Stream Reasoning For Linked Data
M. Balduini, J-P Calbimonte, O. Corcho, D. Dell'Aglio, E. Della Valle, and J.Z. Pan
http://streamreasoning.org/sr4ld2013

RDF stream processing models
Daniele Dell’Aglio, daniele.dellaglio@polimi.it
Jean-Paul Cabilmonte, jp.calbimonte@upm.es
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/
Outline

- Continuous RDF model extensions
  - RDF Streams, timestamps

- Continuous extensions of SPARQL
  - Continuous evaluation
  - Additional operators

- Overview of existing systems
  - Implemented operators
  - Different evaluation approaches
Continuous extensions of RDF

- As you know, “RDF is a standard model for data interchange on the Web” (http://www.w3.org/RDF/)

  \(<\text{sub}_1 \ \text{pred}_1 \ \text{obj}_1>\)

  \(<\text{sub}_2 \ \text{pred}_2 \ \text{obj}_2>\)

- We want to extend RDF to model data streams
- A data stream is an (infinite) ordered sequence of data items
- A data item is a self-consumable informative unit
With **data item** we can refer to:

1. A **triple**

   
   
   <:alice :isWith :bob>

2. A **graph**

   
   
   <:alice :posts :p>  
   <:p :who :bob>  
   <:p :where :redRoom>
Data items and time

- Do we need to associate the time to data items?
  - It depends on what we want to achieve (see next!)

- If yes, how to take into account the time?
  - Time should not (but could) be part of the schema
  - Time should not be accessible through the query language
  - Time as object would require a lot of reification

- How to extend the RDF model to take into account the time?
• A timestamp is a temporal identifier associated to a data item
• The **application time** is a set of one or more timestamps associated to the data item
• Two data items can have the same application time
  • Contemporaneity
• Who does assign the application time to an event?
  • The one that generates the data stream!
A RDF stream without timestamp is an ordered sequence of data items

The order can be exploited to perform queries
- Does Alice meet Bob before Carl?
- Who does Carl meet first?
Application time: one timestamp

- One timestamp: the time on which the data item occurs
- We can start to compose queries taking into account the time
  - How many people has Alice met in the last 5m?
  - Does Diana meet Bob and then Carl within 5m?
Application time: two timestamps

- Two timestamps: the time range on which the data item is valid (from, to]
- It is possible to write even more complex constraints:
  - Which are the meetings the last less than 5m?
  - Which are the meetings with conflicts?
## Classification of existing systems

<table>
<thead>
<tr>
<th></th>
<th>Triple</th>
<th>Graph</th>
</tr>
</thead>
<tbody>
<tr>
<td>No timestamp</td>
<td>Instans</td>
<td></td>
</tr>
<tr>
<td>One timestamp</td>
<td>C-SPARQL</td>
<td>SLD</td>
</tr>
<tr>
<td></td>
<td>CQELS</td>
<td></td>
</tr>
<tr>
<td></td>
<td>SPARQLstream</td>
<td></td>
</tr>
<tr>
<td>Two timestamps</td>
<td>EP-SPARQL/Etalis</td>
<td></td>
</tr>
</tbody>
</table>
Our assumptions

- In the following we will consider the following setting
  - A RDF triple is an event
  - Application time: single timestamp
  - System time = application time

\[
\langle :alice :isWith :bob \rangle : [1] \\
\langle :alice :isWith :carl \rangle : [3] \\
\langle :bob :isWith :diana \rangle : [6] \\
\ldots
\]
Let’s process the RDF streams!

- DSMS and CEP worlds suggest different techniques and approaches to process data streams
- We focus on the CQL/STREAM model
Stream processors can elaborate data streams exploiting the timestamps associated to the events.

When a system receives an event, it could have the need of associating a timestamp
- This is the **system time**

The system time is an **internal** value, it does not exit from the system!

The system time must be **unique**

Can application and system time coincide?
- It depends
- Approximation
**RDF stream**

- An RDF stream is an infinite sequence of timestamped events (triples or graphs)

  \[ \ldots \]

  \[ <\text{event}_i, t_i > \]

  \[ <\text{event}_{i+1}, t_{i+1} > \]

  \[ <\text{event}_{i+2}, t_{i+2} > \]

  \[ \ldots \]

- The (application) timestamps must be non-decreasing

  \[ t_i \leq t_{i+1} \]
CQL model

Streams

Relations

Stream

Relation $R(t)$
Mapping: $T \rightarrow R$

infinite unbounded bag

$\ldots$

$\langle S, T \rangle$

$\ldots$

finite bag

$\langle S_1 \rangle$

$\langle S_2 \rangle$

$\langle S_3 \rangle$

stream-to-relation

relation-to-relation

relation-to-stream
Querying RDF data streams

- CQL model

**Abstract** query processing model
Who are both alice and carl meeting?
R2R operators

- SPARQL operators
  - Graph pattern matching
  - JOIN
  - OPTIONAL JOIN
  - SELECTION
  - UNION
SPARQL: a quick recap

http://streamreasoning.org/sr4ld2013
Case 1: the output is a set of timestamped mappings

```
SELECT ?a ?b ...
FROM ....
WHERE ....
```

```
CONSTRUCT {?a :prop ?b }
FROM ....
WHERE ....
```

```
a→ ... ?b→... [t→1]
a→ ... ?b→...
```

```
a→ ... ?b→... [t→3]
a→ ... ?b→...
```

```
a→ ... ?b→... [t→5]
a→ ... ?b→...
```

```
a→ ... ?b→... [t→7]
a→ ... ?b→...
```

```
<... :prop ... > [t→1]
<... :prop ... >
```

```
<... :prop ... > [t→3]
<... :prop ... >
```

```
<... :prop ... > [t→5]
<... :prop ... >
```

```
<... :prop ... > [t→7]
<... :prop ... >
```

Output: relation
Output: stream

- Case 2: the output is a stream

- R2S operators:
  - CONSTRUCT RSTREAM {?a :prop ?b }
  - FROM ....
  - WHERE ....

- R2S operators:
  - ISTREAM: stream out data in the last step that wasn’t on the previous step
  - DSTREAM: stream out data in the previous step that isn’t in the last step
  - RSTREAM: stream out all data in the last step
Other operators

- **Sequence operators and CEP world**

- **SEQ**: joins $e_{ti,tf}$ and $e'_{ti',tf'}$ if $e'$ occurs after $e$
- **EQUALS**: joins $e_{ti,tf}$ and $e'_{ti',tf'}$ if they occur simultaneously
- **OPTIONALSEQ, OPTIONALEQUALS**: Optional join variants
Existing RSP systems

- C-SPARQL: RDF Store + Stream processor
  - Combined architecture

- CQELS: Implemented from scratch. Focus on performance
  - Native + adaptive joins for static-data and streaming data

Disclaimer: oversimplified descriptions
Existing RSP systems

- **EP-SPARQL**: Complex-event detection
  - SEQ, EQUALS operators

- **SPARQLStream**: Ontology-based stream query answering
  - Virtual RDF views, using R2RML mappings
  - SPARQL stream queries over the original data streams.

- **Instans**: RETE-based evaluation

**Disclaimer**: oversimplified descriptions

More details later on!
Query languages syntax

SELECT ?sensor
FROM NAMED STREAM <http://www.cwi.nl/SRBench/observations> [NOW-3 HOURS SLIDE 10 MINUTES]
WHERE {
  ?observation om-owl:procedure ?sensor ;
  om-owl:observedProperty weather:WindSpeed ;
  om-owl:result [ om-owl:floatValue ?value ] .}
GROUP BY ?sensor HAVING ( AVG(?value) >= "74"^^xsd:float )

SELECT ?sensor
FROM STREAM <http://www.cwi.nl/SRBench/observations> [RANGE 1h STEP 10m]
WHERE {
  ?observation om-owl:procedure ?sensor ;
  om-owl:observedProperty weather:WindSpeed ;
  om-owl:result [ om-owl:floatValue ?value ] .}
GROUP BY ?sensor HAVING ( AVG(?value) >= "74"^^xsd:float )

SELECT ?sensor
WHERE {
  STREAM <http://www.cwi.nl/SRBench/observations> [RANGE 10800s SLIDE 600s] {
    ?observation om-owl:procedure ?sensor ;
    om-owl:observedProperty weather:WindSpeed ;
    om-owl:result [ om-owl:floatValue ?value ] .}
GROUP BY ?sensor HAVING ( AVG(?value) >= "74"^^xsd:float )

http://streamreasoning.org/sr4ld2013
## Classification of existing systems

<table>
<thead>
<tr>
<th>Model</th>
<th>Continuous execution</th>
<th>Union, Join, Optional, Filter</th>
<th>Aggregates</th>
<th>Time window</th>
<th>Triple window</th>
<th>R2S operator</th>
<th>Sequence, Co-occurrence</th>
<th>Time function</th>
</tr>
</thead>
<tbody>
<tr>
<td>TA-SPARQL</td>
<td></td>
<td></td>
<td>Limited</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>tSPARQL</td>
<td>TA-RDF</td>
<td>X</td>
<td>✔</td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td></td>
<td>tRDF</td>
<td>X</td>
<td>✔</td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Streaming SPARQL</td>
<td>RDF Stream</td>
<td>✔</td>
<td>✔</td>
<td>X</td>
<td>✔</td>
<td>✔</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C-SPARQL</td>
<td>RDF Stream</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>X</td>
<td>✔</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CQELS</td>
<td>RDF Stream</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SPARQLStream</td>
<td>(Virtual) RDF Stream</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>X</td>
<td>✔</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EP-SPARQL</td>
<td>RDF Stream</td>
<td>✔</td>
<td>✔</td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>✔</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Instans</td>
<td>RDF</td>
<td>✔</td>
<td>✔</td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>

Disclaimer: other features may be missing
Can we compare these RSPs?

Do RSPs behave the same?

Do we get the same results from RSPs?

Check operational semantics
Operational Semantics

Where are both alice and bob in the last 5s?

System 1:
:hall [5]
:kitchen [10]

System 2:
:hall [3]
:kitchen [10]

Both correct?

Find out more later this week on the ISWC Evaluation Track!
Thursday at noon!
SECRET Model: understand operational semantics

$t_0$: When does the windowing start? (internal window param)

**WINDOW CONTENT**: Which stream elements are in the window?

**REPORT**: When is the window content made available to the R2R operator?
*Non-empty content, Content-change, Window-close, Periodic*

**TICK**: When the data stream are inserted in the window?
*Triple-based vs graph-based*

**W(ω,β)**

R2R operator
### SECRET model classification

<table>
<thead>
<tr>
<th></th>
<th>CQELS</th>
<th>C-SPARQL</th>
<th>SPARQL_{stream}</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Report</strong></td>
<td>Content-change</td>
<td>Window-close Non-empty content</td>
<td>Window-close Non-empty content</td>
</tr>
<tr>
<td><strong>Tick</strong></td>
<td>Tuple-driven</td>
<td>Tuple-driven</td>
<td>Tuple-driven</td>
</tr>
<tr>
<td><strong>Empty relation notification</strong></td>
<td>No</td>
<td>Yes</td>
<td>No</td>
</tr>
</tbody>
</table>

- Characterize non-window-based RSPs?
- Multiple streams?, reasoning?, linking with static data?
Benchmarks and comparing

http://www.w3.org/wiki/SRBench

C-SPARQL

SPARQLStream

CQELS

Not exhaustive!
## Functional Evaluation

<table>
<thead>
<tr>
<th>System</th>
<th>Q1</th>
<th>Q2</th>
<th>Q3</th>
<th>Q4</th>
<th>Q5</th>
<th>Q6</th>
<th>Q7</th>
<th>Q8</th>
<th>Q9</th>
<th>Q10</th>
<th>Q11</th>
<th>Q12</th>
<th>Q13</th>
<th>Q14</th>
<th>Q15</th>
<th>Q16</th>
<th>Q17</th>
</tr>
</thead>
<tbody>
<tr>
<td>SPARQLStream</td>
<td>PP</td>
<td>A</td>
<td>G</td>
<td>G</td>
<td>G</td>
<td>G</td>
<td>G</td>
<td>IF</td>
<td>SD</td>
<td>SD</td>
<td>SD</td>
<td>SD</td>
<td>SD</td>
<td>SD</td>
<td>SD</td>
<td>SD</td>
<td>SD</td>
</tr>
<tr>
<td>CQELS</td>
<td>PP</td>
<td>A</td>
<td></td>
<td>D/N</td>
<td>IF</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>PP</td>
<td>PP</td>
<td>PP</td>
<td>PP</td>
<td>PP</td>
<td>PP</td>
<td>PP</td>
<td></td>
</tr>
<tr>
<td>C-SPARQL</td>
<td>PP</td>
<td>A</td>
<td></td>
<td>D</td>
<td>IF</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>PP</td>
<td>PP</td>
<td>PP</td>
<td>PP</td>
<td>PP</td>
<td>PP</td>
<td>PP</td>
<td>PP</td>
</tr>
</tbody>
</table>

- **Ask**
- **D**-stream
- **G**roup by and aggregations
- **IF** expression
- **N**egation
- **P**roperty Path
- **S**tatic **D**ataset
A lot to do…

- Agree on an RDF model?
  - Metamodel?
  - Timestamps in graphs?
  - Timestamp intervals
  - Compatibility with normal (static) RDF

- Additional operators for SPARQL?
  - Windows (not only time based?)
  - CEP operators
  - Semantics

- Go Web
  - Volatile URIs
  - Serialization: terse, compact
  - Protocols: HTTP, Websockets?
References

RDF stream processing models

Daniele Dell’Aglio, daniele.dellaglio@polimi.it
Jean-Paul Cabilmonte, jp.calbimonte@upm.es