Publishing geospatial information as RDF graphs

Kostis Kyzirakos, Dimitrianos Savva
Outline

• Mapping relational data to RDF graphs
• Mapping non-relational data to RDF graphs
• Geospatial Extensions for mapping geospatial data to RDF graphs
• Implemented Systems
• Demonstration
Mapping relational data to RDF graphs

Natura 2000 is an ecological network designated under the Birds Directive and the Habitats Directive which form the cornerstone of the nature conservation policy of the European Union.


<table>
<thead>
<tr>
<th>Sitecode</th>
<th>Sitename</th>
<th>ReleaseDate</th>
</tr>
</thead>
<tbody>
<tr>
<td>DE0916391</td>
<td>NTP S-H W</td>
<td>2011-01-27</td>
</tr>
<tr>
<td>DE1003301</td>
<td>DOGGERB ANK</td>
<td>2011-01-27</td>
</tr>
</tbody>
</table>
Direct Mapping

W3C Recommendation from 2012
http://www.w3.org/TR/rdb-direct-mapping/

- Relational **tables** are mapped to **classes** defined by an RDF vocabulary.

- **Attributes** of each table are mapped to RDF **properties** that represent the relation between subject and object resources.

- **Identifiers**, class **names**, **properties** and **instances** are **generated automatically** following the labels of the input data.
Direct Mapping - Example

<table>
<thead>
<tr>
<th>Sitecode</th>
<th>Sitename</th>
<th>ReleaseDate</th>
</tr>
</thead>
<tbody>
<tr>
<td>DE0916391</td>
<td>NTP S-H W</td>
<td>2011-01-27</td>
</tr>
<tr>
<td>DE1003301</td>
<td>DOGGERBANK</td>
<td>2011-01-27</td>
</tr>
</tbody>
</table>

@base <http://foo.example/DB/> .
@prefix rdf: <http://www.w3.org/1999/02-22-rdf-syntax-ns#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .

<ProtectedArea/Sitecode=DE0916391> rdf:type <ProtectedArea> .
<ProtectedArea/Sitecode=DE0916391> <ProtectedArea#Sitename> "NTP S-H W" .
<ProtectedArea/Sitecode=DE0916391> <ProtectedArea#ReleaseDate> "2011-01-27"^^xsd:date .

<ProtectedArea/Sitecode=DE1003301> rdf:type <ProtectedArea> .
<ProtectedArea/Sitecode=DE1003301> <ProtectedArea#Sitename> "DOGGERBANK" .
<ProtectedArea/Sitecode=DE1003301> <ProtectedArea#ReleaseDate> "2011-01-27"^^xsd:date .
The language R2RML

• R2RML is a language for expressing customized mappings from relational databases to RDF graphs
• R2RML is a W3C Recommendation from 2012 http://www.w3.org/TR/r2rml/
• R2RML mappings provide the user with the ability to express the desired transformation of existing relational data into the RDF data model, following a structure and a target vocabulary that is chosen by the user.
The language R2RML (cont’d)
The language R2RML (cont’d)

• A **logical table** can be
  • a **relational table** that is explicitly stored in the database
  • an **SQL view**
  • an **SQL select query**

• A **triples map** is a rule that defines how each tuple of the logical table will be mapped to a set of RDF triples. It consists of
  • a subject map
  • zero or more predicate-object maps.
The language R2RML (cont’d)

• A **subject map** is a rule that defines how to generate the URI that will be the subject of each generated RDF triple.

• A **predicate-object map** consists of predicate maps and object maps.

• A **predicate map** defines the **RDF property** to be used to relate the subject and the object of the generated triple.

• An **object map** defines how to generate the **object** of the triple which originates from the current row of the logical table.
The language R2RML (cont’d)

• Subject, predicate, object and graph maps are term maps. A term map is a function that generates an RDF term from a logical table. Three types of term maps are defined:
  • constant-valued term maps
  • column-valued term maps
  • template-valued term maps
The language R2RML (cont’d)

- A **referencing object map** allows using the subjects of another triples map as the objects generated by a predicate-object map.
  - Optionally, it has one or more join condition properties.

[source: http://www.w3.org/TR/r2rml/#dfn-predicate-map]
The language R2RML – Example

<table>
<thead>
<tr>
<th>Sitecode</th>
<th>Sitename</th>
<th>ReleaseDate</th>
</tr>
</thead>
<tbody>
<tr>
<td>DE0916391</td>
<td>NTP S-H W</td>
<td>2011-01-27</td>
</tr>
<tr>
<td>DE1003301</td>
<td>DOGGERBANK</td>
<td>2011-01-27</td>
</tr>
</tbody>
</table>

@base <http://foo.example/DB/> .
<NaturaMapping>
rr:subjectMap [ rr:template "ProtectedArea/SiteCode={SiteCode}";
rr:class <ProtectedArea> ];
rr:predicateObjectMap [ rr:predicate ProtectedArea:SiteName;
rr:objectMap [ rr:column "SiteName"; ]; ] .

<ProtectedArea/Sitecode=DE0916391> rdf:type <ProtectedArea> .
<ProtectedArea/Sitecode=DE0916391> <ProtectedArea#Sitename> "NTP S-H W" .

<ProtectedArea/Sitecode=DE1003301> rdf:type <ProtectedArea> .
<ProtectedArea/Sitecode=DE1003301> <ProtectedArea#Sitename> "DOGGERBANK" .
Mapping non-relational data to RDF graphs

**OpenStreetMap** is a collaborative project for publishing free maps of the world. OSM maintains a community-driven global editable map that gathers map data in a crowdsourcing fashion.


```xml
<ogr:FeatureCollection>
  <gml:featureMember>
    <ogr:waterways fid="waterways.128">
      <ogr:osm_id>8108139</ogr:osm_id>
      <ogr:name>Lech</ogr:name>
      <ogr:type>river</ogr:type>
      <ogr:geometryProperty>
        <gml:LineString>
          <gml:coordinates>
            10.9034096,47.7996669
            10.9037025,47.8003338 ...
          </gml:coordinates>
        </gml:LineString>
      </ogr:geometryProperty>
    </ogr:waterways>
  </gml:featureMember>
</ogr:FeatureCollection>
```
RDF Mapping Language (RML)

- RML is a recently proposed mapping language that defines how to map heterogeneous sources into RDF. [http://semweb.mmlab.be/rml/spec.html](http://semweb.mmlab.be/rml/spec.html)
- RML is defined as a superset of the W3C-standard R2RML

<table>
<thead>
<tr>
<th>R2RML</th>
<th>RML</th>
</tr>
</thead>
<tbody>
<tr>
<td>Logical Table</td>
<td><code>rr:logicalTable</code></td>
</tr>
<tr>
<td>Table Name</td>
<td><code>rr:tableName</code></td>
</tr>
<tr>
<td>Column</td>
<td><code>rr:column</code></td>
</tr>
<tr>
<td>SQL</td>
<td><code>rml:referenceFormulation</code></td>
</tr>
<tr>
<td>per row iteration</td>
<td><code>rml:iterator</code></td>
</tr>
<tr>
<td>URI</td>
<td><code>rml:source</code></td>
</tr>
<tr>
<td>reference</td>
<td><code>rml:reference</code></td>
</tr>
</tbody>
</table>

source: [http://semweb.mmlab.be/rml/RML_R2RML.html](http://semweb.mmlab.be/rml/RML_R2RML.html)
RML extensions

- A **logical source** refers to the input dataset that will be converted to an RDF graph.
- Each logical source has
  - a **source** property pointing to input data
  - a logical **iterator** that defines the iteration pattern over the input data source
  - an optional **reference formulation** property that defines the query language that may be used (e.g., SQL2008, XPath, JSONPath)
- An RML **reference** is a term map that refers to a column name (SQL, CSV), an XML element or attribute, or an JSON object.
### RML Example

```xml
<#waterways>
  rml:logicalSource [ 
    rml:source "~/home/leo/osm.gml";
    rml:referenceFormulation ql:XPath;
    rml:iterator "~/ogr:FeatureCollection/ogr:featureMember/ogr:waterways";
  ];
  rr:subjectMap [ 
    rr:template "http://www.example.com/id/{@fid}";
    rr:class onto:waterways;
  ];
  rr:predicateObjectMap [ 
    rr:predicate onto:hasOgr-Name;
    rr:objectMap [ 
      rr:datatype xsd:string;
      rml:reference "ogr:name";
    ]; ] .
```

```sparql
ex_id:waterways.128 rdf:type onto:waterways ;
onto:hasOgr-Name "Lech" ;
onto:hasFid "waterways.128"^^xsd:ID ;
onto:hasOgr-Osm_id "8108139" ;
onto:hasOgr-Type "river" .
```
Mapping geospatial data to RDF graphs

Geospatial data are available in formats such as:

- ESRI shape files
- KML documents
- GeoJSON documents
- XML documents

Geospatial data may also be stored in spatially-enabled relational databases.
Extending R2ML with transformation-valued term maps
Extending RML with transformation-valued term maps

Graph:
- TriplesMap
  - SubjectMap
    - LogicalSource
    - Source
    - Iterator
    - Reference Formulation
  - PredicateObjectMap
  - PredicateMap
  - ObjectMap
  - RefObjectMap
- TermMap
  - Join
  - Constant
  - Column
  - Template
  - Function
  - Argument Map
  - Function
  - Argument Map
  - Parent
  - Child
Transformation-valued term maps

• A **transformation-valued term maps** is a term map that generates an RDF term by applying a **SPARQL extension function** on one or more term maps.

• A transformation-valued term map has
  
  • exactly one **rrx:function** property that defines a **SPARQL extension function** that performs the desired transformation
  
  • one **rrx:argumentMap** property that has as range an **rdf:List** of term maps that define the **arguments** to be passed to the transformation function
## Transformation-valued term maps (cont’d)

<table>
<thead>
<tr>
<th>Extension Function URI</th>
<th>Corresponding GeoSPARQL property</th>
</tr>
</thead>
<tbody>
<tr>
<td>strdf:dimension</td>
<td>geo:dimension</td>
</tr>
<tr>
<td>strdf:spatialDimension</td>
<td>geo:spatialDimension</td>
</tr>
<tr>
<td>strdf:coordinateDimension</td>
<td>geo:coordinateDimension</td>
</tr>
<tr>
<td>strdf:isEmpty</td>
<td>geo:isEmpty</td>
</tr>
<tr>
<td>strdf:isSimple</td>
<td>geo:isSimple</td>
</tr>
<tr>
<td>strdf:asText</td>
<td>geo:asWKT geo:hasSerialization</td>
</tr>
<tr>
<td>strdf:asGML</td>
<td>geo:asGML</td>
</tr>
</tbody>
</table>
Extending join conditions
Example

<table>
<thead>
<tr>
<th>Sitecode</th>
<th>Sitename</th>
<th>Geometry</th>
</tr>
</thead>
<tbody>
<tr>
<td>DE0916391</td>
<td>NTP S-H W</td>
<td>POLYGON(...)</td>
</tr>
<tr>
<td>DE1003301</td>
<td>DOGGERBANK</td>
<td>POLYGON(...)</td>
</tr>
</tbody>
</table>

```
<NaturaGeometryMapping>
  rr:subjectMap [  
    rr:template "ProtectedArea/Geometry/SiteCode={SiteCode}";
    rr:class geo:Geometry ];
  rr:predicateObjectMap [  
    rr:predicate geo:dimension;
    rr:objectMap [  
      rrx:function strdf:dimension;
      rrx:argumentMap ( [rr:column "\Geom"] ); ] ];

<ProtectedArea/Geometry/Sitecode=DE0916391>
  rdf:type <ProtectedArea> ;
  geo:dimension "2"^xsd:integer .
```
### Example

<table>
<thead>
<tr>
<th>Sitecode</th>
<th>Sitename</th>
<th>Geom</th>
</tr>
</thead>
<tbody>
<tr>
<td>DE0916391</td>
<td>NTP S-H W</td>
<td>POLYGON((...)</td>
</tr>
<tr>
<td>DE1003301</td>
<td>DOGGERB ANK</td>
<td>POLYGON((...)</td>
</tr>
</tbody>
</table>

```xml
<NaturaGeometryMapping>
  rr:subjectMap [
    rr:template "ProtectedArea/Geometry/SiteCode={SiteCode}";
    rr:class geo:Geometry ];
  rr:predicateObjectMap [
    rr:predicate geo:sfIntersects;
    rr:objectMap [
      rr:parentTriplesMap <#waterwaysGeom> ;
      rr:joinCondition [ rr:column "`Geom`" ] ;
      rml:reference "ogr:geometryProperty";
      rr:parentTriplesMap <#waterwaysGeom> ];
</NaturaGeometryMapping>

```

```xml
<OSM Waterways>
  <ogr:FeatureCollection>
    <ogr:waterways fid="waterways.128">
      <ogr:osm_id>8108139</ogr:osm_id>
      <ogr:geometryProperty>
        <gml:LineString>
          <gml:coordinates>
            10.9034096,47.7996669 ...
          </gml:coordinates>
        </gml:LineString>
      </ogr:geometryProperty>
    </ogr:waterways>
  </ogr:FeatureCollection>
</OSM Waterways>

```

```xml
```

```
```
Implemented Systems

• Direct Mapping processors:
  • SquirellRDF

• R2RML processors:
  • D2RQ Platform
  • OpenLink Virtuoso
  • Ultrawrap
  • Morph
  • Ontop
  • Oracle

• RML processor
  • Processor by iMinds Lab, Ghent University

• Other Mapping Language:
  • Triplify

• Geospatial capabilities
• So far:
  • Geometry2RDF
  • Sparqlify
  • TripleGeo
  • GeoTriples
<table>
<thead>
<tr>
<th>OpenLink Virtuoso</th>
<th>Custom Mapping Language</th>
<th>Direct Mapping</th>
<th>R2RML</th>
<th>RML</th>
<th>SPARQL query evaluation</th>
<th>Automatic Mapping Generation</th>
<th>Geospatial support</th>
</tr>
</thead>
<tbody>
<tr>
<td>✓</td>
<td>✓</td>
<td>✓*</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>RDF-RDB2RDF</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>D2RQ Platform</td>
<td>✓</td>
<td>✗</td>
<td>✓</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
<td>✗</td>
</tr>
<tr>
<td>Db2triples</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>Morph</td>
<td>✓</td>
<td>✗</td>
<td>✓</td>
<td>✗</td>
<td>✓</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>Sparqlify</td>
<td>✓</td>
<td>✗</td>
<td>✓*</td>
<td>✗</td>
<td>✓</td>
<td>✗</td>
<td>√</td>
</tr>
<tr>
<td>Ontop</td>
<td>✓</td>
<td>✗</td>
<td>✓</td>
<td>✗</td>
<td>✓</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>Ultrawrap</td>
<td>✓*</td>
<td>✓</td>
<td>✓</td>
<td>✗</td>
<td>✓</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>Oracle</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
<td>✗</td>
</tr>
<tr>
<td>Geometry2RDF</td>
<td>✗</td>
<td>✓</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>√*</td>
</tr>
<tr>
<td>TriplesGeo</td>
<td>✗</td>
<td>✓</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>√*</td>
</tr>
<tr>
<td>iMinds lab RML processor</td>
<td>✗</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>GeoTriples</td>
<td>✗</td>
<td>✗</td>
<td>✓</td>
<td>✗</td>
<td>✓*</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>
## Comparison of Geo2RDF tools

<table>
<thead>
<tr>
<th></th>
<th>Direct Mapping</th>
<th>R2RML</th>
<th>RML</th>
<th>Automatic Mapping Generation</th>
<th>GeoSPARQL compliance</th>
<th>RDBMS</th>
<th>ESRI Shape file</th>
<th>GML</th>
<th>Geo JSON</th>
</tr>
</thead>
<tbody>
<tr>
<td>Geometry2RDF</td>
<td>✔</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✔</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>TriplesGeo</td>
<td>✔</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>(✓)</td>
<td>✔</td>
<td>✔</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>GeoTriples</td>
<td>✗</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
</tbody>
</table>
GeoTriples

• Open Source software
• Released under Mozilla Public Licence v2.0
• Available at: https://github.com/LinkedEOData/GeoTriples
• Extends the D2RQ Platform
• Extends the iMinds lab RML processor
• Provides both a graphical user interface and a command line interface
Architecture of GeoTriples

Earth Observation Acquisitions
Automatic generation of R2RML mappings (cont’d)

• Generate **two** triples maps for each table that has a *geometry* column.
  - **Thematic** triples map for the non-geometric information
  - **Spatial** triples map for the geometric information

• The spatial triples map contains multiple **transformation functions** over the input geometries in order to generate a **GeoSPARQL** compliant dataset.
Automatic generation of RML mappings for GML documents

• Each geometric object is mapped to a `geo:Geometry` instance
• For each geometric object we generate a set of predicate object maps that use the appropriate transformation functions for producing a GeoSPARQL compliant dataset
• Each simple element is mapped to a predicate object map
• Each non simple element is mapped to a triples map
• Appropriate mappings are generated for linking nested elements
Demonstration