IMaRS: Incremental Materialization for RDF Streams

Daniele Dell'Aglio – daniele.dellaglio@polimi.it
Emanuele Della Valle – emanuele.dellavalle@polimi.it
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

To view a copy of this license, visit [http://creativecommons.org/licenses/by/3.0/](http://creativecommons.org/licenses/by/3.0/)
Running Example – Data Model

Streaming information

Observation ⊔

Post ⊔

subClassOf

subPropOf

discusses

who

observes

posts

where

Background information

Room ⊔

Sensor ⊔

Person ⊔

isConnectedTo

subClassOf

isWith

http://streamreasoning.org/events/sr4ld2014
Running Example – Data Model

Observation

Sensor

Person

Post

Room

Instances

isWith

isIn

subClassOf

subPropOf

disscuses

posts

where

observes

streaming information

background information

instances
What

- Add reasoning in window-based RSPs
- Naïve solution: materialize everything, every time
- But windows slide:
  - The materialisation is executed every time the window updates
  - Only part of data changes at each window update
  - Materialisation is (usually) an expensive task
Naïve solution: an example

TBOX

\[
\text{dom}(\text{\textbullet}) \sqsubseteq \{\text{\textbullet}\}
\]

\[
\text{rng}(\text{\textbullet}) \sqsubseteq \{\text{\textbullet}\}
\]
**Naïve solution: an example**

**TBOX**

\[
\text{dom}(\ ) \subseteq \begin{array}{c}
\text{\includegraphics[scale=0.5]{people.png}}
\end{array}
\]

\[
\text{rng}(\ ) \subseteq \begin{array}{c}
\text{\includegraphics[scale=0.5]{speech_bubble.png}}
\end{array}
\]
Naïve solution: an example

TBOX

\[ \text{dom(} \subseteq \text{persons} \]

\[ \text{rng(} \subseteq \text{messages} \]

http://streamreasoning.org/events/sr4ld2014
Naïve solution: an example

TBOX

$$\text{dom}(\text{Alice}) \subseteq \text{people}$$

$$\text{rng}(\text{Message}) \subseteq \text{Messages}$$
Incremental maintenance

- Adopt an incremental approach
- Compute only the differences that should be removed and added from the materialization

```
\[
\Delta \uparrow - \quad \text{Inferred statements} \quad \Delta \uparrow +
\]
```

Explicit statements

To be removed

To be added
Incremental maintenance: an example

TBOX

\[
\text{dom}(\cdot) \subseteq \{\text{people}\} \\
\text{rng}(\cdot) \subseteq \{\text{documents}\}
\]
Incremental maintenance: an example

TBOX

\[ \text{dom}(\text{talk}) \subseteq \{\text{people}\} \]
\[ \text{rng}(\text{message}) \subseteq \{\text{messages}\} \]
Incremental maintenance: an example

TBOX

\[
\text{dom}(\text{\textbullet}) \subseteq \text{\text{\textbullet}}
\]

\[
\text{rng}(\text{\textbullet}) \subseteq \text{\text{\textbullet}}
\]
Incremental maintenance: an example

TBOX

\[ \text{dom}() \subseteq \text{people} \]
\[ \text{rng}() \subseteq \text{speech} \]

To be deleted

To be added

1 \(\in\) people

2 \(\in\) people

3 \(\in\) people
Incremental maintenance: an example

TBOX

$$\text{dom}() \subseteq \text{people}$$

$$\text{rng}() \subseteq \text{messages}$$

To be renewed

To be deleted

To be added
Which technique?

- The common problem in designing incremental maintenance techniques is in the management of **deletions**
- In general it is not possible to foresee the statement deletions
  - DRed works with random insertions and deletions
- In our setting it is possible
  - The window operator allow us to determine when statements will be removed
Variation of DRed for RDF streams

It pushes the maintenance algorithm in the window operator

An IMaRS window is a sliding window with four parameters:
- $\omega$: the size of the window
- $\beta$: the slide of the window
- $T$: the TBox that describes the data model
- $M$: the maintenance program

One of the central IMaRS concepts is the expiration time
### Expiration time

- Every time a statement is added to the window, it is annotated with an **expiration time**
- The expiration time indicates when the statement should be removed from the materialization

<table>
<thead>
<tr>
<th>TBOX</th>
<th>Window (2,1)</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>dom()</code> ∈</td>
<td>Current time</td>
</tr>
<tr>
<td><code>rng()</code> ∈</td>
<td></td>
</tr>
</tbody>
</table>

- **Current time:** 10

---

![Diagram showing the expiration time concept](http://streamreasoning.org/events/sr4ld2014)
Expiration time

- Every time a statement is added to the window, it is annotated with an **expiration time**
- The expiration time indicates when the statement should be removed from the materialization

```
TBOX
\text{dom}(S) \subseteq \text{dom}(T)
\text{rng}(S) \subseteq \text{rng}(T)
```

**Current time**

```
11
```

Window (2,1)

The statement will exit at 13
Expiration time

- Every time a statement is added to the window, it is annotated with an **expiration time**
- The expiration time indicates when the statement should be removed from the materialization

![Diagram showing TBOX, Window (2,1), and current time 11. The statement will exit at 13, and the inferred statements will exit at 13.](http://streamreasoning.org/events/sr4ld2014)
Expiration time

- Every time a statement is added to the window, it is annotated with an **expiration time**
- The expiration time indicates when the statement should be removed from the materialization

### TBOX

- dom(\(\geq\)) ⊑ \(\subseteq\)
- rng(\(\geq\)) ⊑ \(\subseteq\)

### Current time

- 12

### Window (2,1)

- 1
  - ∈
  - \(\subseteq\)
- 13
  - 13

---

http://streamreasoning.org/events/sr4ld2014
Expiration time

- Every time a statement is added to the window, it is annotated with an **expiration time**
- The expiration time indicates when the statement should be removed from the materialization

<table>
<thead>
<tr>
<th>TBOX</th>
<th>Window (2,1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>dom( ) ⊑ rng( )</td>
<td>13</td>
</tr>
<tr>
<td>13</td>
<td></td>
</tr>
</tbody>
</table>

Current time

13

The inferred statements expire
Expiration time

- Every time a statement is added to the window, it is annotated with an **expiration time**
- The expiration time indicates when the statement should be removed from the materialization

The inferred statements expire

<table>
<thead>
<tr>
<th>TBOX</th>
<th>Window (2,1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \text{dom}(\text{ }) \subseteq \text{ })</td>
<td></td>
</tr>
<tr>
<td>( \text{rng}(\text{ }) \subseteq \text{ })</td>
<td></td>
</tr>
</tbody>
</table>
Expiration time generation

- At each window update
  - Statement deletion
  - Expiration time update
  - Expiration time assignment
  - To be removed
  - To be renewed
  - Inferred statements
  - Explicit statements
  - To be added

- The computation is done through the execution of a maintenance program

Expiration time assignment
IMaRS at a glance

DRed
- Delete
- Rederive
- Insert

IMaRS
- Lookup
- Insert + Renew
IMaRS maintenance program

- The maintenance program computes the delta sets $\Delta^\uparrow$ and $\Delta^\uparrow^+$
  - It is a **logic program**

- The program is executed every time the content changes
  - In our context, the program is executed every time the window slides

- The program is composed by **maintenance rules**
  - A maintenance rule adds a statement in a set *(context)* if the preconditions are satisfied
The maintenance program uses four contexts to build the delta sets $\Delta^{\uparrow-}$ and $\Delta^{\uparrow+}$

- $Mat$: the current materialization
- $Ins$: the input that enters the stream and the related inferred statements

Additionally, two support sets are used:

- $New$: statements to be added to the materialization
- $Ren$: renewed statements

The new materialization is computed as

$$Mat \cup \Delta^{\uparrow+} \setminus \Delta^{\uparrow-}$$
Two examples of maintenance rules:

\[ \Delta^-(?s,?p,?o)[e] \leftarrow \text{Mat}(?s,?p,?o)[e] . \]
\[ e < \text{now} \]

A triple is removed by the materialization when its expiration time expires

\[ \text{Ins}(?x,?p,?z)[e] \leftarrow \text{New}(?x,?p,?y)[e1]. \]
\[ \text{Ins}(?x,?p,?y)[e2] . \text{Ins}(?p,\text{isA,TransitiveProperty})[e3]. \]
\[ e = \min\{e1,e2,e3\} \]

When a triple \(<s,p,o>\) enters the window, \(p\) is transitive and there is a triple \(<o,p,k>\) in the \(\text{Ins}\) context, then the triple \(<s,p,k>\) is a candidate for the addition in the materialization
IMaRS maintenance program

- The maintenance program is composed by two sets of maintenance rules:
  - One set of fixed maintenance rules
  - One dependent on the ontological language

- The ontological language should be expressed as a set of inference rules, e.g.

- It does not depend on the TBox!
Generation of the maintenance program

Rewriting functions

Maintenance program generator

Ontological language

Maintenance program

TBox

IMaRS Window \((\omega, \beta)\)
Example: DRed

TBOX

\[ \text{tr}(\text{DRed}) \]

Window (3,1)

Current time

11

Insert DRed
Example: DRed

TBOX

\[ \text{tr}(\text{rng}) \]

Current time

12

Window (3,1)

\[
\begin{array}{ccc}
3 & \rightarrow & 1 \\
2 & \rightarrow & 1 \\
3 & \rightarrow & 2 \\
4 & \rightarrow & 1 \\
\end{array}
\]
Example: DRed

TBOX

\[ \text{tr}(\text{DRed}) \]

Current time

13

Window (3,1)

\[
\begin{array}{ccc}
3 & \leftrightarrow & 1 \\
3 & \leftrightarrow & 2 \\
3 & \leftrightarrow & 4 \\
3 & \leftrightarrow & 1 \\
3 & \leftrightarrow & 4 \\
3 & \leftrightarrow & 1 \\
\end{array}
\]
Example: DRed

```
1
\[ DRed \]
\[ \triangleright \]
\[ \triangleright \]
\[ \triangleright \]
\[ \triangleright \]
\[ \triangleright \]
\[ \triangleright \]
\[ \triangleright \]
\[ \triangleright \]
```

```
<table>
<thead>
<tr>
<th>TBOX</th>
<th>Window (3,1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>tr([[])</td>
<td></td>
</tr>
</tbody>
</table>

```

```
<table>
<thead>
<tr>
<th>Current time</th>
<th>14</th>
</tr>
</thead>
</table>

```

```
<table>
<thead>
<tr>
<th>t</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>3</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>1</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>1</td>
<td>3</td>
<td>4</td>
</tr>
</tbody>
</table>

http://streamreasoning.org/events/sr4ld2014
Example: DRed

TBOX
\( \text{tr}(\square) \)

Current time
14

Window (3,1)

10 11 12 13

14

Delete
Example: DRed

TBOX

\( \text{tr}((\square \rightarrow \Box)) \)

Current time

14

Window (3,1)

1

3

Delete

Insert

DRed
Example: DRed

TBOX
\[ \text{tr}(\sqsubseteq) \]

Current time
14

Window (3,1)

Delete
Rederive
Example: IMaRS

TBOX

\[ \text{tr}(\text{\textbullet}) \]

Current time

11

Window (3,1)

1 2

1 2 14
Example: IMaRS

TBOX
\[\text{tr}(\text{rn})\]

Current time
12

Window (3,1)

http://streamreasoning.org/events/sr4ld2014
Example: IMaRS

TBOX

\[ \text{tr}(\text{rng}) \]

Current time

13

Window (3,1)

<table>
<thead>
<tr>
<th></th>
<th>3</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td></td>
<td>14</td>
</tr>
<tr>
<td>1</td>
<td></td>
<td>14</td>
</tr>
<tr>
<td>3</td>
<td></td>
<td>15</td>
</tr>
<tr>
<td>4</td>
<td></td>
<td>15</td>
</tr>
</tbody>
</table>

\[ \text{STBOX} \]

\[ \text{Window (3,1)} \]
Example: IMaRS

TBOX

\[ \text{tr}(\text{rng}) \]

Current time

13

Window (3,1)

\[ 3 \quad \text{↓} \quad 1 \quad 14 \]

\[ 2 \quad \text{↓} \quad 1 \quad 15 \]

\[ 3 \quad \text{↓} \quad 2 \quad 15 \]

\[ 4 \quad \text{↓} \quad 1 \quad 15 \]

\[ 3 \quad \text{↓} \]

\[ 4 \quad \text{↓} \]

\[ 1 \quad \text{↓} \]

\[ 2 \quad \text{↓} \]

\[ 3 \quad \text{↓} \]

\[ 4 \quad \text{↓} \]
Example: IMaRS

TBOX

\[ \text{tr}(\text{rn}) \]

Current time

13

Window (3,1)

\[
\begin{array}{c}
3 \\
1 \\
\end{array}
\]

\[
\begin{array}{c}
2 \\
1 \\
3 \\
4 \\
\end{array}
\]

\[
\begin{array}{c}
3 \\
2 \\
4 \\
\end{array}
\]

\[
\begin{array}{c}
3 \\
\end{array}
\]

\[
\begin{array}{c}
4 \\
\end{array}
\]

9 10 11 12
Example: IMaRS
IMaRS: Incremental Materialization for RDF Streams

Daniele Dell'Aglio – daniele.dellaglio@polimi.it
Emanuele Della Valle – emanuele.dellavalle@polimi.it