Simplified RDB2RDF Mapping

Claus Stadler, Jörg Unbehauen, Patrick Westphal, Mohamed Ahmed Sherif and Jens Lehmann

presented by Axel-Cyrille Ngonga Ngomo

2015 May 19
Outline

1 Motivation

2 R2RML in a Nutshell

3 SML in a Nutshell

4 SML Step by Step Example

5 Evaluation
Outline

1 Motivation

2 R2RML in a Nutshell

3 SML in a Nutshell

4 SML Step by Step Example

5 Evaluation
Several tools exist that implemented different approaches for mapping relational databases to RDF, of which R2RML became a W3C standard (http://www.w3.org/TR/r2rml/).

**Motivation - RDB2RDF Approaches**

1. map: eventTitle a d2rq: PropertyBridge;
   d2rq: belongsToClassMap map: Conference;
   d2rq: property : eventTitle;
   d2rq: column "Conferences.Name";
   d2rq: datatype xsd: string;

**D2RQ**

1. [MappingDeclaration] @collection [[
2. mappingId Book collection
3. target :BID_{id} a :Book .
4. source SELECT id FROM books
5. ]]

**Ontop**

1. graph <http://localhost/testdata/products#>
   2. subject prd: product_iri(PRODUCT.
   3. PRODUCT_ID)
   4. predicate rdf: type
   5. object prd: Product

**R2RML**

1. <#emps>
   2. rr: logicalTable [  
   3.   rr: tableName "employees"
   4. ];
   5. rr: subjectMap [  
   6.   rr: template "http://ex.org/{id}"
   7.   rr: class foaf: Person
   8. ] .

**Virtuoso RDF views**
From Tables to Triples

All these approaches iterate tables and on every row they first create RDF terms and then arrange them to triples:
Our Approach

- In SQL, there is the well known CREATE VIEW statement to create views from tables and other views.
- Quad stores essentially use a table with four columns to store RDF data.
- Current RDB2RDF approaches are quite different from how views are created in SQL.
- Our approach is to blend the traditional SQL CREATE VIEW statements with SPARQL CONSTRUCT queries:

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX ex: <http://example.org/>
CREATE VIEW emps AS
    CONSTRUCT {
        ?s a foaf:Person
    }
  With
    ?s = uri(ex:, ?id)
  From
    employees
```
Contributions

- Definition of the compact **Sparqlification Mapping Language** (SML) mapping language with equal expressiveness to R2RML
- A unified formal model of RDB2RDF mapping languages.
- User Study which compares SML to R2RML
- Tooling: SML/R2RML Converters and Syntax Highlighters
Outline

1. Motivation

2. R2RML in a Nutshell

3. SML in a Nutshell

4. SML Step by Step Example

5. Evaluation
An **R2RML mapping** is an RDF resource that must be described with the following properties:

- Exactly one `rr:logicalTable`, which refers to the view’s logical table, i.e. an SQL query, SQL table or SQL view.
- Exactly one `rr:subjectMap`, which defines the subject of the triples created from this mapping.
- Zero or more instances of `rr:predicateObjectMap`, that attach a set of predicate/object pairs using `rr:predicateMap` and `rr:objectMap` to the corresponding subject.
- Each of `rr:subjectMap`, `rr:predicateMap` and `rr:objectMap` must be further described to specify what RDF terms to create from every row of the logical table.

Note, that R2RML offers a set of **shortcut properties**, which we do not discuss for brevity.
Generic form of an R2RML mapping without the use of shortcuts:

**R2RML Example:**

```xml
@prefix foaf: <http://xmlns.com/foaf/0.1/> .

<#emps>

  rr:logicalTable [ rr:tableName "employees" ] ;  
  rr:subjectMap [ rr:template "http://example.org/{id}" ];  
  rr:predicateObjectMap [  
    rr:predicateMap [ rr:constant rdf:type ];  
    rr:objectMap [ rr:constant foaf:Person ]  
  ] .
```

Stadler et al. (Univ. Leipzig)

Simplified RDB2RDF Mapping

2015 May 19
Outline

1 Motivation

2 R2RML in a Nutshell

3 SML in a Nutshell

4 SML Step by Step Example

5 Evaluation

Stadler et al.  (Univ. Leipzig)  Simplified RDB2RDF Mapping  2015 May 19  11 / 32
A SML view comprises:

- A name
- A `CONSTRUCT` clause for which quads to create
- A `FROM` clause for the underlying logical table.
- A `WITH` clause that creates RDF terms from the columns of the logical table and assigns them to variables.
- Optionally, a `CONSTRAINT` clause, where URI prefixes of variables can be stated (can be used for pruning joins in SPARQL-to-SQL rewriters).
Example of an SML View

SML Example:

```sml
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX ex: <http://example.org/>
CREATE VIEW emps AS
  CONSTRUCT {
    ?s a foaf:Person
  }
  With
  ?s = uri(ex:, ?id)
From
  employees
```
# Creating RDF Terms in SML and R2RML

<table>
<thead>
<tr>
<th>SML RDF term constructor</th>
<th>R2RML term map</th>
</tr>
</thead>
<tbody>
<tr>
<td>bNode(?COL)</td>
<td>... [ rr:column &quot;COL&quot; ; rr:termType rr:blankNode ]</td>
</tr>
<tr>
<td>bNode(expr)</td>
<td>... [ rr:template &quot;asTemplate(expr)&quot; ; rr:termType rr:blankNode ]</td>
</tr>
<tr>
<td>uri(expr)</td>
<td>... [ rr:(constant</td>
</tr>
<tr>
<td>plainLiteral(?COL)</td>
<td>... [ rr:column &quot;COL&quot; ]</td>
</tr>
<tr>
<td>plainLiteral(expr)</td>
<td>... [ rr:template &quot;asTemplate(expr)&quot; ]</td>
</tr>
<tr>
<td>typedLiteral(?COL, xsd:int)</td>
<td>... [ rr:column &quot;COL&quot; ; rr:datatype xsd:int ]</td>
</tr>
<tr>
<td>typedLiteral(expression, xsd:int)</td>
<td>... [ rr:template &quot;asTemplate(expr)&quot; ; rr:datatype xsd:int ]</td>
</tr>
</tbody>
</table>

*Table*: Transformation of SML term constructors to R2RML term maps
Outline

1 Motivation

2 R2RML in a Nutshell

3 SML in a Nutshell

4 SML Step by Step Example

5 Evaluation
The following slides demonstrate how to map relational data to RDF with the Sparqlification Mapping Language (SML).

Thereby, these prefixes are used:

<table>
<thead>
<tr>
<th>Prefix</th>
<th>IRI</th>
</tr>
</thead>
<tbody>
<tr>
<td>rdfs</td>
<td><a href="http://www.w3.org/2000/01/rdf-schema#">http://www.w3.org/2000/01/rdf-schema#</a></td>
</tr>
<tr>
<td>ogc</td>
<td><a href="http://www.opengis.net/ont/geosparql#">http://www.opengis.net/ont/geosparql#</a></td>
</tr>
<tr>
<td>geom</td>
<td><a href="http://geovocab.org/geometry#">http://geovocab.org/geometry#</a></td>
</tr>
<tr>
<td>lgd</td>
<td><a href="http://linkedgeodata.org/triplify/">http://linkedgeodata.org/triplify/</a></td>
</tr>
<tr>
<td>lgd-geom</td>
<td><a href="http://linkedgeodata.org/geometry/">http://linkedgeodata.org/geometry/</a></td>
</tr>
</tbody>
</table>
SML - Mapping Example: The Goal (1/4)

Input Table

<table>
<thead>
<tr>
<th>nodes</th>
<th>geom</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>POINT(0 0)</td>
</tr>
<tr>
<td>2</td>
<td>POINT(1 1)</td>
</tr>
</tbody>
</table>

How to map tables to RDF?

How to introduce the commonly used distinction in GIS between feature and geometry?

Aimed for RDF Output

@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .

... lgd:node1 geom:geometry lgd-geom:node1 . lgd:node2 geom:geometry lgd-geom:node2 . lgd-geom:node1 ogc:asWKT "POINT(0 0)"^^ogc:wktLiteral . lgd-geom:node2 ogc:asWKT "POINT(1 1)"^^ogc:wktLiteral .
Create View myNodesView As
Construct {
    
    
}  
With 
    
    
From 
    

Aimed for RDF Output

@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
...

lgd:node1 geom:geometry lgd-geom:node1 .
lgd:node2 geom:geometry lgd-geom:node2 .

lgd-geom:node1 ogc:asWKT "POINT(0 0)"^^ogc:wktLiteral .
lgd-geom:node2 ogc:asWKT "POINT(1 1)"^^ogc:wktLiteral .

Input Table

<table>
<thead>
<tr>
<th>id</th>
<th>geom</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>POINT(0 0)</td>
</tr>
<tr>
<td>2</td>
<td>POINT(1 1)</td>
</tr>
</tbody>
</table>
Input Table

<table>
<thead>
<tr>
<th>id</th>
<th>geom</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>POINT(0 0)</td>
</tr>
<tr>
<td>2</td>
<td>POINT(1 1)</td>
</tr>
</tbody>
</table>

Create View myNodesView As
Construct {
  ?n geom:geometry ?g .
  ?g ogc:asWKT ?o
}
With
...
From nodes

Aimed for RDF Output

@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
...

lgd:node1 geom:geometry lgd-geom:node1 .
lgd:node2 geom:geometry lgd-geom:node2 .

lgd-geom:node1 ogc:asWKT "POINT(0 0)"^^ogc:wktLiteral .
lgd-geom:node2 ogc:asWKT "POINT(1 1)"^^ogc:wktLiteral .
### Input Table

<table>
<thead>
<tr>
<th>id</th>
<th>geom</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>POINT(0 0)</td>
</tr>
<tr>
<td>2</td>
<td>POINT(1 1)</td>
</tr>
</tbody>
</table>

Create View myNodesView As

Construct {
    ?n geom:geometry ?g .
    ?g ogc:asWKT ?o
}

With

?n = uri(lgd:node, ?id)
?g = uri(lgd-geom:node, ?id)
?o = typedLiteral(?geom,
                   ogc:wktLiteral)

From nodes

Aimed for RDF Output

@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .

...  
lgd:node1 geom:geometry lgd-geom:node1 .
lgd:node2 geom:geometry lgd-geom:node2 .

lgd-geom:node1 ogc:asWKT "POINT(0 0)"^^ogc:wktLiteral .
lgd-geom:node2 ogc:asWKT "POINT(1 1)"^^ogc:wktLiteral .
Tooling

Website: http://sml.aksw.org

- R2RML $\leftrightarrow$ SML converter
- Syntax Highlighters for vim and CodeMirror (a JavaScript IDE component; used in the user study).
- SML in use at LinkedGeoData and Panlex
Outline

1 Motivation

2 R2RML in a Nutshell

3 SML in a Nutshell

4 SML Step by Step Example

5 Evaluation
We performed a user study with the goal to answer the following questions:

- Is SML easier to read than R2RML and does SML have a lower entry barrier than R2RML?
- Can people understand SML mappings or R2RML mappings faster?
- If given the choice, would people prefer SML or R2RML?

46 humans completed the survey of which 28 performed all tasks correctly.
User Study - Approach

- Participants first were asked to do a self-assessment on their familiarity with technologies related to RDB2RDF.
- Then they were presented 5 multiple-choice tasks each for R2RML and SML (10 tasks in total).
- Finally, after having completed the tasks, users could score their impression and preference on R2RML / SML.
### Familiarity

- **The topic of RDB2RDF is (or may become) relevant for one of my projects**
  
  1 = not at all ... 5 = absolutely
  
  ![RDB2RDF relevance scale](1 2 3 4 5)

- **I am familiar with the Turtle RDF syntax**
  
  1 = not at all, 2 = have seen it before, 3 = know some basic concepts, 4 = capable of working with it, 5 = can write it from scratch
  
  ![Turtle RDF familiarity scale](1 2 3 4 5)

- **I am familiar with the SPARQL syntax**
  
  1 = not at all, 2 = have seen it before, 3 = know some basic concepts, 4 = can write some simple queries from scratch, 5 = can write rather sophisticated queries from scratch
  
  ![SPARQL familiarity scale](1 2 3 4 5)

- **I am familiar with the SQL syntax**
  
  1 = not at all, 2 = have seen it before, 3 = know some basic concepts, 4 = can write some simple queries from scratch, 5 = can write rather sophisticated queries from scratch
  
  ![SQL familiarity scale](1 2 3 4 5)

- **I am familiar with R2RML**
  
  1 = not at all, 2 = have seen it before, 3 = know some basic concepts, 4 = capable of using it with reference information, 5 = can write mappings from scratch
  
  ![R2RML familiarity scale](1 2 3 4 5)

- **I am familiar with SML**
  
  1 = not at all, 2 = have seen it before, 3 = know some basic concepts, 4 = capable of using it with reference information, 5 = can write mappings from scratch
  
  ![SML familiarity scale](1 2 3 4 5)
User Study - Task 1 - SML

Task 1 (SML): Find the Output [Warm Up]
One simple task for each R2RML and SML

* Mark all the triples that are generated from the given table using the given view.
(Please note that the symbol a is a shorthand for rdf:type.)

<table>
<thead>
<tr>
<th>departments</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>id (int)</td>
<td>name (text)</td>
</tr>
<tr>
<td>1</td>
<td>Development</td>
</tr>
</tbody>
</table>

Prefix ex: <http://example.com/>

Create View DepartmentsView As
Construct {
  ?s a ex:Department
}
With
  ?s = uri(ex:, ?id)
From
  departments

Check any that apply

- 1 <http://example.com/1> ex:id 1 .
- 1 <http://example.com/Department> a ex:Department .
- 1 <http://example.com/1> a ex:Department .
- 1 ex:Department a "1" .
- I cannot make sense out of this mapping
Task 1 (R2RML): Find the Output [Warm Up]

- Mark all the triples that are generated from the given table using the given view.
(Please note that the symbol a is a shorthand for rdf:type.)

<table>
<thead>
<tr>
<th>employees</th>
</tr>
</thead>
<tbody>
<tr>
<td>id (int)</td>
</tr>
<tr>
<td>1</td>
</tr>
</tbody>
</table>

```rml
@prefix rr: <http://www.w3.org/ns/r2rml#> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix ex: <http://example.com/> .

<EmployeesMap>
a rr:TriplesMap;
  rr:logicalTable [ rr:tableName "employees" ];
  rr:subjectMap [ rr:template "http://example.com/{id}" ];
  rr:predicateObjectMap [ rr:predicate rdf:type ;
    rr:object ex:Employee ];
</EmployeesMap>

Check any that apply

- [ ] 1 <http://example.com/1> ex:id 1 .
- [ ] 1 ex:Employee a "1" .
- [ ] 1 <http://example.com/Susan> a ex:Employee .
- [ ] 1 <http://example.com/1> a ex:Employee .
- [ ] I cannot make sense out of this mapping
### User Study - Readability

1. **I found the tasks too difficult (1=not at all ... 5=absolutely)**
   - 1  2  3  4  5

2. **I was able to make sense of the SML mappings (1=not at all ... 5=absolutely)**
   - 1  2  3  4  5

3. **I was able to make sense of the R2RML mappings (1=not at all ... 5=absolutely)**
   - 1  2  3  4  5

4. **I found SML to be easily readable (1=not at all ... 5=absolutely)**
   - 1  2  3  4  5

5. **I found R2RML to be easily readable (1=not at all ... 5=absolutely)**
   - 1  2  3  4  5

6. **I could imagine using SML for solving RDB2RDF mapping tasks (1=not at all ... 5=absolutely)**
   - 1  2  3  4  5

• Which of the languages did you prefer over the other?
  1=strong preference for R2RML, 2=weak preference for R2RML
  3=indifferent
  4=weak preference for SML, 5=strong preference for SML

   - 1  2  3  4  5
Results: Readability

- Readability of SML better than R2RML for novices.
Results: Preference

- Novice = Self assessment in R2RML familiarity $\leq 3$
- Expert = Self assessment in R2RML familiarity $\geq 4$
Conclusions and Future Work

- We introduced the novel Sparqlification Mapping Language (SML) and showed how it relates to R2RML.
- Evaluation shows a favor in SML by RDB2RDF novices, providing evidence that SML could simplify RDB2RDF mapping.
- We provided tooling to bridge the gap between SML and R2RML.

Future Work

- More testing of the converters (WIP)
- Possibly streamline some language features, such as
  - Usage SPARQL 1.1’s `strdt` and `strlang` in favor of `plainLiteral` and `typedLiteral`
  - Introduction of a `FROM QUERY` syntax instead of interpreting content of triple quotes as an SQL query.
SML Resources: http://sml.aksw.org

Claus Stadler
cstadler@informatik.uni-leipzig.de
AKSW/Uni Leipzig

Jens Lehmann
lehmann@informatik.uni-leipzig.de
AKSW/Uni Leipzig

http://geoknow.eu