAMED STREAM <http://www.streamreasoning.org/streams/socialsensor.srdf> [NOW-10 S]
WHERE {
...
```

- **Select**  \(\rightarrow\) `SELECT' [Xstream] [Distinct | Reduced] ...`
- **Xstream**  \(\rightarrow\) `RSTREAM' | 'ISTREAM' | 'DSTREAM'`
SPARQLStream: examples

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
Morph-streams: Overview

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

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

Morph-streams processing SPARQL\textsubscript{Stream} queries

http://streamreasoning.org/events/sr4ld2014

https://github.com/jpcik/morph-streams
## Underlying Query Processors

<table>
<thead>
<tr>
<th>Esper</th>
<th>SNEE</th>
</tr>
</thead>
<tbody>
<tr>
<td>• CEP/DSMS</td>
<td>• DSMS/Sensor Network Query Evaluator</td>
</tr>
<tr>
<td>• EPL language</td>
<td>• Compile queries to sensor code</td>
</tr>
</tbody>
</table>

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

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

<table>
<thead>
<tr>
<th>GSN</th>
<th>Cosm/Xively</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Sensor middleware</td>
<td>• Sensor middleware</td>
</tr>
<tr>
<td>• REST API</td>
<td>• Open platform</td>
</tr>
</tbody>
</table>

```
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
```
Underlying Query Processors

**SPARQLStream**

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

**R2RML**

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

**SNEE (DSMS)**

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

**Esper (CEP)**

```
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
```

**GSN (middlwr)**

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

**Cosm Xively**

```
http://montblanc.slf.ch:22001/multidata?
```
### 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>✗</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>
Morph-streams: With reasoning!

Rewrite taking into account the ontology TBox

Query rewriting

Ontology TBox

Query Processing

Algebra expression

[tuples]

Data translation

[triples/bindings]

SPARQL-Stream

R2RML Mappings

Client

SNEE

Esper

GSN

Cosm

Other

Morph-streams processing SPARQL_stream queries

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

http://streamreasoning.org/events/sr4ld2014
Rewriting the SPARQLStream queries:

SELECT ?x
FROM NAMED STREAM <http://linkeddata.es/s/obs.srdf> [NOW - 5 MINUTES]
WHERE {
  ?x ssn:observedBy ?y
}

SELECT ?x
FROM NAMED STREAM <http://linkeddata.es/s/obs.srdf> [NOW - 5 MINUTES]
WHERE {
  {?x ssn:observedBy ?y}
  UNION
  {?x a ssn:Observation}
  UNION
  {?x a aws:TemperatureObservation}
  UNION
  {?x a aws:HumidityObservation}
}
Morph-streams: With reasoning!

Rewrite only happens once

Then continuous query is registered

SPARQL-Stream

Ontology
TBox

Query rewriting
+ translation

R2RML
Mappings

Data translation

Query Processing

Algebra expression

[triples/bindings]

[tuples]

pull/push

SNEE

Esper

GSN

Cosm

Other

http://streamreasoning.org/events/sr4ld2014

https://github.com/jpcik/morph-streams
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
- https://github.com/jpcik/morph-web
- 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 ...
val 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
case class Query

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)
```

**For Java users: Exactly the same interface!**
SPARQLStream from command line


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

Disclaimer: Simplistic, not implementing all of the SPARQL protocol
Sample result

```json
{
    "head": {
        "vars": [ "timeto" , "obs" ]
    },
    "results": {
        "bindings": [
            {
                "timeto": { "datatype": "http://www.w3.org/2001/XMLSchema#string" , "type": "typed-literal" , "value": "0" } ,
            }
        ]
    }
}
```
- Virtual RDF views over data streams
- Ontology-based access to data streams
  - Examples
  - Architecture
  - Underlying query processors
- SPARQLStream language
- Query rewriting
  - R2RML mappings
- Resources
Resources

- Morph-Streams
  - [https://github.com/jpcik/morph-streams](https://github.com/jpcik/morph-streams)

- See demos

- Read out more

- Contact point
  - jean-paul.calbimonte@epfl.ch
  - ocorcho@fi.upm.es
Morph-streams: SPARQLStream OBDA in action

Jean-Paul Calbimonte
Jean-paul.calbimonte@epfl.ch
Lsir.epfl.ch
RDF Streams

For streams?

$$\langle s, p, o \rangle, \tau$$

$$(\langle s, p, o \rangle, 34532)$$

timestamped triples

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