Open Information Systems

Lecture 7: UPLIFT – Mapping Relational Databases to RDF

Christophe Debruyne

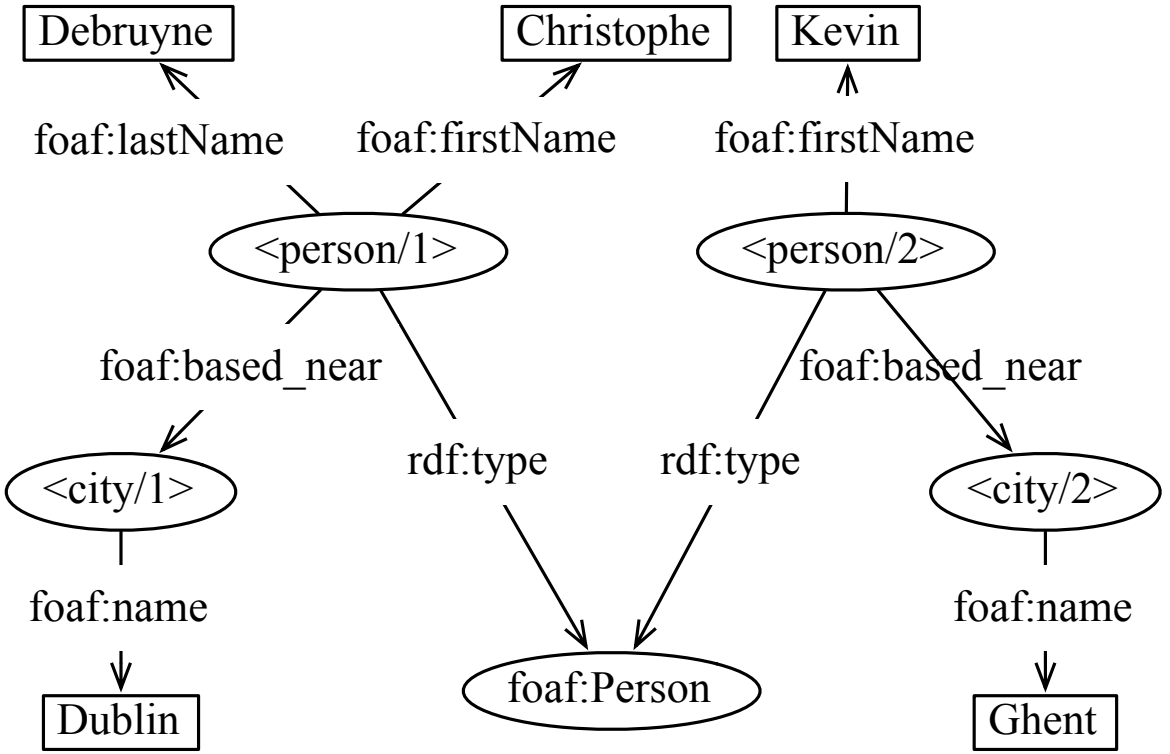
RDB2RDF

Person

ID	Fist	Last	CityID
1	Christophe	Debruyne	1
2	Kevin	*NULL*	2

City

ID	Name
1	Dublin
2	Ghent



RDB2RDF: W3C Recommendations

- It all started with Tim Berners-Lee proposing a direct mapping from RDBs to RDF.
- Over the years, two recommendations (standards) to map relational data to RDF emerged.
  - A **Direct Mapping** of relational data to RDF
  - **R2RML**: an RDB to RDF Mapping Language that is highly customizable to annotate relational data with ontologies to generate RDF.

RDB2RDF

The “started” started with Berners-Lee discussing a set of mappings – that can be generated automatically – between relational databases and RDF, see “Relational Databases on the Semantic Web” via http://www.w3.org/DesignIssues/RDB-RDF.html

“The semantic web data model is very directly connected with the model of relational databases. A relational database consists of tables, which consists of rows, or records. Each record consists of a set of fields. The record is nothing but the content of its fields, just as an RDF node is nothing but the connections: the property values. **The mapping is very direct**

- **a record is an RDF node;**
- **the field (column) name is RDF propertyType; and**
- **the record field (table cell) is a value.”**

Indeed, it will not always be that simple. Can you give examples?

Direct Mappings

TBL proposed a direct mapping. **Direct mappings** immediately reflect the structure of the database → The target RDF vocabulary directly reflects the names of database schema elements, and neither structure nor target vocabulary can be changed. [R2RML]

Process:

- **Existing table and column names are encoded into URIs.**
- Data is (i) *extracted*, (ii) *transformed* into RDF and then (iii) *loaded* into a triplestore.
- This is thus an ETL process.

This proposal – over time – was refined into a W3C recommendation, published in fall 2012, called "A Direct Mapping of Relational Data to RDF" http://www.w3.org/TR/rdb-direct-mapping/

Direct Mappings

- The database (both schema and data), primary keys and foreign keys are given to a *direct mapping engine* to produce an RDF graph.
  - Fields are mapping to literals;
  - Primary keys are used to construct URIs for resources;
  - And foreign keys are used to construct properties and relate resources.

- Example…

Direct Mapping Example

People		
PK		*→Addresses(ID)*
ID	**fname**	**addr**
1	Christophe	1
2	Kevin	NULL

Addresses	
PK	
ID	**city**
1	Brussels

Given a base URI
http://foo.example/DB/

```
@base <http://foo.example/DB/> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .

<People/ID=1> a <People> .
<People/ID=1> <People#ID> 1 .
<People/ID=1> <People#fname> "Christophe" .
<People/ID=1> <People#addr> 1 .
<People/ID=1> <People#ref-addr> <Addresses/ID=1> .
<People/ID=2> a <People> .
<People/ID=2> <People#ID> 2 .
<People/ID=2> <People#fname> "Kevin" .

<Addresses/ID=1> a <Addresses> .
<Addresses/ID=1> <Addresses#ID> 1 .
<Addresses/ID=1> <Addresses#city> ”Brussels" .
```

Direct Mappings

Small discussion:

Are direct mappings meaningful?

Can you identify potential problems?

R2RML

- R2RML: RDB to RDF Mapping Language
  - A W3C Recommendation since fall 2012
  - http://www.w3.org/TR/r2rml/

- Creating an R2RML file that annotates a relational database with existing vocabularies and/or ontologies (RDFS or OWL).
- That R2RML file goes through an *R2RML Mapping Engine* to produce RDF.
- R2RML specified
  - An ontology to specify those mappings;
  - How those mappings should be interpreted to produce RDF.
  - R2RML files are thus stored as RDF.

R2RML: Running Example

Addresses	
PK	
ID	**city**
1	Brussels

What is being mapped? A logical table/view or an SQL query.

How to generate and state something about the subject of those triples.

How to generate predicates and objects.

```
@prefix rr: <http://www.w3.org/ns/r2rml#> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
@prefix dbpedia: <http://dbpedia.org/ontology/> .

<#AddressTripleMap>
     a rr:TriplesMap ;

     rr:logicalTable [ rr:tableName "Addresses" ] ;

     rr:subjectMap [
          rr:template "http://foo.example/Addresses/{ID}" ;
          rr:class dbpedia:Place
     ] ;

     rr:predicateObjectMap [
          rr:predicate foaf:name ;
          rr:objectMap [ rr:column "city" ]
     ] ;
     .
```

R2RML: Running Example

Addresses	
PK	
ID	**city**
1	Brussels

```
@prefix rr: <http://www.w3.org/ns/r2rml#> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
@prefix dbpedia: <http://dbpedia.org/ontology/> .

<#AddressTripleMap>
     a rr:TriplesMap ;
     rr:logicalTable [
          rr:sqlQuery """SELECT ID, city FROM
                         Addresses WHERE 1"""
     ] ;
     rr:subjectMap [
          rr:template "http://foo.example/Addresses/{ID}" ;
          rr:class dbpedia:Place
     ] ;
     rr:predicateObjectMap [
          rr:predicate foaf:name ;
          rr:objectMap [ rr:column "city" ]
     ] ;
     .
```

R2RML: Running Example

People		
PK		*→Addresses(ID)*
ID	**fname**	**addr**
1	Christophe	1
2	Kevin	NULL

Addresses	
PK	
ID	**city**
1	Brussels

Relating People to Addresses.

```
<#PersonTriplesMap>
     a rr:TriplesMap;
     rr:logicalTable [ rr:tableName "People" ] ;
     rr:subjectMap [
          rr:template "http://foo.example/Person/{ID}" ;
          rr:class foaf:Person
     ] ;
     rr:predicateObjectMap [
          rr:predicate foaf:name ;
          rr:objectMap [ rr:column "fname" ]
     ] ;
     rr:predicateObjectMap [
          rr:predicate foaf:based_near ;
          rr:objectMap [
               rr:parentTriplesMap <#AddressTripleMap> ;
               rr:joinCondition [
                    rr:child "addr" ;
                    rr:parent "ID"
               ]
          ]
     ] ;
     .
```

R2RML: Running Example

- Using R2RML-F (http://github.com/chrdebru/r2rml)
- The configuration file contains:

```
connectionURL = jdbc:mysql://localhost/r2rml
user = foo
password = bar
mappingFile = db-mapping.ttl
outputFile = output.ttl
format = TURTLE
```

- And then we execute:

```
$ java -jar r2rml/r2rml.jar db-config.properties
```

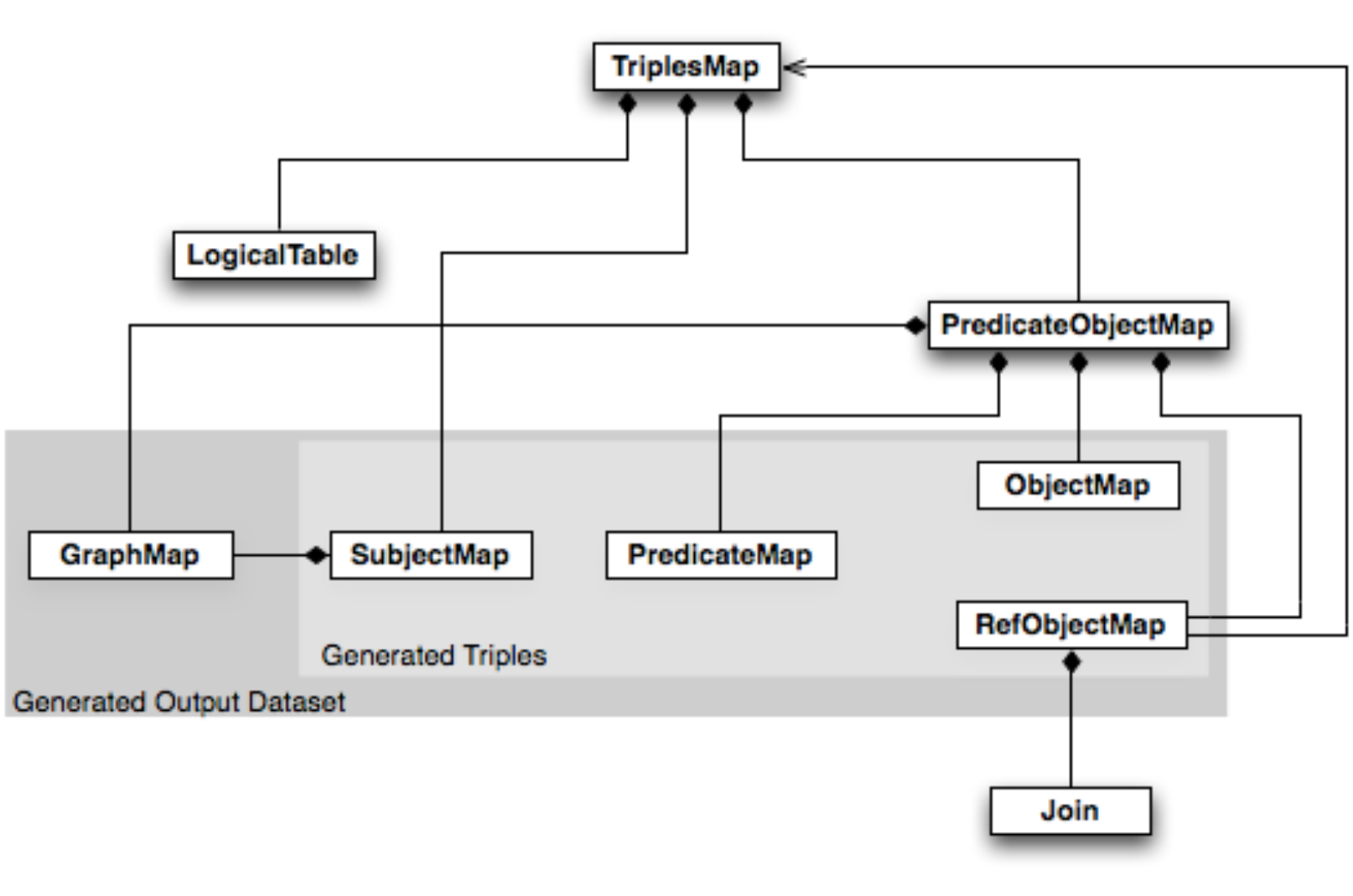
R2RML: Running Example

output.ttl — r2rml-ois Add License

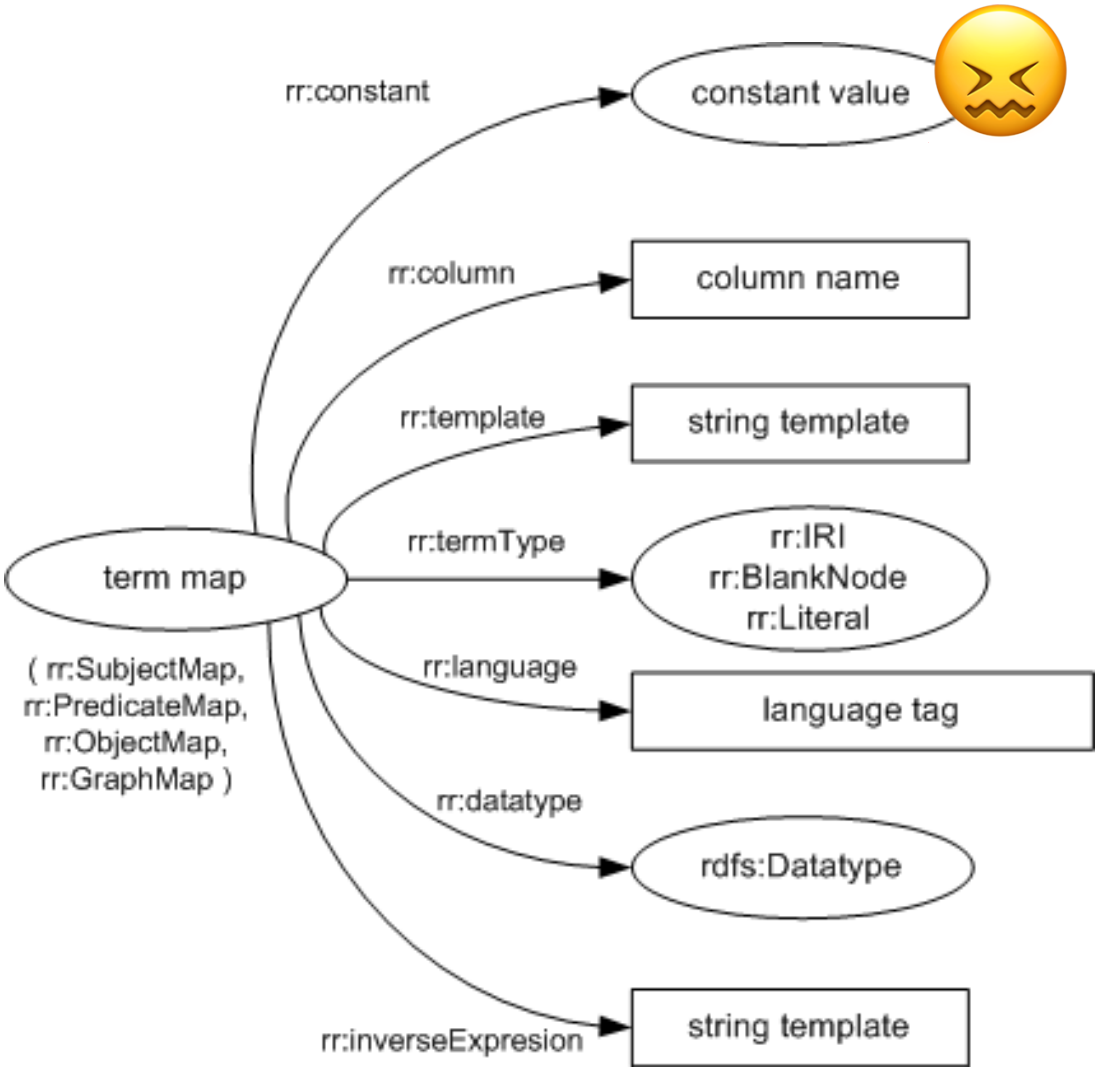
```
<http://foo.example/Person/2>
        a       <http://xmlns.com/foaf/0.1/Person> ;
        <http://xmlns.com/foaf/0.1/name>
                "Kevin" .

<http://foo.example/Person/1>
        a       <http://xmlns.com/foaf/0.1/Person> ;
        <http://xmlns.com/foaf/0.1/based_near>
                <http://foo.example/Addresses/1> ;
        <http://xmlns.com/foaf/0.1/name>
                "Christophe" .

<http://foo.example/Addresses/1>
        a       <http://dbpedia.org/ontology/Place> ;
        <http://xmlns.com/foaf/0.1/name>
                "Brussels" .
```

Line: 1 Turtle Tab Size: 4

R2RML


TriplesMap
LogicalTable
PredicateObjectMap
ObjectMap
GraphMap
SubjectMap
PredicateMap
RefObjectMap
Generated Triples
Generated Output Dataset
Join

R2RML



- If rr:constant is used for a Subject Map, Graph Map, or Predicate Map, then the value must be an IRI.
- If rr:constant is used for an Object Map, then its value must be either a literal or an IRI.
- So either a rectangle or an ellipse, depending on which…

- rr:inverseExpressions are useful only when accessing RDBs as virtual graphs (they are an "optimization hint").

rr:class in rr:SubjectMaps

```
rr:subjectMap [
   rr:template "http://foo.example/Addresses/{ID}" ;
   rr:class dbpedia:Place
] ;
```

This subject map generates for each subject a triple with rdf:type as predicate and dbpedia:Place as object. In other words, rdf:type and dbpedia:Place are constants. Can we use an rr:PredicateObjectMap with two constants?

Yes:

```
rr:predicateObjectMap [
   rr:predicate rdf:type ;
   rr:object dbpedia:Place
] ;
```

When the FK is in your table

People		
PK		*→Addresses(ID)*
ID	**fname**	**addr**

Addresses	
PK	
ID	**city**

When you join tables based on a FK, and the PK of the parent is used for the Subject Map, there can be a possibility to avoid join. How?

```
<#PersonTriplesMap>
     a rr:TriplesMap;
     rr:logicalTable [ rr:tableName "People" ] ;
     rr:subjectMap [
       rr:template "http://foo.example/Person/{ID}" ;
       rr:class foaf:Person
     ] ;
     rr:predicateObjectMap [
       rr:predicate foaf:name ;
       rr:objectMap [ rr:column "fname" ]
     ] ;
     rr:predicateObjectMap [
       rr:predicate foaf:based_near ;
       rr:objectMap [
          rr:parentTriplesMap <#AddressTripleMap> ;
          rr:joinCondition [
            rr:child "addr" ;
            rr:parent "ID"
          ]
       ]
     ] ;
     .
```

When the FK is in your table

People		
PK		*→Addresses(ID)*
ID	**fname**	**addr**

Addresses	
PK	
ID	**city**

Create a Term Map that generates the same resource!
Prone to errors if, for instance, templates change, though.

```
<#PersonTriplesMap>
     a rr:TriplesMap;
     rr:logicalTable [ rr:tableName "People" ] ;
     rr:subjectMap [
       rr:template "http://foo.example/Person/{ID}" ;
       rr:class foaf:Person
     ] ;
     rr:predicateObjectMap [
       rr:predicate foaf:name ;
       rr:objectMap [ rr:column "fname" ]
     ] ;
     rr:predicateObjectMap [
       rr:predicate foaf:based_near ;
       rr:objectMap [
         rr:template "http://foo.example/Addresses/{addr}" ;
       ]
     ] ;
     .
```

Graph Maps

- Any subject map or predicate-object map may have one or more associated graph maps. They are specified in one of two ways. Either by associating a constant with the property rr:graph, or by providing a Graph Map (which returns IRIs) with the property rr:graphMap. rr:defaultGraph is reserved for the default (nameless) graph.

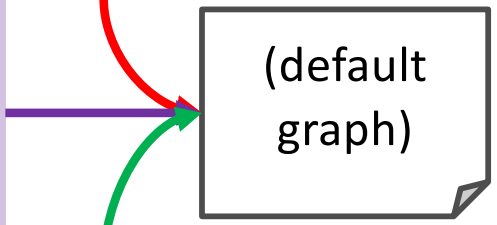
- If a Subject Map has no Graph Maps then the set of Graph Maps is {rr:defaultGraph}.

- If both the Subject Map and a Predicate Object Map have no Graph Maps, then the set of Graph Maps is {rr:defaultGraph}. Otherwise it is, for each Predicate Object Map, the union of both graph sets.

- It is actually more simple than it sounds, let's exemplify!

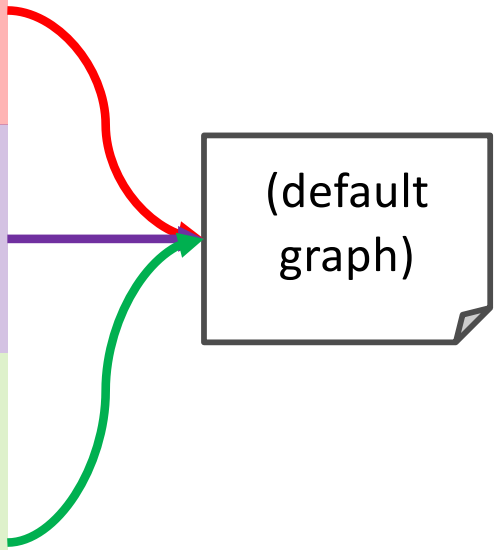
Graph Maps

```
<#PersonTriplesMap>
    a rr:TriplesMap;
    rr:logicalTable [ rr:tableName "People" ] ;
    rr:subjectMap [
      rr:template "http://foo.example/Person/{ID}" ;
      rr:class foaf:Person ;

    ] ;
    rr:predicateObjectMap [
      rr:predicate foaf:name ;
      rr:objectMap [ rr:column "fname" ] ;

    ] ;
    rr:predicateObjectMap [
      rr:predicate foaf:based_near ;
      rr:objectMap [
        rr:parentTriplesMap <#AddressTripleMap> ;
        rr:joinCondition [ rr:child "addr" ; rr:parent "ID" ]
      ] ;

    ] ;
    .
```



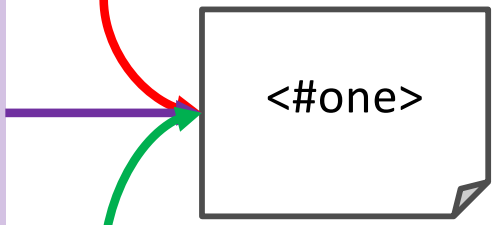
Graph Maps

```
<#PersonTriplesMap>
    a rr:TriplesMap;
    rr:logicalTable [ rr:tableName "People" ] ;
    rr:subjectMap [
      rr:template "http://foo.example/Person/{ID}" ;
      rr:class foaf:Person ;
      rr:graph rr:defaultGraph
    ] ;
    rr:predicateObjectMap [
      rr:predicate foaf:name ;
      rr:objectMap [ rr:column "fname" ] ;

    ] ;
    rr:predicateObjectMap [
      rr:predicate foaf:based_near ;
      rr:objectMap [
        rr:parentTriplesMap <#AddressTripleMap> ;
        rr:joinCondition [ rr:child "addr" ; rr:parent "ID" ]
      ] ;

    ] ;
    .
```



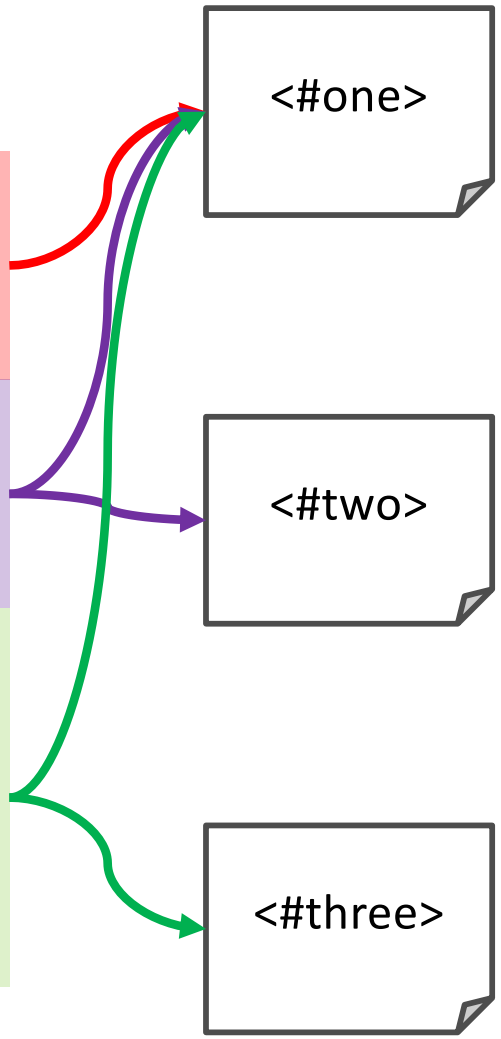
Graph Maps

```
<#PersonTriplesMap>
    a rr:TriplesMap;
    rr:logicalTable [ rr:tableName "People" ] ;
    rr:subjectMap [
      rr:template "http://foo.example/Person/{ID}" ;
      rr:class foaf:Person ;
      rr:graph <#one>
    ] ;
    rr:predicateObjectMap [
      rr:predicate foaf:name ;
      rr:objectMap [ rr:column "fname" ] ;

    ] ;
    rr:predicateObjectMap [
      rr:predicate foaf:based_near ;
      rr:objectMap [
        rr:parentTriplesMap <#AddressTripleMap> ;
        rr:joinCondition [ rr:child "addr" ; rr:parent "ID" ]
      ] ;

    ] ;
    .
```



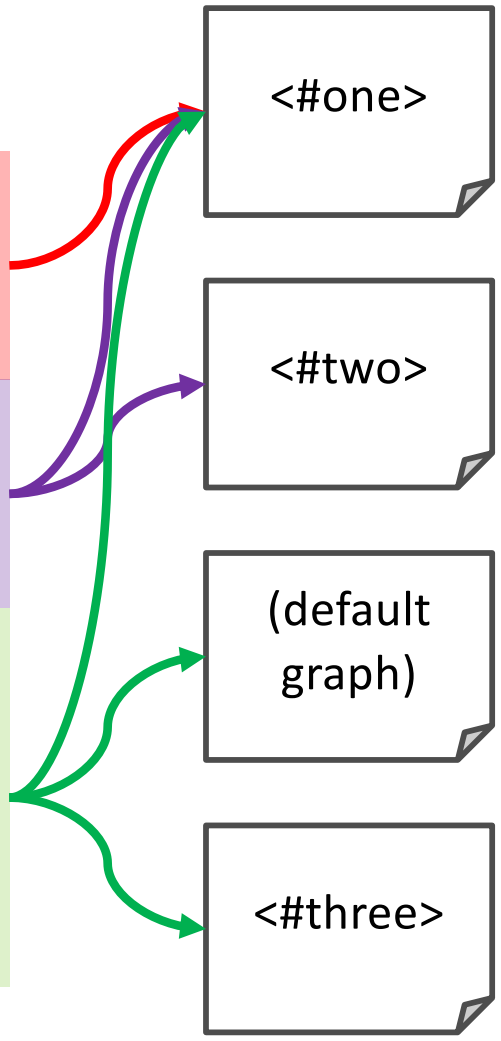
Graph Maps

```
<#PersonTriplesMap>
    a rr:TriplesMap;
    rr:logicalTable [ rr:tableName "People" ] ;
    rr:subjectMap [
      rr:template "http://foo.example/Person/{ID}" ;
      rr:class foaf:Person ;
      rr:graph <#one>
    ] ;
    rr:predicateObjectMap [
      rr:predicate foaf:name ;
      rr:objectMap [ rr:column "fname" ] ;
      rr:graph <#two>
    ] ;
    rr:predicateObjectMap [
      rr:predicate foaf:based_near ;
      rr:objectMap [
        rr:parentTriplesMap <#AddressTripleMap> ;
        rr:joinCondition [ rr:child "addr" ; rr:parent "ID" ]
      ] ;
      rr:graph <#three>
    ] ;
    .
```



Graph Maps

```
<#PersonTriplesMap>
    a rr:TriplesMap;
    rr:logicalTable [ rr:tableName "People" ] ;
    rr:subjectMap [
      rr:template "http://foo.example/Person/{ID}" ;
      rr:class foaf:Person ;
      rr:graph <#one>
    ] ;
    rr:predicateObjectMap [
      rr:predicate foaf:name ;
      rr:objectMap [ rr:column "fname" ] ;
      rr:graph <#two>
    ] ;
    rr:predicateObjectMap [
      rr:predicate foaf:based_near ;
      rr:objectMap [
        rr:parentTriplesMap <#AddressTripleMap> ;
        rr:joinCondition [ rr:child "addr" ; rr:parent "ID" ]
      ] ;
      rr:graph <#three>, rr:defaultGraph
    ] ;
    .
```



R2RML

- R2RML supports mapping values with constants, column values or column values applied to a template.
- If an objectMap does not refer to a column or has no language tag, then term types default to IRIs unless you explicitly specify it to be a Literal.

```
…
rr:predicateObjectMap [
   rr:predicateMap [ rr:constant foaf:name ] ;
      rr:objectMap [
         rr:template "{first} {last}" ;
         rr:termType rr:Literal ;
      ]
   ]
…
```

R2RML

- Term maps with a TermType of rr:Literal can have a language tag.

```
…
rr:predicateObjectMap [
   rr:predicateMap [ rr:constant rdfs:label ] ;
      rr:objectMap[
         rr:column "{title}" ;
         rr:language "en" ;
      ]
   ]
…
```

- Why is the above TermType implied to be a rr:Literal? It uses a column.

R2RML: Multiple Languages

What if you have a table with multiple languages?

Assuming you have a discriminator, you can

- Create a logical table with an SQL query for each language.
  - How?

- Create one logical table and use a language column to create a single mapping for all languages.
  - Unfortunately, this is not part of the recommendation and support depends on the implementation…

```
rr:objectMap [
   rr:column "TITLE" ;
   rrx:languageColumn "TITLE_LANG" ;
].
```

R2RML: Datatypes

Datatypes can only be declared for TermMaps that are of type rr:Literal and without a language tag.

```
rr:objectMap [
   rr:column "EMPNO" ;
   rr:datatype xsd:positiveInteger
]
```

R2RML

- R2RML provides a highly customizable language for mapping relational databases to triples.
- Unlike direct mapping that reflects the database's structure, the author of the mapping decides on the structure and ontology.
  - Mapping *projections*, e.g., a tripleMap for $\pi_{\{city\}}(Addresses)$ to create instances of the concept ex:City (cfr. "degrouping" from lecture 1)
  - Mapping *selections*, e.g., a tripleMap for $\sigma_{\{type='Cat'\}}(Pet)$ to create instances of ex:Cat
  - …

rr:inverseExpression

- Interesting for virtual graphs – retrieving RDF by translating SPARQL queries into SQL queries via a mediator.
- R2RML transforms “database terms” into RDF terms. Inverse expressions help one to transform RDF terms back into “database terms” which may increase performance.

- I am not aware of any implementation that supports or implements this, as it is not necessary for R2RML views to work.

rr:inverseExpression

R2RML: we assume surnames are stored in capital letters

```
<TriplesMap1> a rr:TriplesMap ;
    rr:logicalTable [
      rr:sqlQuery "SELECT email, LOWER(surname) AS lsur FROM emp" ; ] ;
    rr:subjectMap [
        rr:template "http://example.org/person/{lsur}" ;

    ] ;
    rr:predicateObjectMap [
        rr:predicate foaf:mbox ;
        rr:objectMap [ rr:column "email" ]
    ] .
```

The SQL query for retrieving the value of an email address should correspond with:
SELECT email FROM emp WHERE LOWER(surname) = 'debruyne'
This may be slow

SPARQL:

```
SELECT ?email {
  <http://example.org/person/debruyne> foaf:mbox ?email .
}
```

rr:inverseExpression

R2RML: we assume surnames are stored in capital letters

```
<TriplesMap1> a rr:TriplesMap ;
    rr:logicalTable [
      rr:sqlQuery "SELECT email, LOWER(surname) AS lsur FROM emp" ; ] ;
    rr:subjectMap [
        rr:template "http://example.org/person/{lsur}" ;
        rr:inverseExpression "{surname} = UPPER({lsur})";
    ] ;
    rr:predicateObjectMap [
        rr:predicate foaf:mbox ;
        rr:objectMap [ rr:column "email" ]
    ] .
```

We can now optimize the query to
SELECT email FROM emp
WHERE surname = UPPER('debruyne')
If surname were to be indexed, this query would be much faster!

SPARQL:

```
SELECT ?email {
  <http://example.org/person/debruyne> foaf:mbox ?email .
}
```

RDB2RDF

- Translate data to RDF: generates an RDF dump for immediate consumption, but will be harder to maintain (A)
  - Loaded into triplestores such as Jena TDB (with Fuseki for a SPARQL endpoint), or Virtuoso.
- A mapping from the RDB to RDF, generating SPARQL queries into (intermediate) SQL queries, but will have longer query times (B)

- The TA will demonstrate how you can use OnTop to mediate between ontologies and a database via mappings.

R2RML-F

- Introduction of a *Function Valued* Term Map that allow for user defined functions at the cost of tractability.

- C. Debruyne and D. O'Sullivan. R2RML-F: Towards Sharing and Executing Domain Logic in R2RML Mappings. In Proceedings of the Workshop on Linked Data on the Web, LDOW 2016, co-located with the 25th International World Wide Web Conference (WWW 2016), Montreal, Canada, April 12th, 2016, 2016.

- https://github.com/chrdebru/r2rml

Extending R2RML

- Namespace rrf: http://kdeg.scss.tcd.ie/ns/rrf#

- *Functions* have a function *name* and *body*.

```
<#Multiply>
 rrf:functionName "multiply" ;
 rrf:functionBody """
  function multiply(var1, var2) {
   return var1 * var2 ;
  }
 """ ;
.
```

- Functions are written in ECMAScript.

Extending R2RML

A "function valued" term map *calls a function* and the *parameters* are themselves term maps.

```
<#TriplesMap1>
 rr:logicalTable [ rr:tableName "Employee"; ];
 rr:subjectMap [ rr:template "http://org.com/employee/{ID}"; ] ;
 rr:predicateObjectMap [
  rr:predicate ex:salary ;
  rr:objectMap [
   rr:datatype xsd:double ;
   rrf:functionCall [
    rrf:function <#Multiply> ;
    rrf:parameterBindings (
     [ rr:constant "12"^^xsd:integer ]
     [ rr:column "monthly_salary" ]
    ) ;
   ] ;
  ] ;
```

Parameter bindings as an **RDF Collection**.

Parameter bindings can be empty.

Term Maps as parameters.

RML

- Developed by imec at UGent
  - http://semweb.mmlab.be/rml/spec.html
  - https://github.com/mmlab/RMLProcessor
- An extension of R2RML to support XML, HTML, CSV, JSON…
  - "Basically" a superset of R2RML.

- See also: A. Dimou, M. Vander Sande, P. Colpaert, R. Verborgh, E. Mannens, R. Van De Walle: "RML: A Generic Language for Integrated RDF Mappings of Heterogeneous Data", in Proceedings of the 7th Workshop on Linked Data on the Web, WWW14, 2014 Seoul, Korea.

Other tools and mapping languages

- Stardog -- http://stardog.com/
- D2RQ -- http://d2rq.org/
- Virtuoso -- http://virtuoso.openlinksw.com/

An interactive demonstration

Weatherstations.csv – Weather Readings omitted in this slide.

Name	Weather_Reading	Agency	LAT	LONG
M50 Blanchardstown	…	National Roads Authority	53.37046603	-6.380851447
M50 Dublin Airport	…	National Roads Authority	53.40964111	-6.227597428
Dublin Airport	…	Met …ireann	53.42150608	-6.29784754

Using the GeoSPARQL, we will create instances of geo:Feature (the weather stations) and their instances of geo:Geometry (a point using the latitude and longitude).

References

- Satya S. Sahoo et al. A Survey of Current Approaches for Mapping of Relational Databases to RDF. W3C RDB2RDF XG Report, W3C, 2009. http://www.w3.org/2005/Incubator/rdb2rdf/RDB2RDF_SurveyReport.pdf
- Berners-Lee, T. Relational Databases on the Semantic Web, 1998, via http://www.w3.org/DesignIssues/RDB-RDF.html
- R. Cyganiak, C. Bizer, J. Garbers, O. Maresch, and C. Becker. The D2RQ mapping language. http://d2rq.org/d2rq-language, March 2012.
- E. Prud'hommeaux and A. Seaborne. SPARQL Query Language for RDF. W3C Recommendation, W3C, January 2008.