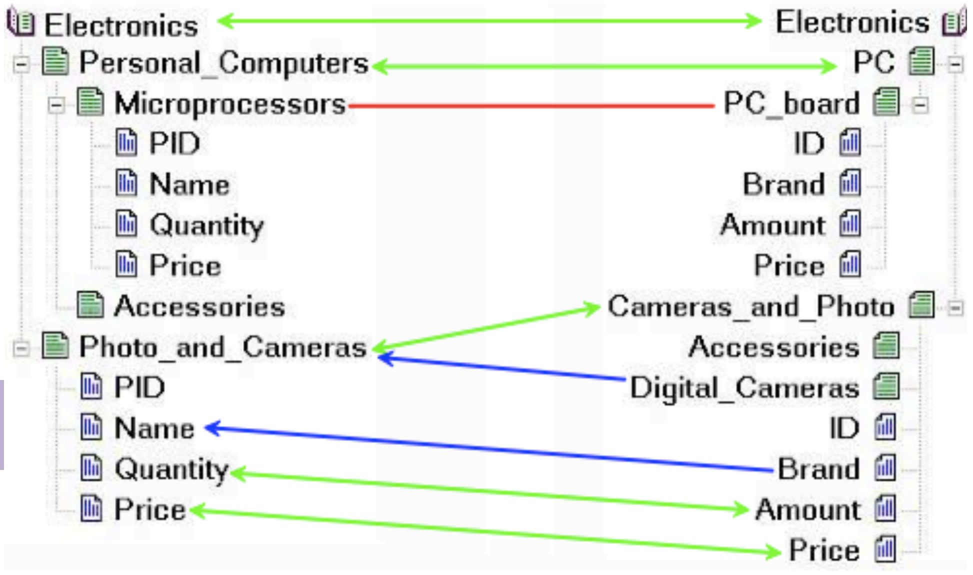
Student Online Teaching Advice Notice

UPLIFT Mapping Overview

Course Notes from Dr. Christophe Debruyne

Terminology

- **Correspondences vs. Mappings**
  - **Correspondences** capture how entities are related
  - **Mappings** describe how to relate entities
  - Correspondences are 'symmetrical', mappings have a direction
  - *Matching vs. Mapping*
- **Different types of mappings**
  - Between ontologies
  - Between datasets or instances
  - **Uplift**: From non-RDF to RDF
  - **Downlift**: from RDF to non-RDF



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.

Slide source: Christophe Debruyne

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.”**

Slide source: Christophe Debruyne

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.
  - Record 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

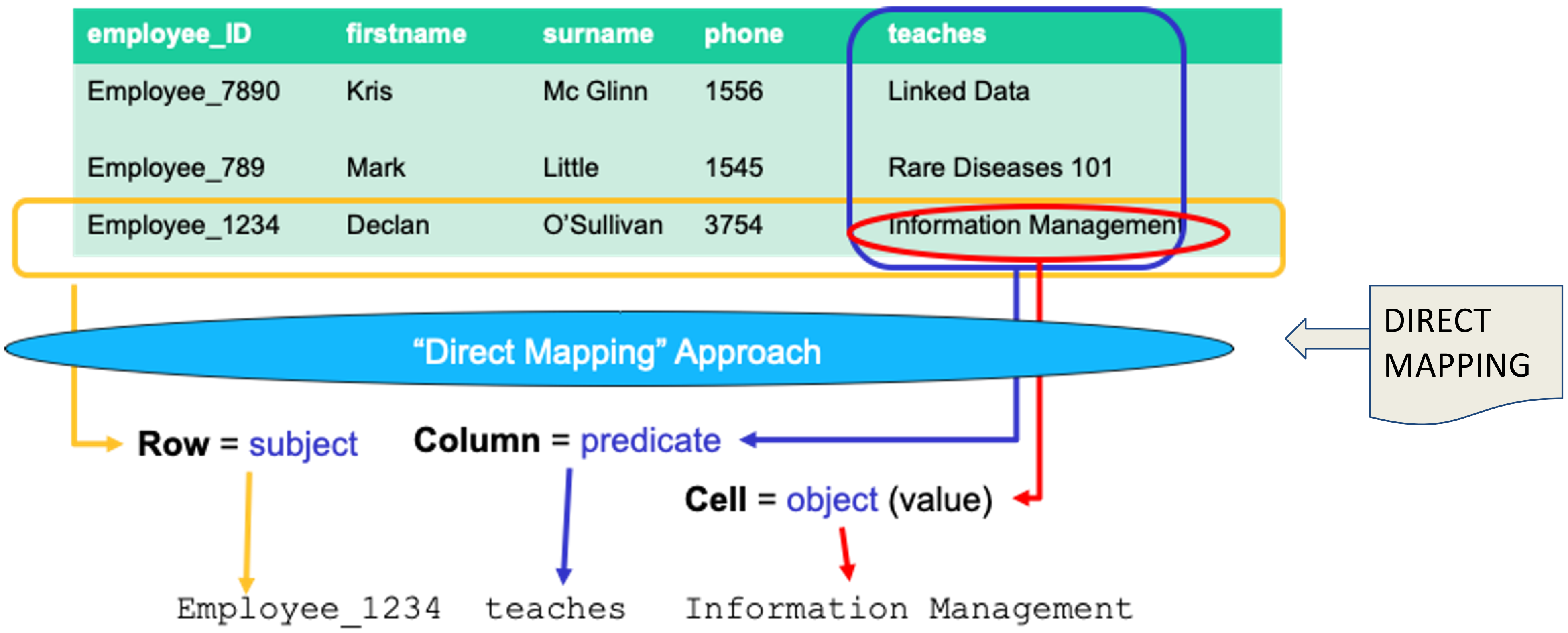
People		
PK		*→Addresses(ID)*
ID	**fname**	**addr**
1	Christophe	1
2	Kevin	NULL

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> rdf:type <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> rdf:type <People> .
<People/ID=2> <People#ID> 2 .
<People/ID=2> <People#fname> "Kevin" .

<Addresses/ID=1> rdf:type <Addresses> .
<Addresses/ID=1> <Addresses#ID> 1 .
<Addresses/ID=1> <Addresses#city> ”Brussels" .
```

Slide source: Christophe Debruyne



employee_ID	firstname	surname	phone	teaches
Employee_7890	Kris	Mc Glinn	1556	Linked Data
Employee_789	Mark	Little	1545	Rare Diseases 101
Employee_1234	Declan	O'Sullivan	3754	Information Management

"Direct Mapping" Approach
DIRECT MAPPING
Row = subject
Column = predicate
Cell = object (value)
Employee_1234 teaches Information Management

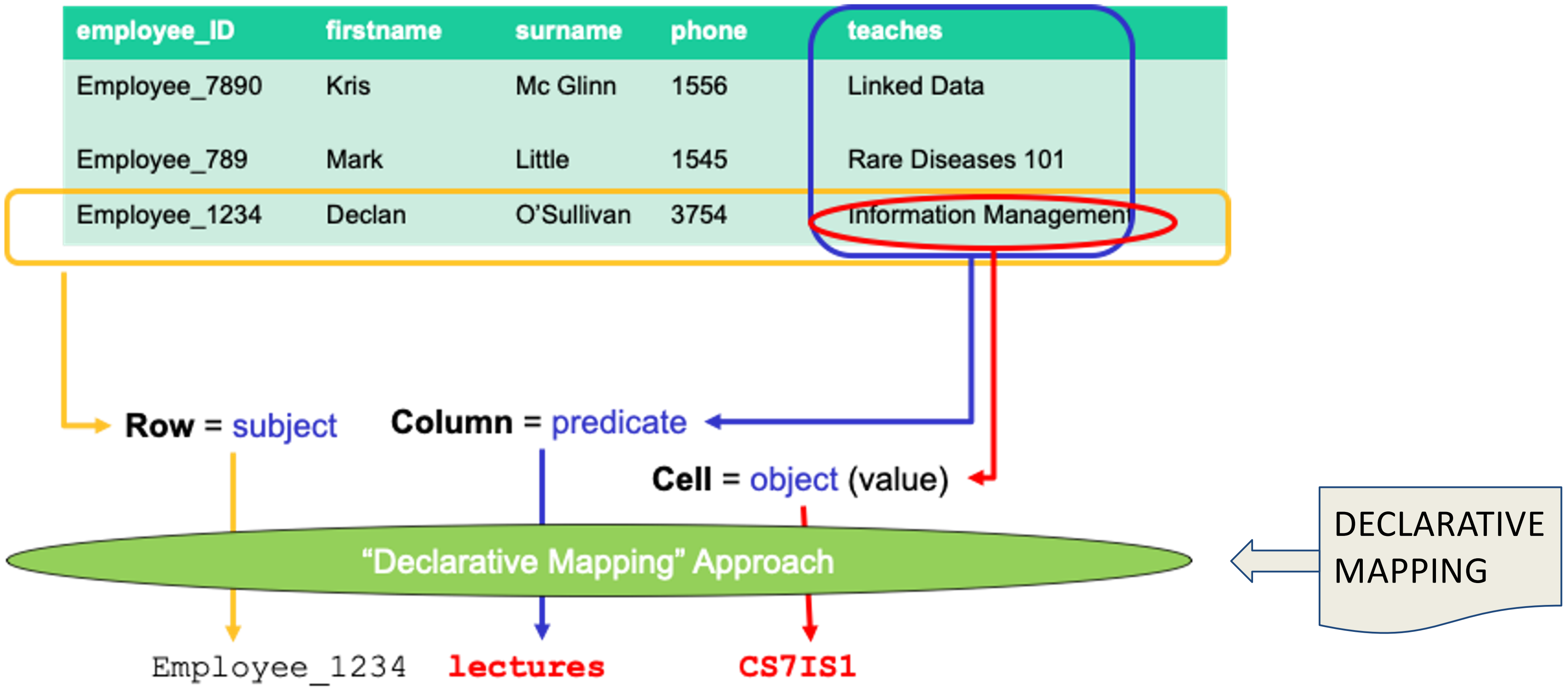
.. and so on Employee_1234 firstname Declan, Employee_1234 surname O'Sullivan, etc. for every key/column combination

By default the Direct Mapping approach uplifts all the data available into the graph without much control….

Direct Mappings

Small discussion:

Simple approach but why not always appropriate?



employee_ID	firstname	surname	phone	teaches
Employee_7890	Kris	Mc Glinn	1556	Linked Data
Employee_789	Mark	Little	1545	Rare Diseases 101
Employee_1234	Declan	O'Sullivan	3754	Information Management

.. and in this case we decide only to uplift the data from column' teaches' and not from the other columns....

Declarative Mapping provides very detailed control on uplift process, as to how information appears in graph

Declarative Approach: R2RML

- R2RML: RDB to RDF Mapping Language
  - A W3C Recommendation since Autumn 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 specifies
  - A vocabulary to specify those mappings;
  - How those mappings should be interpreted to produce RDF.
  - R2RML files are thus stored as RDF.

A Triples Map

A triples map has two parts:

- a subject map
- several predicate-object maps
  (combining predicate and object maps).

Input of a map:

- a row of the logical table

Output of a map: for each row,

- a subject resource (IRI or blank node),
  often generated from primary key values
- several triples with the same subject,
  but varying predicates and objects,
  generated from the attributes of the row

Slide source: Mariano Rodriguez

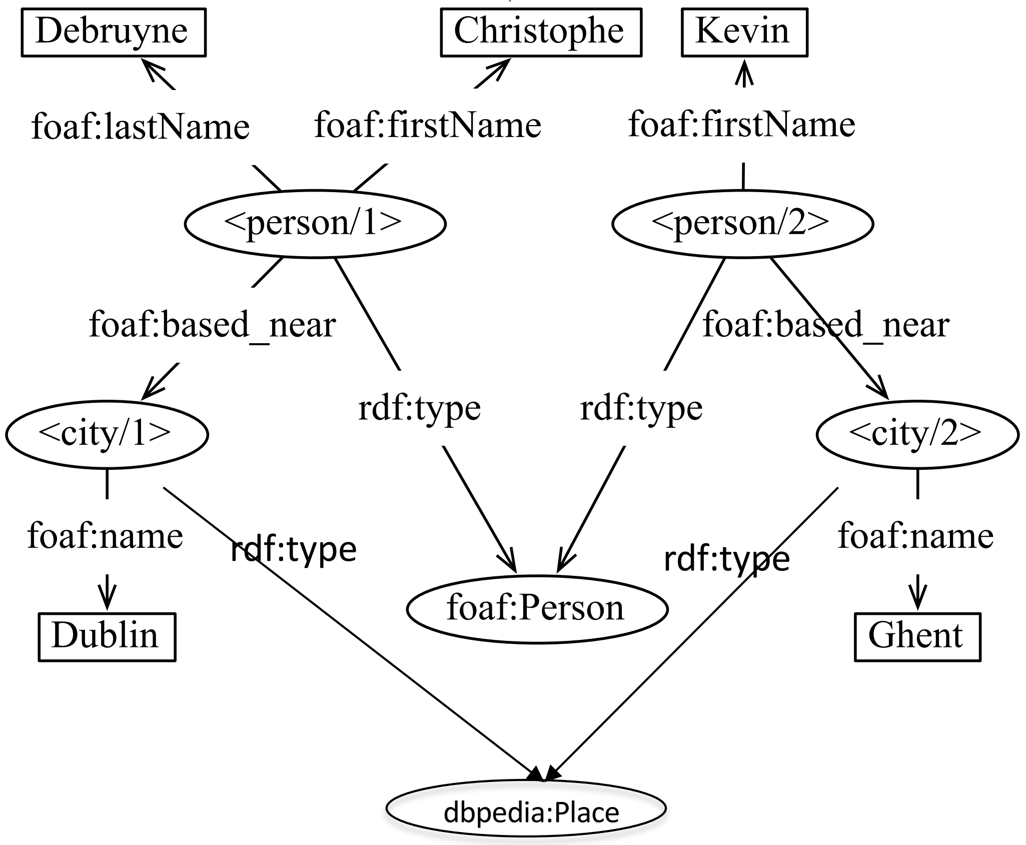
From RDB/Tabular to RDF

Person

<u>ID</u>	First	Last	CityID
1	Christophe	Debruyne	1
2	Kevin	NULL	2

City

<u>ID</u>	Name
1	Dublin
2	Ghent



Slide source: Christophe Debruyne

Example

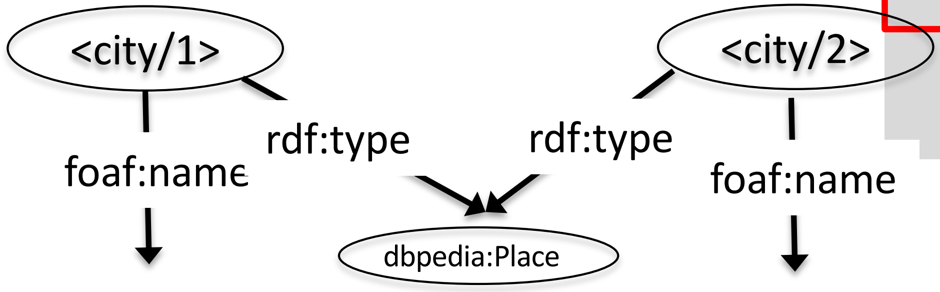
CITY	
ID	**NAME**
1	Dublin
2	Ghent

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

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

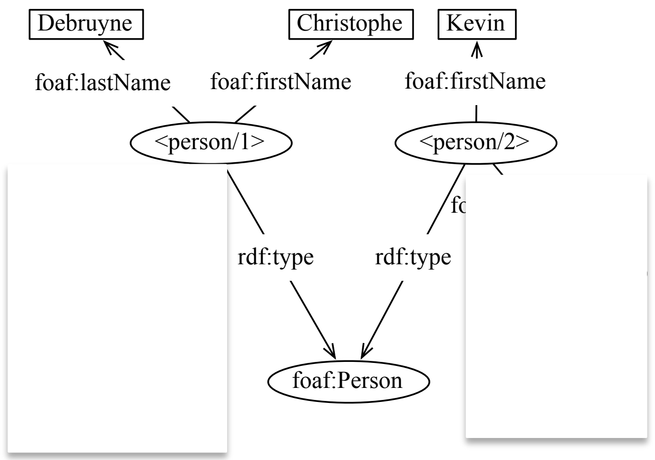
How to generate predicates and objects.

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

<#CityTriplesMap>
        a rr:TriplesMap ;

        rr:logicalTable [ rr:tableName ”CITY" ] ;

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

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



Example

PERSON			
ID	**FIRST**	**LAST**	**CITYID**
1	Christophe	Debruyne	1
2	Kevin	NULL	2

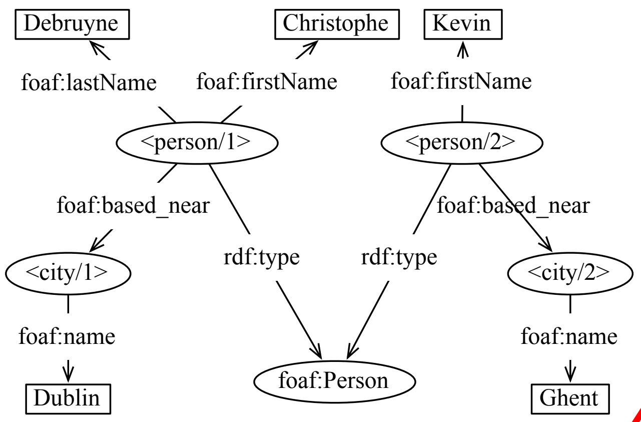


```
<#PersonTriplesMap>
        a rr:TriplesMap;
        rr:logicalTable [ rr:tableName ”PERSON" ] ;
        rr:subjectMap [
                rr:template "http://foo.example/Person/{ID}" ;
                rr:class foaf:Person ;
        ];
        rr:predicateObjectMap [
                rr:predicate foaf:name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

        .
```

Example

Person			
ID	**FIRST**	**LAST**	**CITYID**
1	Christophe	Debruyne	1
2	Kevin	NULL	2

City	
ID	**NAME**
1	Dublin
2	Ghent



Relating People to Addresses.

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

- Note:
  - child = referencing map
  - parent = referenced map

R2RML: Running Example: SQL version

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" ]
     ] ;
     .
```

Slide source: Christophe Debruyne

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**

```
<#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}" ;
       ]
     ] ;
     .
```

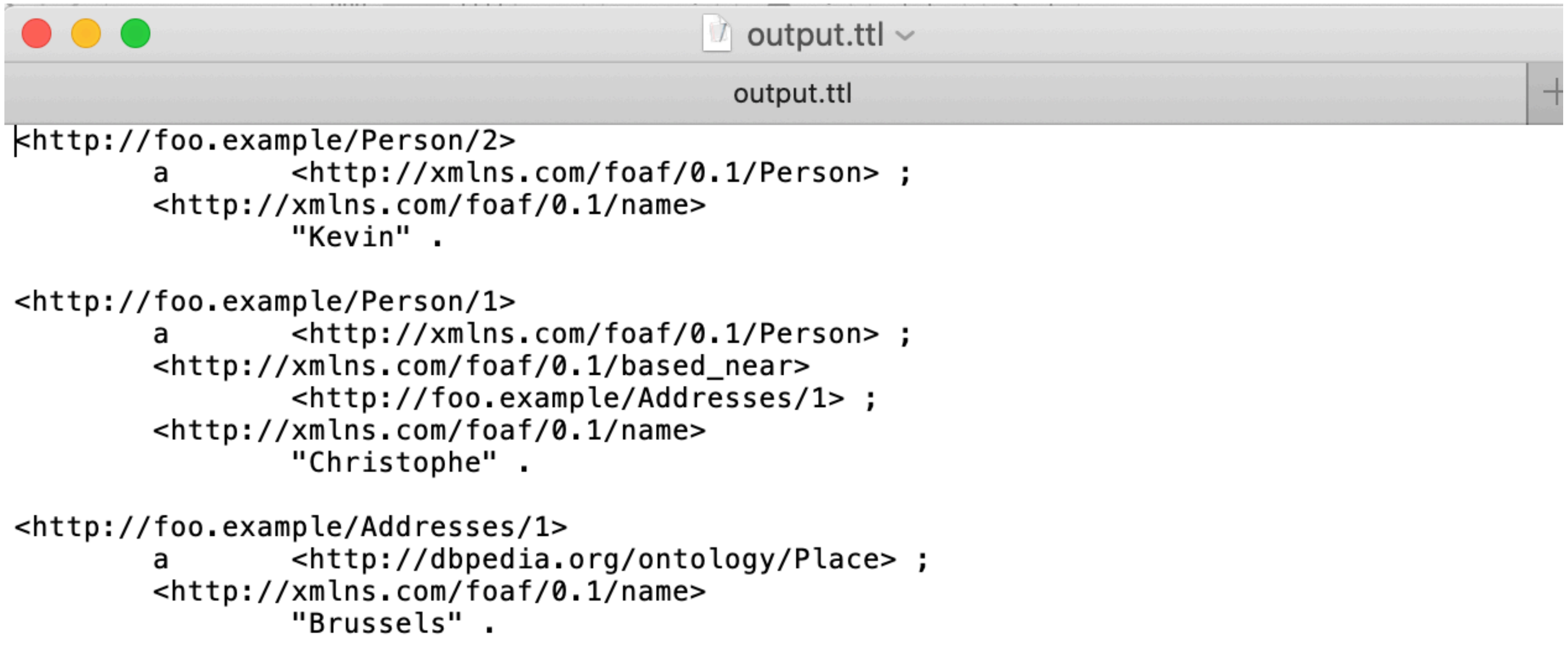
Create a Term Map that generates the same resource!
Prone to errors if, for instance, templates change, though.

Slide source: Christophe Debruyne

TryItOut

1. Download ‘TryItOut Uplift.zip’ file from Resources on Blackboard
2. Unzip
3. The Java Jar file for R2RML engine is in folder ‘r2rml’
4. Go into folder ‘Lecturedata’
5. Have a look at ‘csv-config.properties’ file to see what name of input files and output file being specified, and the mapping file name
6. Run ‘java -jar ../r2rml/r2rml.jar csv-config.properties’ from this folder
7. Have a look at the generated triples in **output.ttl**

TryItOut: Should see something like this in output file



```
<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" .
```

Slide source: Christophe Debruyne

TryItOut: Self Directed Task 8

- Part A
  - In this part you will create the **MetEireannStationDetails.ttl** file you used in the SPARQL TryItOut
  - From the download earlier in the lecture
  - Go into the folder named ‘Tutorial-PartA’
  - Run ‘java -jar ../r2rml/r2rml.jar Map_MetEireannStationsDetails.properties’
  - Have a look at the generated **MetEireannStationDetails.ttl**

- Part B
  - This is a slightly more expanded version of Part A
  - Go into the folder named ‘Tutorial-PartB’
  - Follow the instructions in the README
  - Use the R2RML engine you already downloaded and have used

For Portfolio
Self-Directed Exercise Task 9 (SDT9) – before Monday 09 November

- Create a **CSV** with the following column headings
- ArtistName, CountryOfOrigin, TitleOfTopWork, DateProduced
  - Artist can be a Singer, Band, Actor, Artist, Director etc.
  - TopWork could be a movie, art work, song etc. depending on the Artist
- Put in at least 20 rows of unique data (see example below)
- Create a R2RML mapping to uplift the data into RDF
- Consider using FOAF for vocab for some of your predicates or search LOV (https://lov.linkeddata.es/dataset/lov/) for appropriate vocab term for some of your predicates
- Post the R2RML mapping and the CSV into your portfolio

ArtistName	CountryOfOrigin	TitleOfTopWork	DateProduced
U2	Ireland	Pride	1984
Leonardo Da Vinci	Italy	Mona Lisa	1503
Michael Buble	Canada	Always on My Mind	2007
Taika Waititi	New Zealand	Hunt for the Wilderpeople	2016

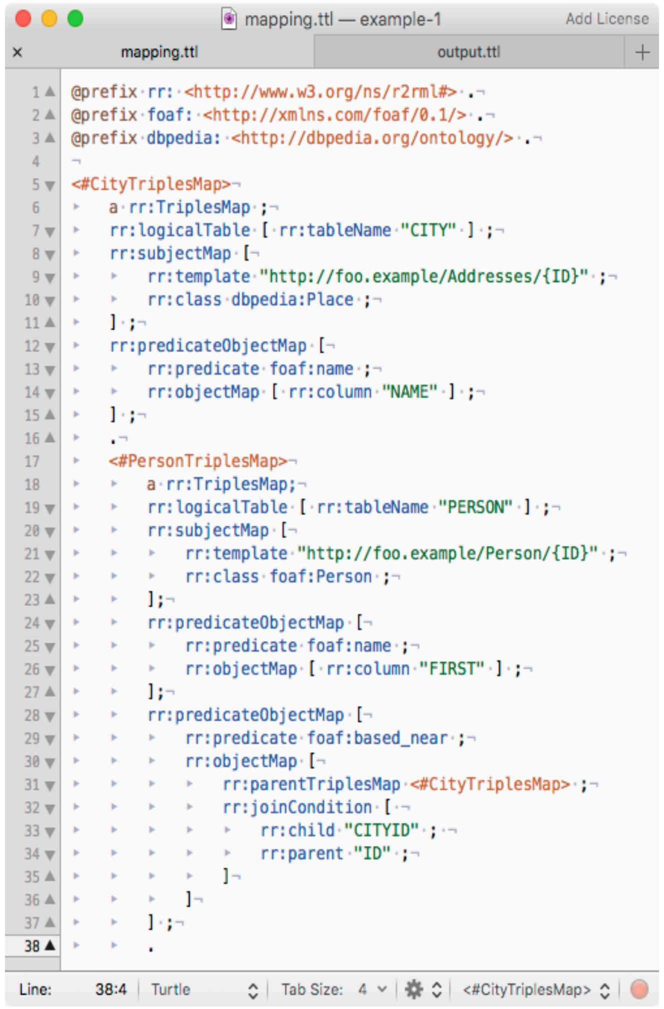
Self Directed Task 10: Juma Uplift – Sign up

- Go to
  - http://juma.adaptcentre.ie/juma-editor/
  - Click on sign up to create an account
- We will use this tool on Thursday during the session

R2RML engine implementation used

R2RML engine implementation
https://github.com/chrdebru/r2rml

Works with RDB and CSV data



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

<#CityTriplesMap>
    a rr:TriplesMap ;
    rr:logicalTable [ rr:tableName "CITY" ] ;
    rr:subjectMap [
        rr:template "http://foo.example/Addresses/{ID}" ;
        rr:class dbpedia:Place ;
    ] ;
    rr:predicateObjectMap [
        rr:predicate foaf:name ;
        rr:objectMap [ rr:column "NAME" ] ;
    ] ;
    .
    <#PersonTriplesMap>
        a rr:TriplesMap;
        rr:logicalTable [ rr:tableName "PERSON" ] ;
        rr:subjectMap [
            rr:template "http://foo.example/Person/{ID}" ;
            rr:class foaf:Person ;
        ];
        rr:predicateObjectMap [
            rr:predicate foaf:name ;
            rr:objectMap [ rr:column "FIRST" ] ;
        ];
        rr:predicateObjectMap [
            rr:predicate foaf:based_near ;
            rr:objectMap [
                rr:parentTriplesMap <#CityTriplesMap> ;
                rr:joinCondition [
                    rr:child "CITYID" ;
                    rr:parent "ID" ;
                ]
            ]
        ] ;
        .
```

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.

Slide source: Christophe Debruyne

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.

Slide source: Christophe Debruyne

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.

Other Serialisations

- Why have other serialisations?
  - Innovation?
  - Make easier for non-specialist Knowledge Engineers?

RML

- Developed by imec at UGent
  - http://semweb.mmlab.be/rml/spec.html
  - https://github.com/mmlab/RMLProcessor
- Extends R2RML approach to support mapping XML, HTML, CSV, JSON data into RDF
  - “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.

SML - Sparqlification Mapping Language (SML)

- SML is based on SQL CREATE VIEWS and SPARQL CONSTRUCT queries.
- **Construct.** Consists of triple patterns, similar to SPARQL CONSTRUCT queries. These are used as templates for the construction of the RDF triples.
- **With.** This clause is used to specify variables whose values are RDF terms from rows of the logical table. These variables may be used in the construct clause to form RDF triples.
- **From.** Define the logical table. As in R2RML, it may be a table, view, or a SQL query.

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

Create View view1 As
 Construct {
    ?s1  a foaf:Person.
    ?s1   foaf:name    ?o1. }
 With
  ?s1 = uri(concat('http://example.org/', ?id))
  ?o1 = plainLiteral(?name)
 From
  Person
```

Stadler, C., Unbehauen, J., Westphal, P., Sherif, M. A., & Lehmann, J. (2015). Simplified RDB2RDF Mapping. In *Proceedings of the Workshop on Linked Data on the Web, LDOW 2015, co-located with the 24th International World Wide Web Conference (WWW 2015)*

Existing Relational Database data and RDF

- Two approaches
  - Translate relational database data to RDF: generate an RDF dump for immediate consumption, but will need a process to maintain data replication (A)
  - Sit a translation layer on top of RDB, translating SPARQL queries into (intermediate) SQL queries and results in RDF
    - but will have longer query times (B)

Slide source: Christophe Debruyne

R2RML enabled triplestore implementations

- Stardog -- http://stardog.com/
  - implements both R2RML and their own mapping language.
- Virtuoso – http://virtuoso.openlinksw.com/
- Oracle Spatial and Graph – https://www.oracle.com/database/spatial/index.html
  - Support for both transforming relational databases into RDF using R2RML as well as creating R2RML views using those mappings
- OnTop -- http://ontop.inf.unibz.it/
  - Access relational databases as virtual graphs using mappings

Slide source: Christophe Debruyne

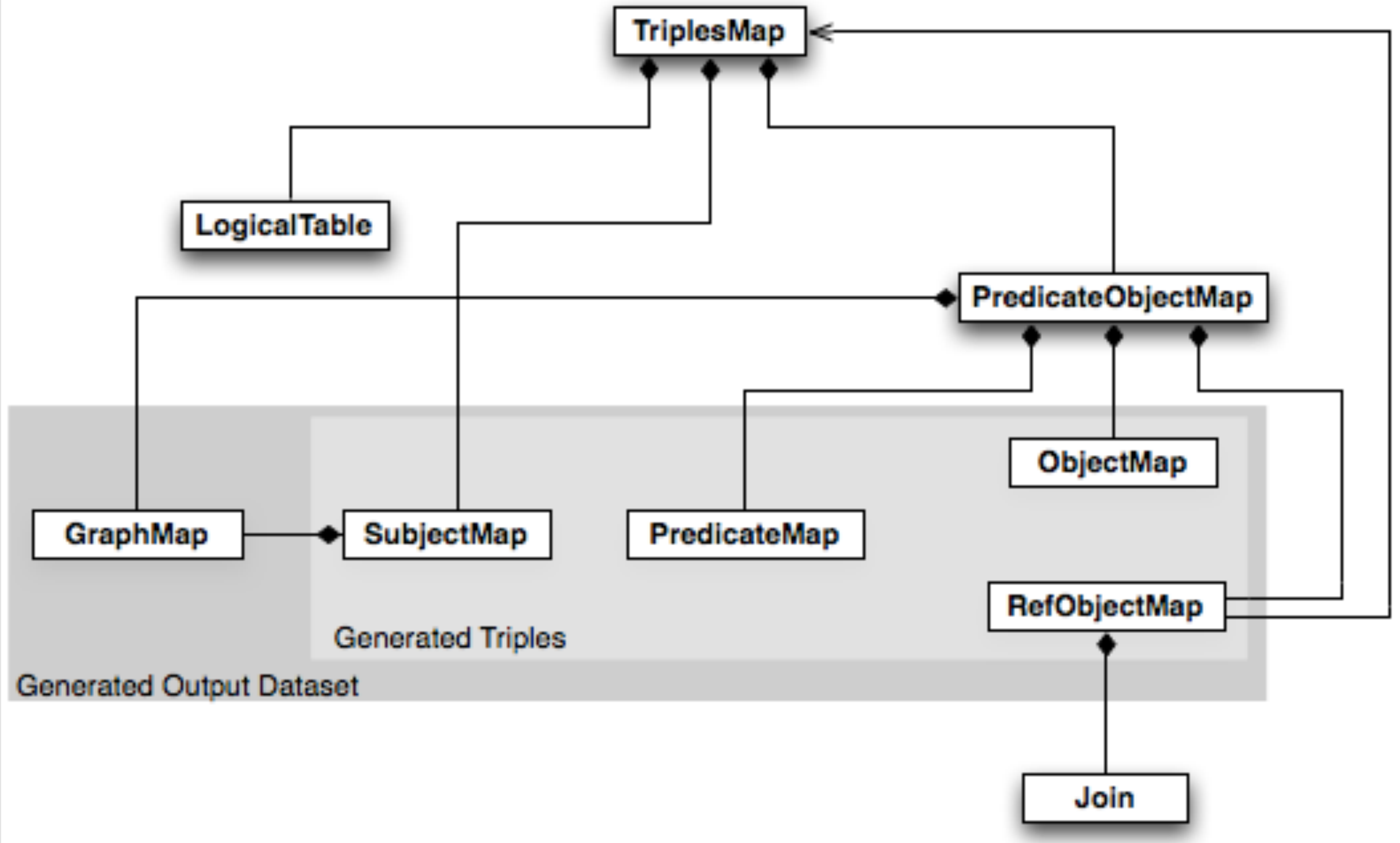
References

- 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.
- W3C R2RML Specification https://www.w3.org/TR/r2rml/

R2RML Further Detail

https://www.w3.org/TR/r2rml/

R2RML



Slide source: Christophe Debruyne

Subject Map

- A subject map is a **term map**. It specifies a rule for generating the subjects of the RDF triples generated by a triples map.
- **Term maps** are used to generate the subjects, predicates and objects of the RDF triples that are generated by a triples map. Consequently, there are several kinds of term maps, depending on where in the mapping they occur: subject maps, predicate maps, object maps and graph maps.
- A **term map** must be exactly one of the following:
  - a constant-valued term map,
  - a column-valued term map,
  - a template-valued term map.

Term Maps

A ***constant-valued term map*** is a term map that ignores the logical table row and always generates the same RDF term. A constant-valued term map is represented by a resource that has exactly one `rr:constant` property.

A ***column-valued term map*** is a term map that is represented by a resource that has exactly one rr:column property.
The value of the rr:column property must be a valid column name.
The ***column value*** of the term map is the data value of that column in a given logical table row.

A ***template-valued term map*** is a term map that is represented by a resource that has exactly one rr:template property. The value of the rr:template property must be a valid string template.
A ***string template*** is a format string that can be used to build strings from multiple components. It can reference column names by enclosing them in curly braces ("{" and "}").

Predicate and Object Maps

A predicate-object map is represented by a resource that references the following other resources:

- **One or more predicate maps**. Each of them may be specified in one of two ways:
  - using the **rr:predicateMap** property, whose value must be a **predicate map**, or
  - using the **constant shortcut property** rr:predicate.
- **One or more object maps** or **referencing object maps**. Each of them may be specified in one of two ways:
  - using the **rr:objectMap** property, whose value must be either an **object map**, or a **referencing object map**.
  - using the **constant shortcut property** rr:object.

A **predicate map** is a term map.

An **object map** is a term map.

Slide source: Mariano Rodriguez

Referencing Object Maps

Example R2RML mapping

```
[] rr:predicateObjectMap [
    rr:predicate ex:department;
    rr:objectMap [
        rr:parentTriplesMap <#TriplesMap2>;
        rr:joinCondition [
            rr:child "DEPTNO";
            rr:parent "DEPTNO";
        ];
    ];
].
```

A **referencing object map** allows using the subjects of another triples map as the objects generated by a predicate-object map. Since both triples maps may be based on different logical tables, this may require a join between the logical tables. This is not restricted to 1:1 joins.

A **referencing object map** is represented by a resource that:

- has exactly one **rr:parentTriplesMap** property, whose value must be a triples map, known as the referencing object map's parent triples map.
- may have one or more **rr:joinCondition** properties, whose values must be join conditions.

A **join condition** is represented by a resource that has exactly one value for each of the following two properties:

- **rr:child**, whose value is known as the join condition's child column and must be a column name that exists in the logical table of the triples map that contains the referencing object map
- **rr:parent**, whose value is known as the join condition's parent column and must be a column name that exists in the logical table of the referencing object map's parent triples map.

Graph Maps

- By default, all RDF triples are in the default graph of the output dataset.
- A TriplesMap can contain Graph Maps that place some or all of the triples into Named Graphs instead.
- 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.

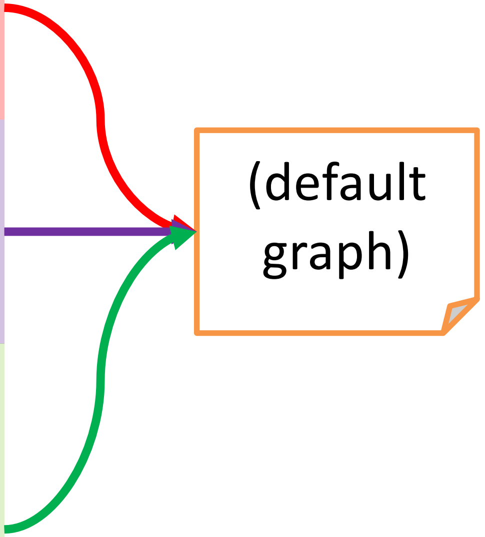
- If a Subject Map has no Graph Maps then the set of Graph Maps is {rr:defaultGraph}.

- 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

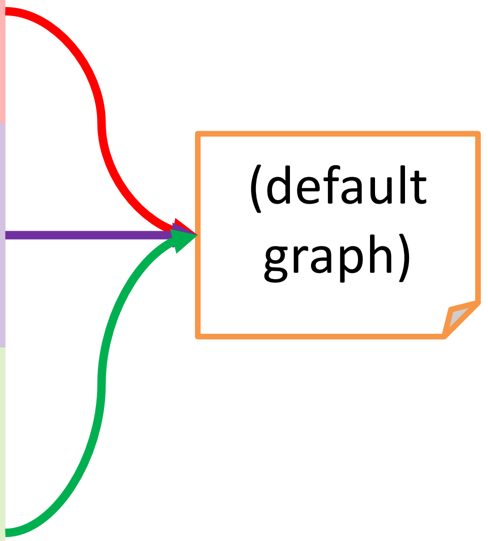
```
<#PersonTriplesMap>
     a rr:TriplesMap;
     rr:logicalTable [ rr:tableName "People" ] ;
     rr:subjectMap [
       rr:template "http://foo.example/Person/{ID}" ;
       rr:class foaf:Person ;

     ] ;
     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" ]
       ] ;

     ] ;
     .
```



Slide source: Christophe Debruyne

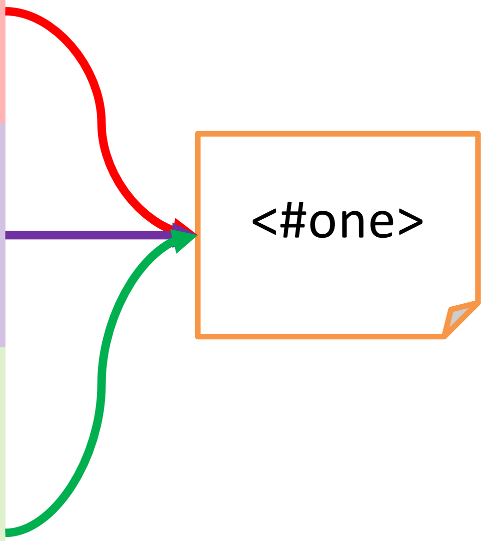
Graph Maps

```
<#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

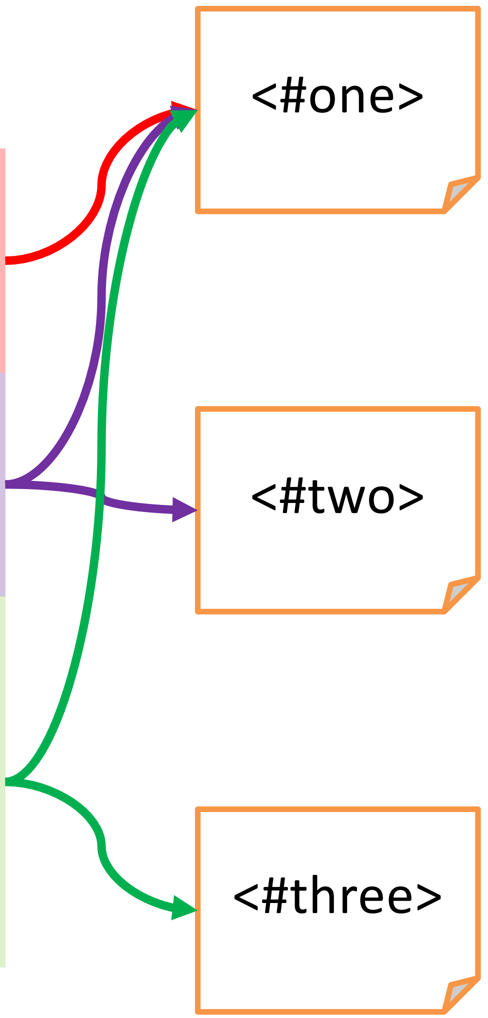
```
<#PersonTriplesMap>
     a rr:TriplesMap;
     rr:logicalTable [ rr:tableName "People" ] ;
     rr:subjectMap [
       rr:template "http://foo.example/Person/{ID}" ;
       rr:class foaf:Person ;
       rr:graph <#one>
     ] ;
     rr:predicateObjectMap [
       rr:predicate foaf:name ;
       rr:objectMap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

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

     ] ;
     .
```



Slide source: Christophe Debruyne

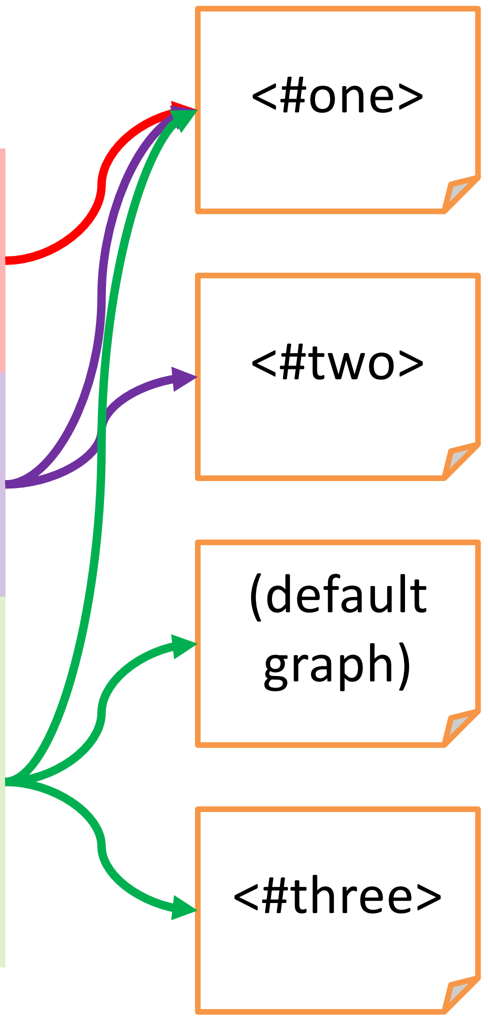
Graph Maps

```
<#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
     ] ;
     .
```



Slide source: Christophe Debruyne

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.
- W3C R2RML Specification https://www.w3.org/TR/r2rml/