RDF Mapping Language (RML)
A Generic Language for Integrated RDF Mappings of Heterogeneous Data

Anastasia Dimou, Miel Vander Sande, Pieter Colpaert, Ruben Verborgh, Erik Mannens and Rik Van de Walle
Ghent University – iMinds – Multimedia Lab
http://semweb.mmlab.be/rml

LDOW14, WWW14
Seoul, Korea, 8th April 2014
The five stars of the Linked Open Data scheme are approached as a set of consecutive steps.
... and are applied to a single input source every time
Limitations of current solutions

The semantic representation of each mapped resource is

Individually defined
disregarding its possible prior definitions
and its links to other resources

Manually aligned
by reconstructing the same URIs

Not linked to other resources
links are defined after the data are mapped and published
Need for a well-considered policy regarding mapping and primary interlinking of data in the context of a certain knowledge domain
No mapping formalization exists that defines how to map heterogeneous sources into RDF using integrated and interoperable mappings.
Relational Database to RDF (R2RML W3C)

Data OWNER / PUBLISHER

defines

DB

R2RML mappings

R2RML processor

RDF
Mapping heterogeneous resources to RDF

Data OWNER / PUBLISHER defines

R2RML mappings → R2RML processor

DB → RDF

CSV → RDF
Mapping heterogeneous resources to RDF

Data OWNER / PUBLISHER defines R2RML mappings to R2RML processor:
- DB -> RDF
- CSV -> RDF
- XML -> RDF

R2RML mappings
Current limitation: mapping data on a per-source & per-format basis

Data OWNER / PUBLISHER defines R2RML mappings → R2RML processor → DB, CSV, XML, JSON → RDF
Further limitation: lack of **uniform and interoperable** solutions

The mappings are **tied to the implementations not interoperable across different implementations**

No uniform way to describe mappings of heterogeneous resources that describe complementarily the same domain

Mapping definitions are **not reused** for data in the **same or different formats**
Uniform way for integrated mapping of heterogeneous sources

Data OWNER / PUBLISHER

defines

Mappings definitions?

DB

CSV

XML

JSON

processor

RDF
R2RML mapping definition

- Triples Map
  - Logical Table
    - Table Name
      - 1 Subject Map
  - Subject Map
    - 0 or more Predicate-Object Maps
      - Predicate Map
        - Object Map
          - Predicate-Object Map
          - Predicate-Object Map
          - Predicate-Object Map
R2RML mapping definition

- Triples Map
  - Logical Table
    - Table Name
  - Subject Map
    - Predicate-Object Map
    - Predicate Map
      - Object Map
    - Predicate-Object Map
    - Predicate Map
    - Object Map
From R2RML to a generic mapping language

RDF Term:
- a URI,
- a literal,
- a blank node
R2RML Mapping

<table>
<thead>
<tr>
<th>Suitcase</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>567</td>
<td>Samsonite DeLux 45</td>
</tr>
</tbody>
</table>

ex:567 a schema:Product;
  rdfs:label “Samsonite DeLux 45”.

<#ProductMapping>
  rr:logicalTable [
    rr:tableName “Suitcase” ];

  rr:subjectMap [
    rr:template "http://ex.com/{Suitcase}";
    rr:class ex:Person ];

  rr:predicateObjectMap [
    rr:predicate rdfs:label;
    rr:objectMap “Name” ].
from **R2RML** to a generic mapping language

<table>
<thead>
<tr>
<th>R2RML</th>
<th>Generic mapping language</th>
</tr>
</thead>
<tbody>
<tr>
<td>Logical Table</td>
<td>Logical Source (CSV, XML, JSON)</td>
</tr>
<tr>
<td>Table Name</td>
<td>Source name / URI</td>
</tr>
<tr>
<td>Column</td>
<td></td>
</tr>
<tr>
<td>per row iteration</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
References to values of heterogeneous resources

<PendingOrders>...
  <Order id="398">
    <Product>
      <Id>AE5982</Id>
      <Name>Samsonite DeLux 45</Name>
    </Product>
  </Order>...
<PendingOrders>

{ ...,  
  "ProductInStock" :  
  { "ID": "567",  
    "Name": "Samsonite DeLux 45",  
    "type": "suitcase",  
  }, ...  }

**XPath for XML**

Reference:  
“Order@Id”  
Iterator:  
“/PendingOrders /Order”

**JSONPath for JSON**

Reference:  
“$. ProductInStock.ID”  
Iterator:  
“$.ProductInStock”
from R2RML to a generic mapping language

<table>
<thead>
<tr>
<th>R2RML</th>
<th>R2RML</th>
</tr>
</thead>
<tbody>
<tr>
<td>Logical Table</td>
<td>Logical Source (CSV, XML, JSON)</td>
</tr>
<tr>
<td>Table Name</td>
<td>Source name / URI</td>
</tr>
<tr>
<td>Column</td>
<td>Reference (defined Reference Formulation)</td>
</tr>
<tr>
<td>per row iteration</td>
<td>defined Ion Iterator</td>
</tr>
</tbody>
</table>
<PendingOrders>...
  <Order id="398">
    <Product>
      <Id>AE5982</Id>
      <Name>Samsonite DeLux 45</Name>
    </Product>
  </Order> ...
</PendingOrders>

ex:AE5982 a schema:Product ;
  rdfs:label "Samsonite DeLux 45".

<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html>
<html lang="en">
  <head>
    <title>Mapping XML files</title>
  </head>
  <body>
    <h1>Mapping XML files</h1>
    <pre>&lt;#OrdersMapping&gt;
      rml:logicalSource [rml:source "orders.xml";]
      rml:referenceFormulation ql:XPath;
      rml:iterator "/PendingOrders/Order/Product" ];
      rr:subjectMap [
        rr:template http://ex.com/{Id};
        rr:class schema:Product ];
      rr:predicateObjectMap [
        rr:predicate rdfs:label ;
        rr:object "Product/Name" ].
    &lt;/#OrdersMapping&gt;</pre>
  </body>
</html>
Mapping JSON files

```json
{ ...
  "ProductInStock":
  { "ID": "567",
    "Name": "Samsonite DeLux 45",
    "type": "suitcase"
  }, ...
}

ex:567 a schema:Product ;
  rdfs:label "Samsonite DeLux 45" .

<#ProductInStockMapping>
  rml:iterator ";.ProductInStock"];
```
Robust cross-references

```json
{ ...
  "Performance": {
    "Perf_ID": "567",
    "Location": {
      "lat": "51.043611",
      "long": "3.717222"
    }
  },
  ...
}
```

```xml
<#PerformancesMapping>
  rr:subjectMap [
    rr:template "http://ex.com/{Perf_ID}";]
  rr:predicateObjectMap [
    rr:predicate ex:location;
    rr:objectMap [
      rr:parentTriplesMap <#LocationMapping> ] ].
</#PerformancesMapping>

```xml
<Events> ...
  <Exhibition id="398">
    <Location>
      <lat>51.043611</lat>
      <long>3.717222</long>
    </Location>
  </Exhibition> ...
</Events>
```xml

```xml
<#EventsMapping>
  rr:subjectMap [
    rr:template "http://ex.com/{@id}" ];
  rr:predicateObjectMap [
    rr:predicate ex:location;
    rr:objectMap [
      rr:parentTriplesMap <#LocationMapping> ] ];
</#EventsMapping>
```
Robust cross-references

{ "Performance": {
   "Perf_ID": "567",
   "Location": {
      "lat": "51.043611",
      "long": "3.717222"
   }
}, ...

<Events> ...
<Exhibition id="398">
   <Location>
      <lat>51.076891</lat>
      <long>3.717222</long>
   </Location>
</Exhibition> ...
</Events>

<#LocationMapping>
   rr:subjectMap [ rr:template "http://ex.com/{lat},{long}" ];
   rr:predicateObjectMap [ rr:predicate ex:lat; rr:objectMap [ rml:reference "lat" ] ];

   ex:567 ex:location ex:51.043611, 3.717222
   ex:398 ex:location ex:51.076891, 3.717222

   ex:51.043611, 3.717222
   ex:lat ex:3.717222
   ex:long ex:51.043611.
Primary Interlinking

```json
{ ... "Performance": {
    "Perf_ID": "567",
    "Venue": {
        "Name": "STAM",
        "Venue_ID": "78"
    },
    "Location": {
        "long": "3.717222",
        "lat": "51.043611"
    }
}, ... }

<#PerformancesMapping>
rr:subjectMap [ 
    rr:template "http://ex.com/{Perf_ID}" ];
rr:predicateObjectMap [ 
    rr:predicate ex:venue; 
    rr:objectMap [ 
        rr:parentTriplesMap <#VenueMapping> ] ].
</#PerformancesMapping>

<#VenueMapping>
rr:logicalSource [ 
    rml:source "http://ex.com/performances.json";
    rml:referenceFormulation ql:JSONPath;
    rml:iterator "$.Performance.Venue.[*]" ];
rr:subjectMap [ 
    rr:template "http://ex.com/{Venue_ID}";
    rr:class ex:Venue ].
</#VenueMapping>
```
Primary Interlinking

{ ... "Performance" : 
{ "Perf_ID": "567", 
  "Venue": { 
    "Name": "STAM", 
    "Venue_ID": "78" }, 
  ... }

<Events> ...
  <Exhibition id="398">
    <Venue>STAM</Venue>
  </Exhibition> ... ...
</Events>

<#EventsMapping>
  rr:subjectMap [ 
    rr:template "http://ex.com/{@id}" ];

  rr:predicateObjectMap [ 
    rr:predicate ex:venue; 
    rr:objectMap [ 
      rr:parentTriplesMap <#VenueMapping>; 
      rr:joinCondition [ 
        rr:child ".Performance.Venue.Name"; 

ex:567 ex:venue ex:78. 
ex:398 ex:venue ex:78.
Robust cross-references and primary interlinking

Avoid redefining and replicating URI patterns

Uniquely define the URI patterns that generates a resource and refer to its definition

Modifications to the patterns or data values are propagated to every other reference of the resource

Links between resources in different inputs are defined already on mapping level

New mappings are automatically aligning
Address the mappings definition in a generic way scale over the input data extracts.

Distinct and not interdependent references to the data extracts and the mappings

*Proof:* CSS3 selectors to map HTML documents enrich the aforementioned data with data from [amazon.com](http://amazon.com) and [ebay](http://ebay).
Conclusions: Addressed Limitation

Limitations:
Mapping of data on a per-source and per-format basis
Mapping definitions are tied to the implementation
Lack of Mapping definitions’ reuse

RDF Mapping Language (RML):
Uniform and interoperable mapping definitions
Robust cross-references and interlinking
Scalable mapping language
RDF Mapping Language (RML)

generic language for mapping heterogeneous resources into RDF in an integrate and interoperable fashion

RML: http://semweb.mmlab.be/rml
RML Processor: https://github.com/mmlab/RMLProcessor

Contact us
Anastasia Dimou anastasia.dimou@ugent.be @natadimou
Miel Vander Sande miel.vandersande@ugent.be @Miel_vds