RDB2RDF

Jose Luis Ambite

based on a tutorial by

Juan Sequeda Capsenta TEXAS

Barry Norton ontotext

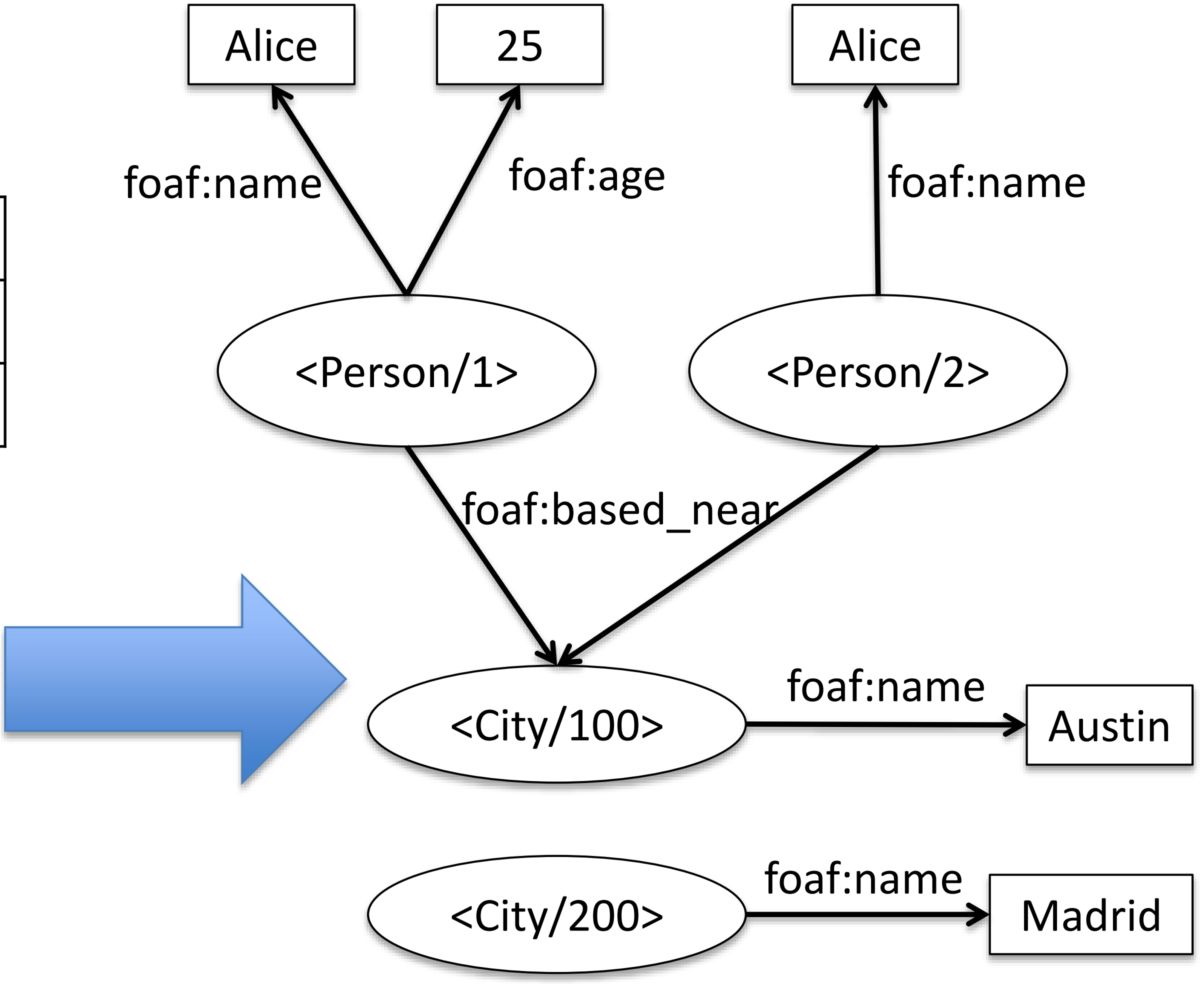
RDB2RDF: Mapping relational DB to RDF

Person

ID	NAME	AGE	CID
1	Alice	25	100
2	Bob	NULL	100

City

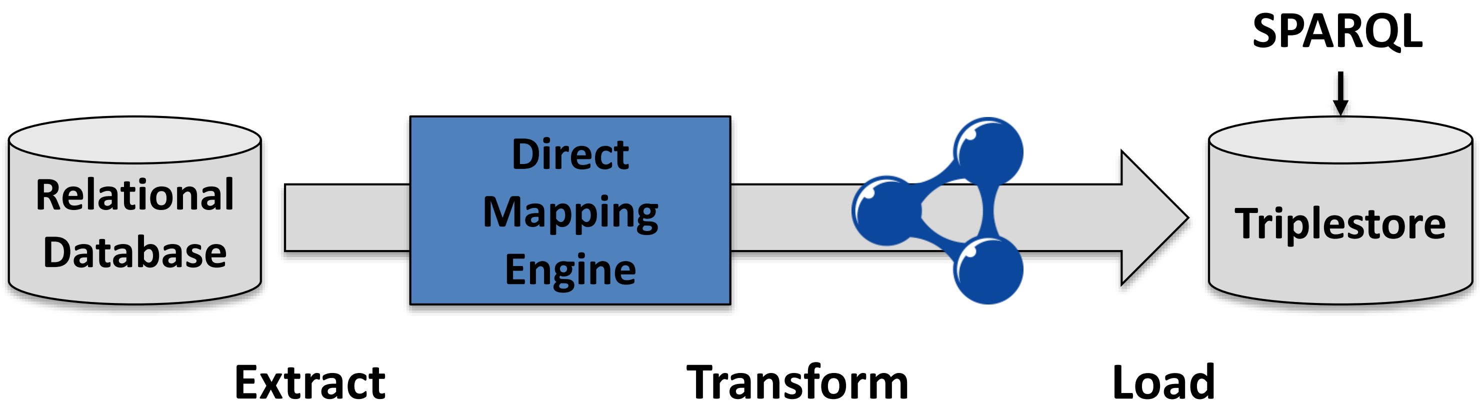
CID	NAME
100	Austin
200	Madrid



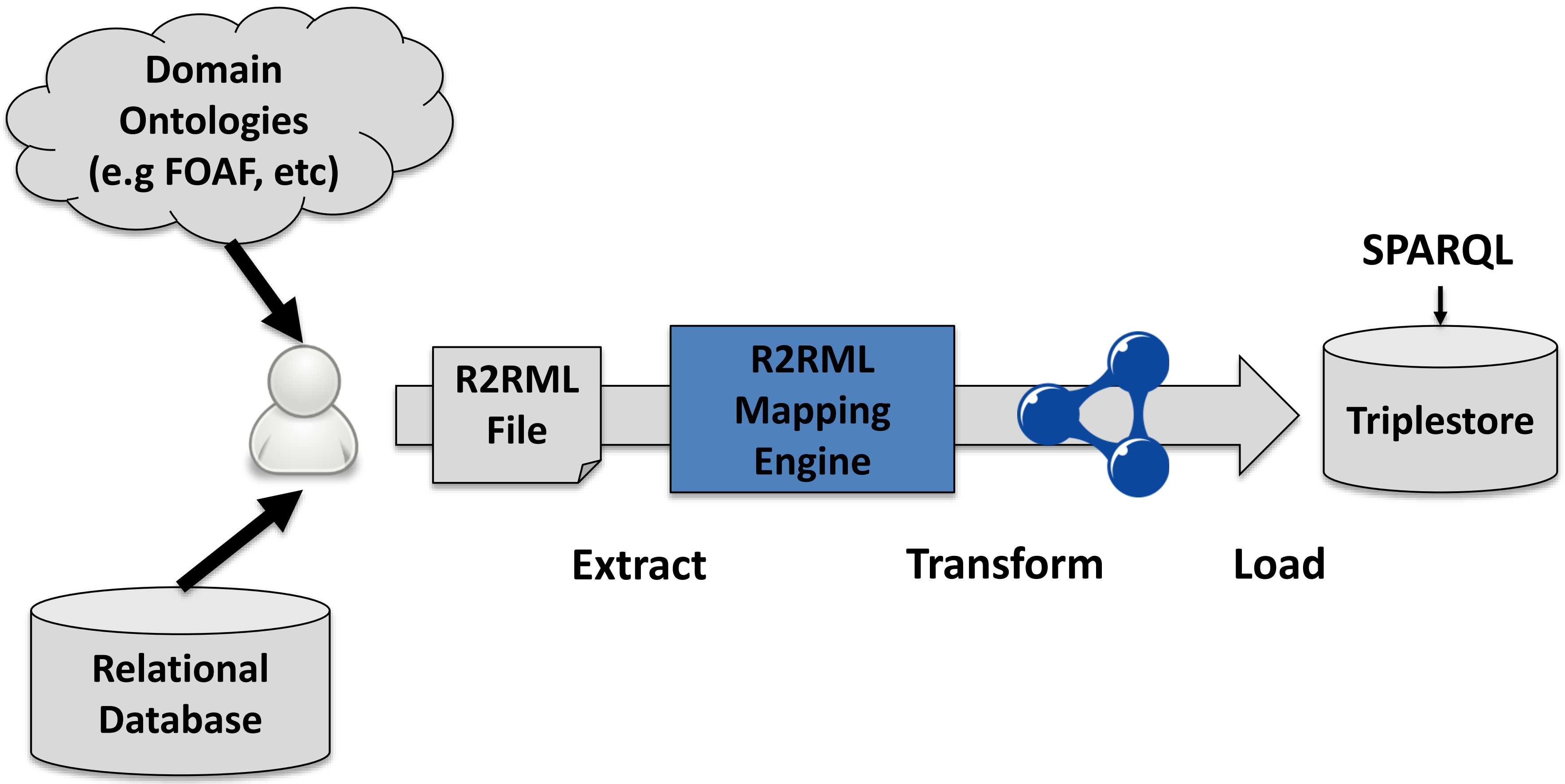
W3C RDB2RDF Standards

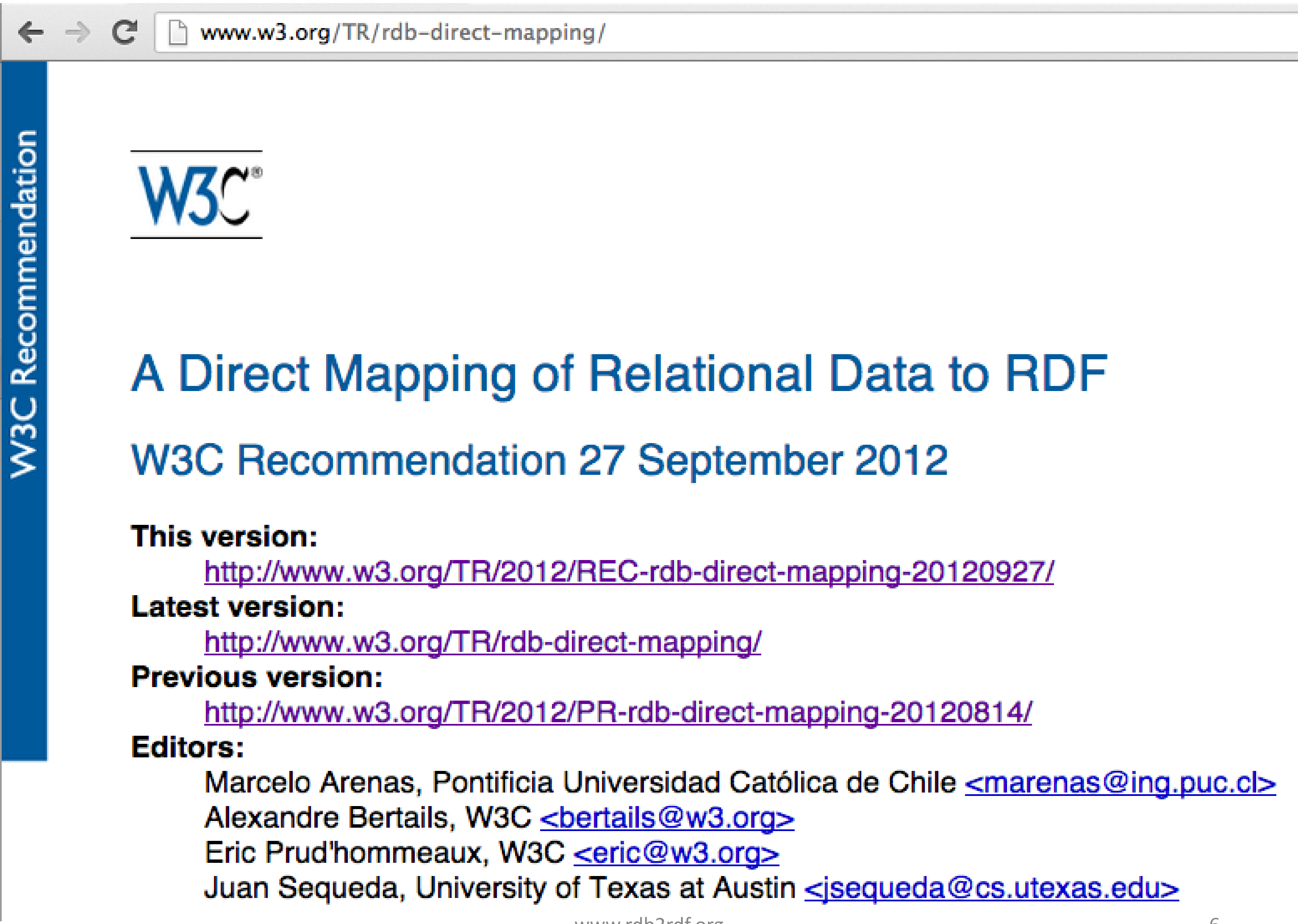
- Standards to map relational data to RDF
- A Direct Mapping of Relational Data to RDF
  - Default automatic mapping of relational data to RDF
- R2RML: RDB to RDF Mapping Language
  - Customizable language to map relational data to RDF

Direct Mapping



R2RML


Domain Ontologies (e.g FOAF, etc)
R2RML File
R2RML Mapping Engine
SPARQL
Triplestore
Extract
Transform
Load
Relational Database



www.w3.org/TR/rdb-direct-mapping/

W3C Recommendation

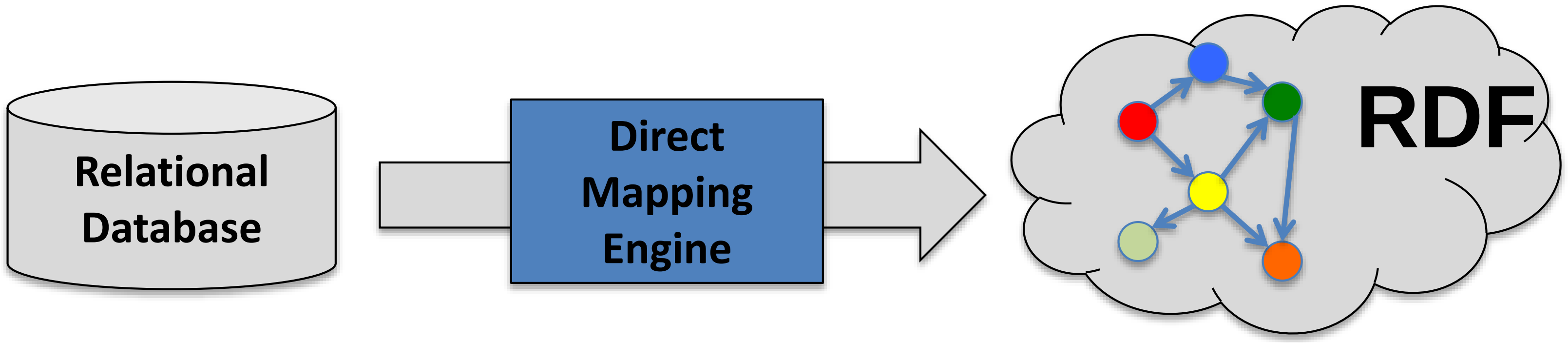
A Direct Mapping of Relational Data to RDF

W3C Recommendation 27 September 2012

This version:
http://www.w3.org/TR/2012/REC-rdb-direct-mapping-20120927/
Latest version:
http://www.w3.org/TR/rdb-direct-mapping/
Previous version:
http://www.w3.org/TR/2012/PR-rdb-direct-mapping-20120814/
Editors:

Marcelo Arenas, Pontificia Universidad Católica de Chile <marenas@ing.puc.cl>
Alexandre Bertails, W3C <bertails@w3.org>
Eric Prud'hommeaux, W3C <eric@w3.org>
Juan Sequeda, University of Texas at Austin <jsequeda@cs.utexas.edu>

Direct Mapping



Input:

- Database (Schema and Data)
- Primary Keys
- Foreign Keys

Output

- RDF graph

Generating Identifiers

- Identifier for rows, tables, columns and foreign keys
- If a table has a primary key,
  - then the row identifier will be an IRI,
  - otherwise a blank node
- The identifiers for table, columns and foreign keys are IRIs
- IRIs are generated by appending to a given base IRI
- All strings are percent encoded

Row IRI

$$\text{1) <}\overbrace{\text{http://www.ex.com/}}^{\text{Base IRI}}\overbrace{\text{Person/ID=1}}^{\text{“Table Name”/“PK attr”=“PK value”}}\text{>}$$

$$\text{2) <}\overbrace{\text{http://www.ex.com/}}^{\text{Base IRI}}\overbrace{\text{Person/ID=1;SID=123}}^{\text{“Table Name”/“PK attr”=“PK value”}}\text{>}$$

3) Fresh Blank Node (if table has no keys)

Table, Attribute, Foreign Key IRIs

1) $\langle\overbrace{\text{http://www.ex.com/}}^{\text{Base IRI}}\overbrace{\text{Person}}^{\text{“Table Name”}}\rangle$

2) $\langle\overbrace{\text{http://www.ex.com/}}^{\text{Base IRI}}\overbrace{\text{Person\#NAME}}^{\text{“Table Name”}\textbf{\#}\text{“Attribute”}}\rangle$

3) $\langle\overbrace{\text{http://www.ex.com/}}^{\text{Base IRI}}\overbrace{\text{Person\#}\textbf{ref-}\text{CID}}^{\text{“Table Name”}\textbf{\#ref-}\text{“Attribute”}}\rangle$

Table rows: Instance Type Triple

Person

ID (pk)	NAME	AGE
1	Alice	25
2	Bob	NULL

<http://www.ex.com/Person/ID=1>

rdf:type

<http://www.ex.com/Person>

Row values: Literal Triples

Person

ID (pk)	NAME	AGE
1	Alice	25
2	Bob	NULL

<http://www.ex.com/Person/ID=1>

<http://www.ex.com/Person#NAME>

"Alice" .

Foreign Keys: Reference Triples

Person

ID (pk)	NAME	AGE	CID (fk)
1	Alice	25	100
2	Bob	NULL	200

City

CID (pk)	TITLE
100	Austin
200	Madrid

<http://www.ex.com/Person/ID=1>

<http://www.ex.com/Person#**ref-**CID>

<http://www.ex.com/City/CID=100>.

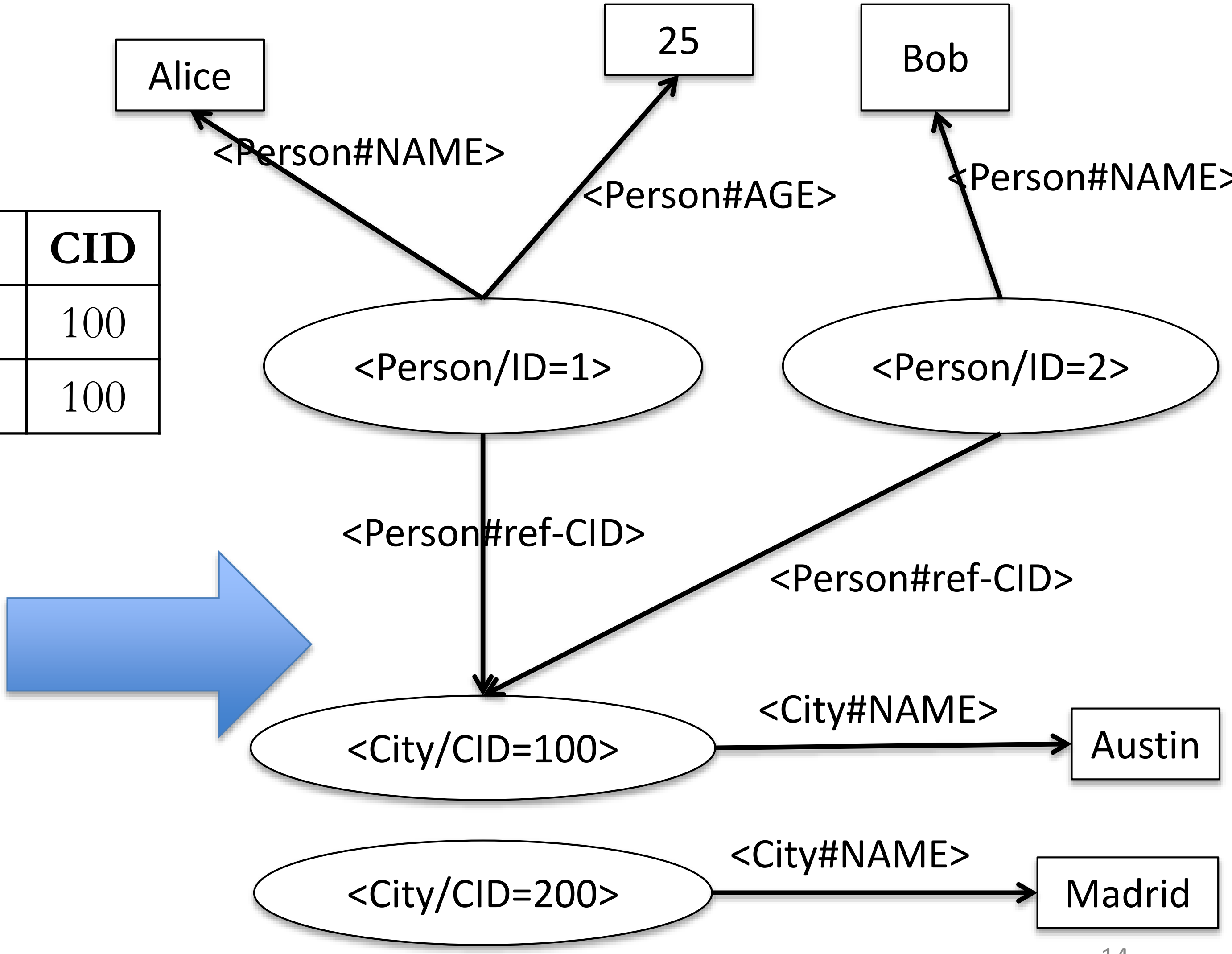
Direct Mapping Result

Person

ID	NAME	AGE	CID
1	Alice	25	100
2	Bob	NULL	100

City

CID	NAME
100	Austin
200	Madrid



Summary: Direct Mapping

- Default and Automatic Mapping
- URIs are automatically generated
  - <table>
  - <table#attribute>
  - <table#ref-attribute>
  - <Table#pkAttr=pkValue>
- RDF represents the same relational schema
- RDF can be transformed by SPARQL CONSTRUCT
  - RDF represents the structure and ontology of mapping author’s choice

What is missing from the Direct Mapping?

- No mapping to a desired ontology specified
- NULL values?
- “Ugly” IRIs

NULL

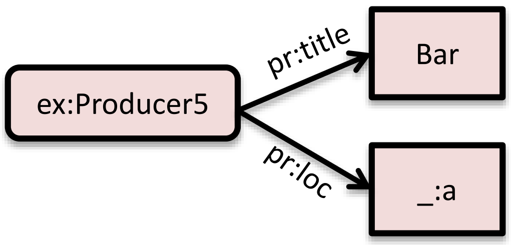
“The direct mapping does not generate triples for NULL values. Note that it is not known how to relate the behavior of the obtained RDF graph with the standard SQL semantics of the NULL values of the source RDB.”

A Direct Mapping of Relational Data to RDF.

W3C Recommendation

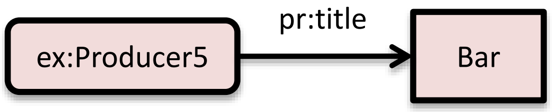
NULLs

- What should we do with NULLs?
  - Generate a Blank Node

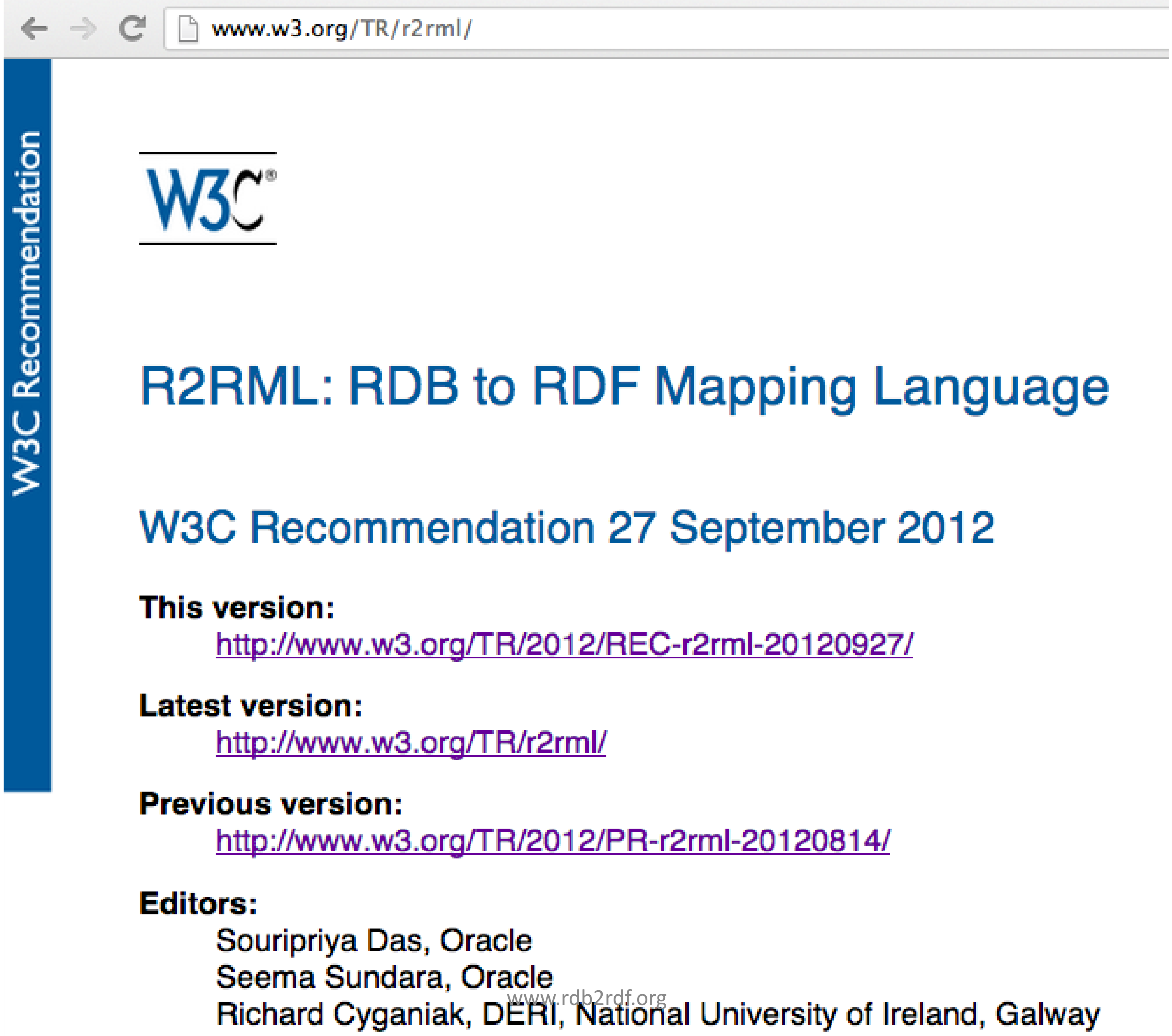


prID	title	loc
4	Foo	TX
5	Bar	NULL

  - Don’t generate a triple



How do we reconstruct the NULL?



www.w3.org/TR/r2rml/

W3C Recommendation

R2RML: RDB to RDF Mapping Language

W3C Recommendation 27 September 2012

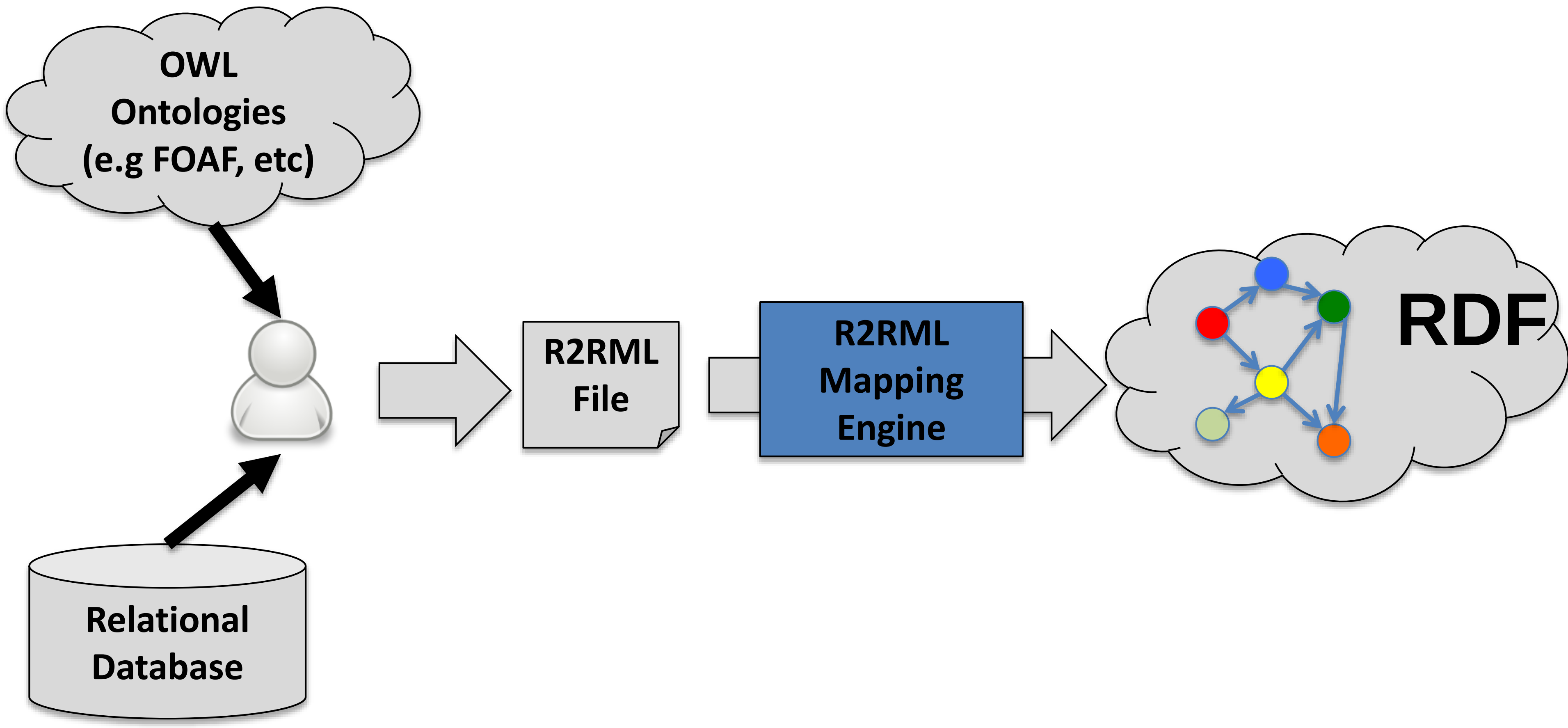
This version:
http://www.w3.org/TR/2012/REC-r2rml-20120927/

Latest version:
http://www.w3.org/TR/r2rml/

Previous version:
http://www.w3.org/TR/2012/PR-r2rml-20120814/

Editors:
Souripriya Das, Oracle
Seema Sundara, Oracle
Richard Cyganiak, DERI, National University of Ireland, Galway

R2RML


OWL
Ontologies
(e.g FOAF, etc)
R2RML
File
R2RML
Mapping
Engine
RDF
Relational
Database

R2RML Mapping

- An R2RML Mapping is represented as an RDF Graph itself.

- Associated RDFS schema
  - http://www.w3.org/ns/r2rml

- Turtle is the recommended syntax

Create R2RML mapping file

- Input
  - Knowledge of the database (schema and data)
  - Knowledge of the domain ontologies
  - Knowledge of mappings
- Output
  - R2RML file

R2RML Mapping

- A R2RML Mapping *M* consists of a finite set ***TM*** **TripleMaps**.
- Each *TM* ∈***TM*** consists of a tuple (*LT*, *SM*, ***POM***)
  - LT: **LogicalTable**
  - SM: **SubjectMap**
  - POM: **PredicateObjectMap**
- Each *POM*∈***POM*** consists of a pair (PM, OM)
  - PM: **PredicateMap**
  - OM: **ObjectMap**

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

<TriplesMap1>
  a rr:TriplesMap;

  rr:logicalTable [ rr:tableName ”Person”];

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

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

rr:logicalTable

Specifies table to be mapped to RDF

1. SQL base table or view
   – rr:tableName
2. R2RML View
   – rr:sqlQuery

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

<TriplesMap1>
  a rr:TriplesMap;

  rr:logicalTable [ rr:tableName ”Person”];

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

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

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

<TriplesMap1>
    a rr:TriplesMap;

  rr:logicalTable [ rr:sqlQuery
       “””SELECT ID, NAME
       FROM Person WHERE gender = “F” “””];

  rr:subjectMap [
    rr:template "http://www.ex.com/Person/{ID}";
    rr:class <http://www.ex.com/Woman>
  ];

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

Generating SPO

- TermMap that specifies what RDF term should be for S, P, O
  - **SubjectMap**
  - **PredicateMap**
  - **ObjectMap**

rr:subjectMap

- Specifies how to generate subject of triple
  - Usually based on a template
- Has to be an IRI or Blank Node
- May have one or more Class IRIs associated
  - rr:class
  - Generates rdf:type triples

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

<TriplesMap1>
  a rr:TriplesMap;

  rr:logicalTable [ rr:tableName ”Person”];

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

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

Example 1

- We now have sufficient elements to create a mapping that will generate
  - A Subject IRI
  - rdf:Type triple(s)

Student

sid	name	pid
1	Juan	100
2	Martin	200

```
@prefix ex: <http://example.com/ns/>.

ex:Student1 rdf:type ex:Student .
ex:Student2 rdf:type ex:Student .
```

Example 1

```
@prefix rr: <http://www.w3.org/ns/r2rml#>.
@prefix ex: <http://example.com/ns/>.

<#TriplesMap1>
  rr:logicalTable [ rr:tableName ”Student”];
  rr:subjectMap [
    rr:template "http://example.com/ns/Student{sid}";
    rr:class ex:Student;
].
```

Logical Table is a Table Name

SubjectMap is a Template-valued TermMap
And it has one Class IRI

Example 1

```
@prefix rr: <http://www.w3.org/ns/r2rml#>.
@prefix ex: <http://example.com/ns/>.

<#TriplesMap1>
  rr:logicalTable [ rr:tableName ”Student”];
  rr:subjectMap [
    rr:template "http://example.com/ns/{sid}";
    rr:class ex:Student;
].
```

Logical Table is a Table Name

SubjectMap is a Template-valued TermMap And it has one Class IRI

rr:predicateObjectMap

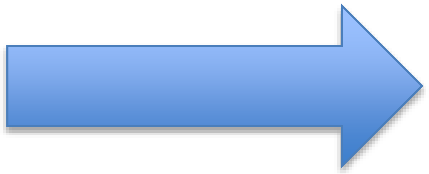
- Creates one or more predicate-object pairs for each logical table row.
- Used in conjunction with a SubjectMap to generate RDF triples in a TriplesMap.
- A predicate-object pair consists of
  - One or more PredicateMaps
  - One or more ObjectMaps or ReferencingObjectMaps

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

<TriplesMap1>
  a rr:TriplesMap;

  rr:logicalTable [ rr:tableName ”Person”];

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

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

Example 2

Student

sid	name	pid
1	Juan	100
2	Martin	200



```
@prefix ex: <http://example.com/ns/>.

ex:Student1 rdf:type ex:Student .
ex:Student1 ex:name “Juan” .
ex:Student2 rdf:type ex:Student .
ex:Student2 ex:name “Martin” .
```

Example 2

```
@prefix rr: <http://www.w3.org/ns/r2rml#>.
@prefix ex: <http://example.com/ns/>.

<#TriplesMap1>
  rr:logicalTable [ rr:tableName ”Student”];
  rr:subjectMap [
    rr:template "http://example.com/ns/{sid}";
    rr:class ex:Student;
  ];
  rr:predicateObjectMap [
    rr:predicate ex:name;
    rr:objectMap [ rr:column “name”];
  ].
```

Logical Table is a Table Name

SubjectMap is a
Template-valued TermMap
And it has one Class IRI

PredicateObjectMap

PredicateMap which is a
Constant-valued TermMap

ObjectMap which is a
Column-valued TermMap

Example 3

Student

sid	name	pid
1	Juan	100
2	Martin	200

TripleMap

```
@prefix ex: <http://example.com/ns/>.

ex:Student1 rdf:type ex:Student .
ex:Student1 ex:comment “Juan is a Student” .
ex:Student2 rdf:type ex:Student .
ex:Student2 ex:comment “Martin is a Student” .
```

Example 3

```
@prefix rr: <http://www.w3.org/ns/r2rml#>.
@prefix ex: <http://example.com/ns/>.

<#TriplesMap1>
  rr:logicalTable [ rr:tableName ”Student”];
  rr:subjectMap [
    rr:template "http://example.com/ns/{sid}";
    rr:class ex:Student;
  ];
  rr:predicateObjectMap [
    rr:predicate ex:comment;
    rr:objectMap [
      rr:template “{name} is a Student”;
      rr:termType rr:Literal;
    ];
  ].
```

Logical Table is a Table Name

SubjectMap is a
Template-valued TermMap
And it has one Class IRI

PredicateObjectMap

ObjectMap which is a
Template-valued TermMap

PredicateMap which is a
Constant-valued TermMap

Example 4

Student

sid	name	pid
1	Juan	100
2	Martin	200

TripleMap

```
@prefix ex: <http://example.com/ns/>.

ex:Student1 rdf:type ex:Student .
ex:Student1 ex:webpage <http://ex.com/Juan>.
ex:Student2 rdf:type ex:Student .
ex:Student2 ex:webpage <http://ex.com/Martin>.
```

Example 4

```
@prefix rr: <http://www.w3.org/ns/r2rml#>.
@prefix ex: <http://example.com/ns/>.

<#TriplesMap1>
  rr:logicalTable [ rr:tableName ”Student”];
  rr:subjectMap [
    rr:template "http://example.com/ns/{sid}";
    rr:class ex:Student;
  ];
  rr:predicateObjectMap [
    rr:predicate ex:webpage;
    rr:objectMap [
      rr:template “http://ex.com/{name}”;
    ];
  ].
```

Logical Table is a Table Name

SubjectMap is a Template-valued TermMap And it has one Class IRI

PredicateObjectMap

ObjectMap which is a Template-valued TermMap

PredicateMap which is a Constant-valued TermMap

Note that there is not **<u>TermType</u>**

Example 5

Student

sid	name	pid
1	Juan	100
2	Martin	200

TripleMap

```
@prefix ex: <http://example.com/ns/>.

ex:Student1 rdf:type ex:Student .
ex:Student1 ex:studentType ex:GradStudent.
ex:Student2 rdf:type ex:Student .
ex:Student2 ex:studentType ex:GradStudent.
```

Example 5

```
@prefix rr: <http://www.w3.org/ns/r2rml#>.
@prefix ex: <http://example.com/ns/>.

<#TriplesMap1>
  rr:logicalTable [ rr:tableName ”Student”];
  rr:subjectMap [
    rr:template "http://example.com/ns/{sid}";
    rr:class ex:Student;
  ];
  rr:predicateObjectMap [
    rr:predicate ex:studentType;
    rr:object ex:GradStudent ;
  ].
```

Logical Table is a Table Name

SubjectMap is a
Template-valued TermMap
And it has one Class IRI

PredicateObjectMap

ObjectMap which is a
Constant-valued TermMap

PredicateMap which is a
Constant-valued TermMap

RefObjectMap

- A RefObjectMap (Referencing ObjectMap) allows using the subject of another TriplesMap as the object generated by a ObjectMap.
- rr:objectMap
- A RefObjectMap defined by
  - Exactly one **ParentTripleMap**, which must be a TripleMap
  - May have one or more **JoinConditions**

Example 6

Student

sid	name	pid
1	Juan	100
2	Martin	200

Professor

pid	name
100	Dan
200	Marcelo

R2RML Mapping

$\longrightarrow$

ex:Student1 rdf:type ex:Student .
ex:Student2 rdf:type ex:Student .
ex:Professor100 rdf:type ex:Professor .
ex:Professor200 rdf:type ex:Professor .
ex:Student1 ex:hasAdvisor ex:Professor100 .
ex:Student2 ex:hasAdvisor ex:Professor200

Example 6

```
@prefix rr: <http://www.w3.org/ns/r2rml#>.
@prefix ex: <http://example.com/ns/>.

<#TriplesMap1>
  rr:logicalTable [ rr:tableName ”Student”];
  rr:subjectMap [
    rr:template "http://example.com/ns/{sid}";
    rr:class ex:Student;
  ];
  rr:predicateObjectMap [
    rr:predicate ex:hasAdvisor;
    rr:objectMap [
      rr:parentTriplesMap <#TriplesMap2>;
      rr:joinCondition [
       rr:child “pid”;
       rr:parent “pid”;
      ]
    ]
  ].
```

RefObjectMap

Parent TriplesMap

JoinCondition

```
<#TriplesMap2>
  rr:logicalTable [ rr:tableName ”Professor”];
  rr:subjectMap [
    rr:template "http://example.com/ns/{pid}";
    rr:class ex:Professor;
  ].
```

```
@prefix rr: <http://www.w3.org/ns/r2rml#>.
@prefix ex: <http://example.com/ns/>.
<#TriplesMap1>
 rr:logicalTable [ rr:tableName ”Student”];
 rr:subjectMap [
  rr:template "http://example.com/ns/{sid}";
  rr:class ex:Student; ];
 rr:predicateObjectMap [
  rr:predicate ex:name;
  rr:objectMap [ rr:column “name”];  ];
 rr:predicateObjectMap [
  rr:predicate ex:comment;
  rr:objectMap [
   rr:template “{name} is a Student”;
   rr:termType rr:Literal;  ];
 rr:predicateObjectMap [
  rr:predicate ex:webpage;
  rr:objectMap [
   rr:template “http://ex.com/{name}”;    ];
rr:predicateObjectMap [
  rr:predicate ex:studentType;
  rr:object ex:GradStudent ; ];
rr:predicateObjectMap [
  rr:predicate ex:hasAdvisor;
  rr:objectMap [
   rr:parentTriplesMap <#TriplesMap2>;
   rr:joinCondition [
    rr:child “pid”;
    rr:parent “pid”;      ]  ]  ].

<#TriplesMap2>
 rr:logicalTable [ rr:tableName ”Professor”];
 rr:subjectMap [
  rr:template "http://example.com/ns/{pid}";
  rr:class ex:Professor; ];
 rr:predicateObjectMap [
  rr:predicate ex:name;
  rr:objectMap [ rr:column “name”];  ];  ].
```

How I would have done the R2RDF Standard

```
@prefix ex: <http://example.com/ns/>

Student(sid, sname, pid) ^ professor(pid, pname) →
    template("http://example.com/ns/Student{sid}", S) ^          Ex1
    ex:Student(S) ^                                              Ex1
    ex:name(S sname) ^                                           Ex2
    template("{sname} is a Student", C) ^                        Ex3
    ex:comment(S, C) ^ rr:Literal(C) ^                           Ex3
    template("http://ex.com/{sname}", U) ^                       Ex4
    ex:webpage(S, U) ^                                           Ex4
    ex:studentType(S, ex:GradStudent) ^                          Ex5
    template("http://example.com/ns/Professor{pid}", P) ^        Ex6
    ex:Professor(P) ^                                            Ex6
    ex:hasAdvisor(S, P) ^                                        Ex6
    ex:name(P pname)                                             Extra ☺
```

Student

sid	name	pid
1	Juan	100
2	Martin	200

Professor

pid	name
100	Dan
200	Marcelo

How I would have done the R2RDF Standard

```
@prefix ex: <http://example.com/ns/>

Student(sid, sname, pid) ^ professor(pid, pname) →
    uri(“http://example.com/ns/Student{sid}”) = S ^              Ex1
    ex:Student(S) ^                                              Ex1
    ex:name(S sname) ^                                           Ex2
    string(“{sname} is a Student”) = C ^                         Ex3
    ex:comment(S, C) ^ rr:Literal(C) ^                           Ex3
    uri(“http://ex.com/{sname}”) = U ^                           Ex4
    ex:webpage(S, U) ^                                           Ex4
    ex:studentType(S, ex:GradStudent) ^                          Ex5
    uri(“http://example.com/ns/Professor{pid}” = P ^             Ex6
    ex:Professor(P) ^                                            Ex6
    ex:hasAdvisor(S, P) ^                                        Ex6
    ex:name(P pname)                                             Extra ☺
```

Student

sid	name	pid
1	Juan	100
2	Martin	200

Professor

pid	name
100	Dan
200	Marcelo