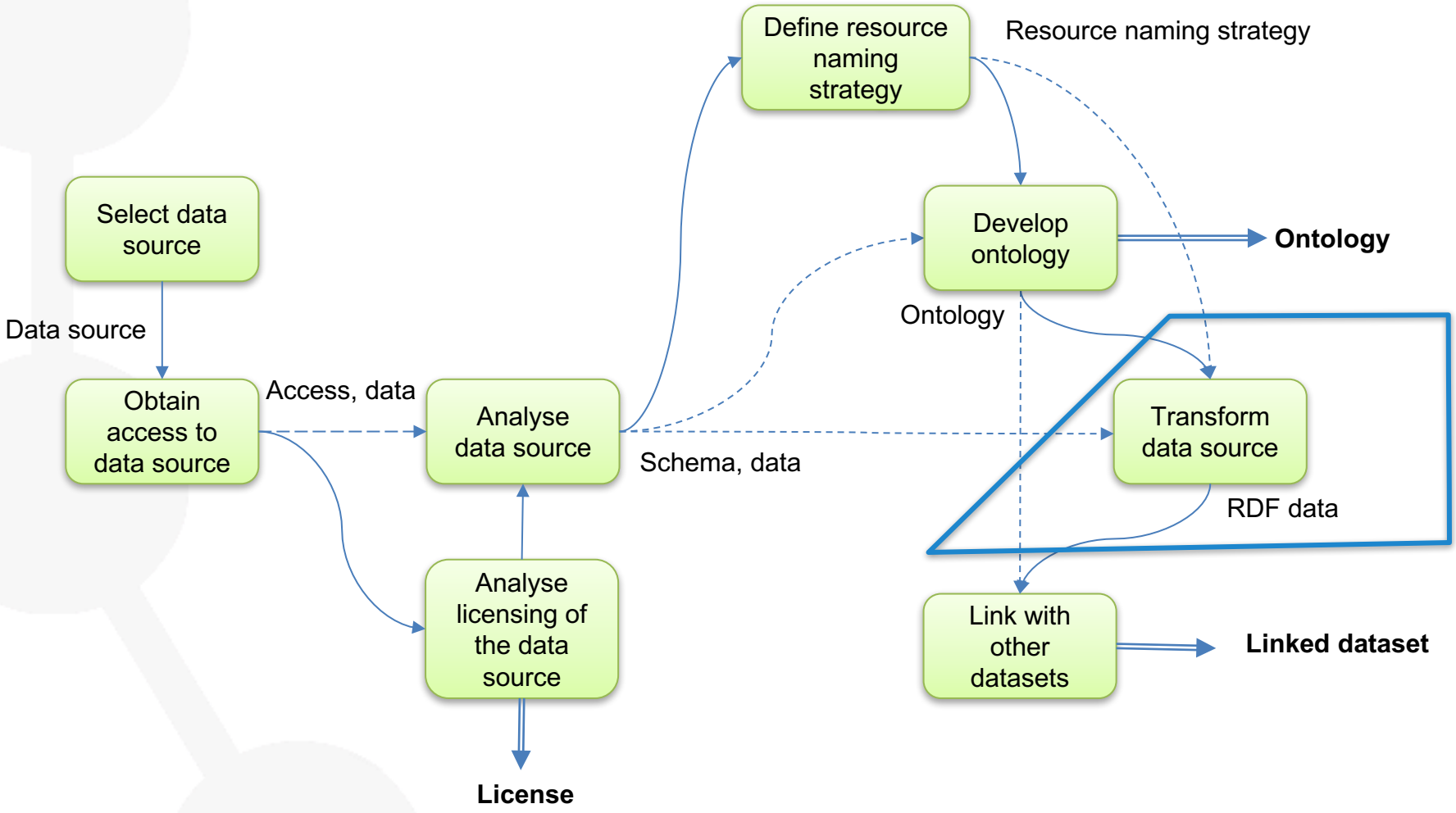
Publishing Linked Data from RDB (RDB2RDF)

Oscar Corcho, Freddy Priyatna, Raúl García-Castro
Escuela Técnica Superior de Ingenieros Informáticos,
Universidad Politécnica de Madrid
Campus de Montegancedo sn, 28660 Boadilla del Monte, Madrid
http://www.oeg-upm.net/
{ocorcho,fpriyatna,rgarcia}@fi.upm.es

Acknowledgements: Some slides are taken from Juan Sequeda's RDB2RDF tutorial at ISWC2013

Table of Contents

Linked Data generation process



Deliverable D4.1: Requirements and guidelines for energy data generation. *Filip Radulovic, Raúl García-Castro, María Poveda-Villalón, Matthias Weise, Thanasis Tryferidis.* READY4SmartCities technical report. May 2014

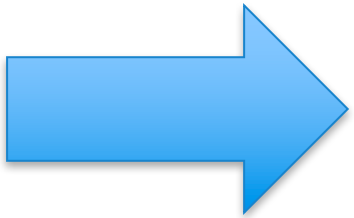
What is RDB2RDF?

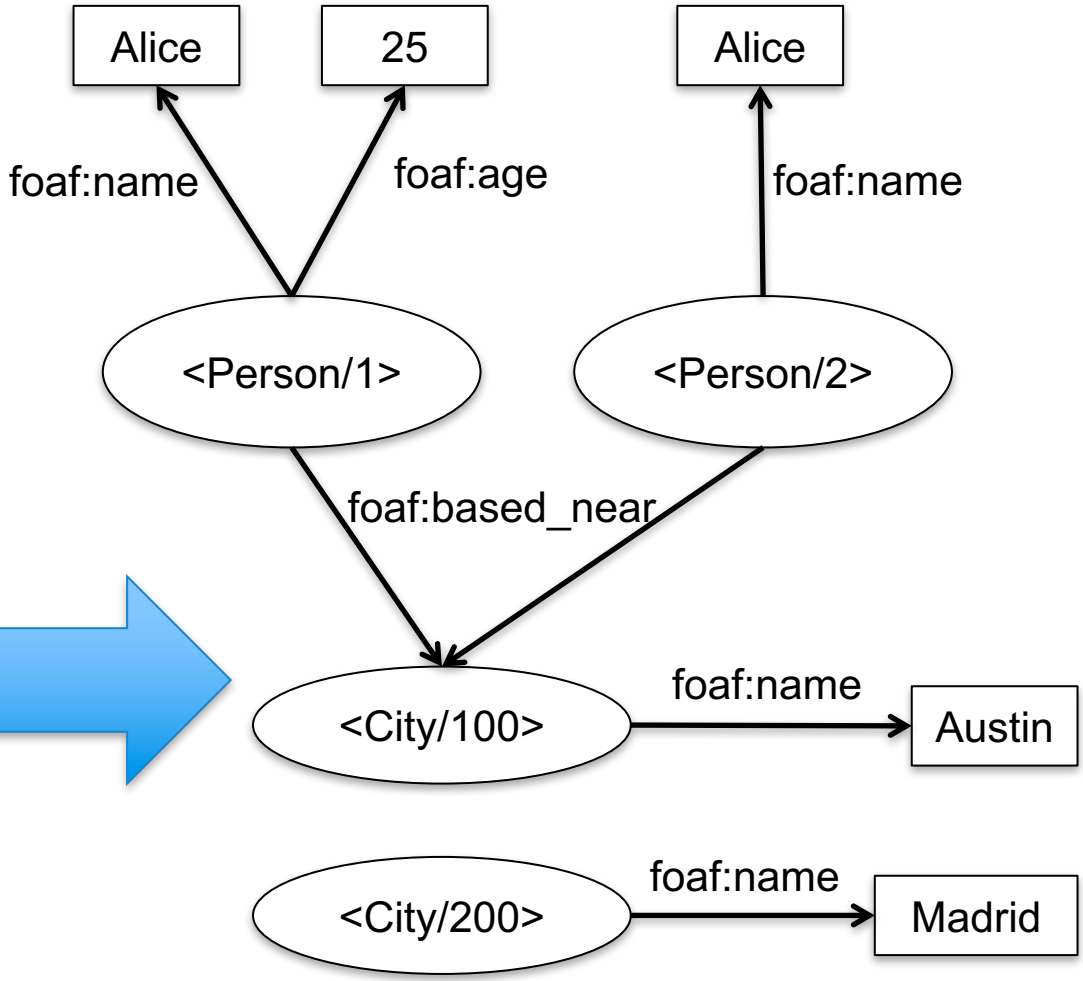
Person

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

City

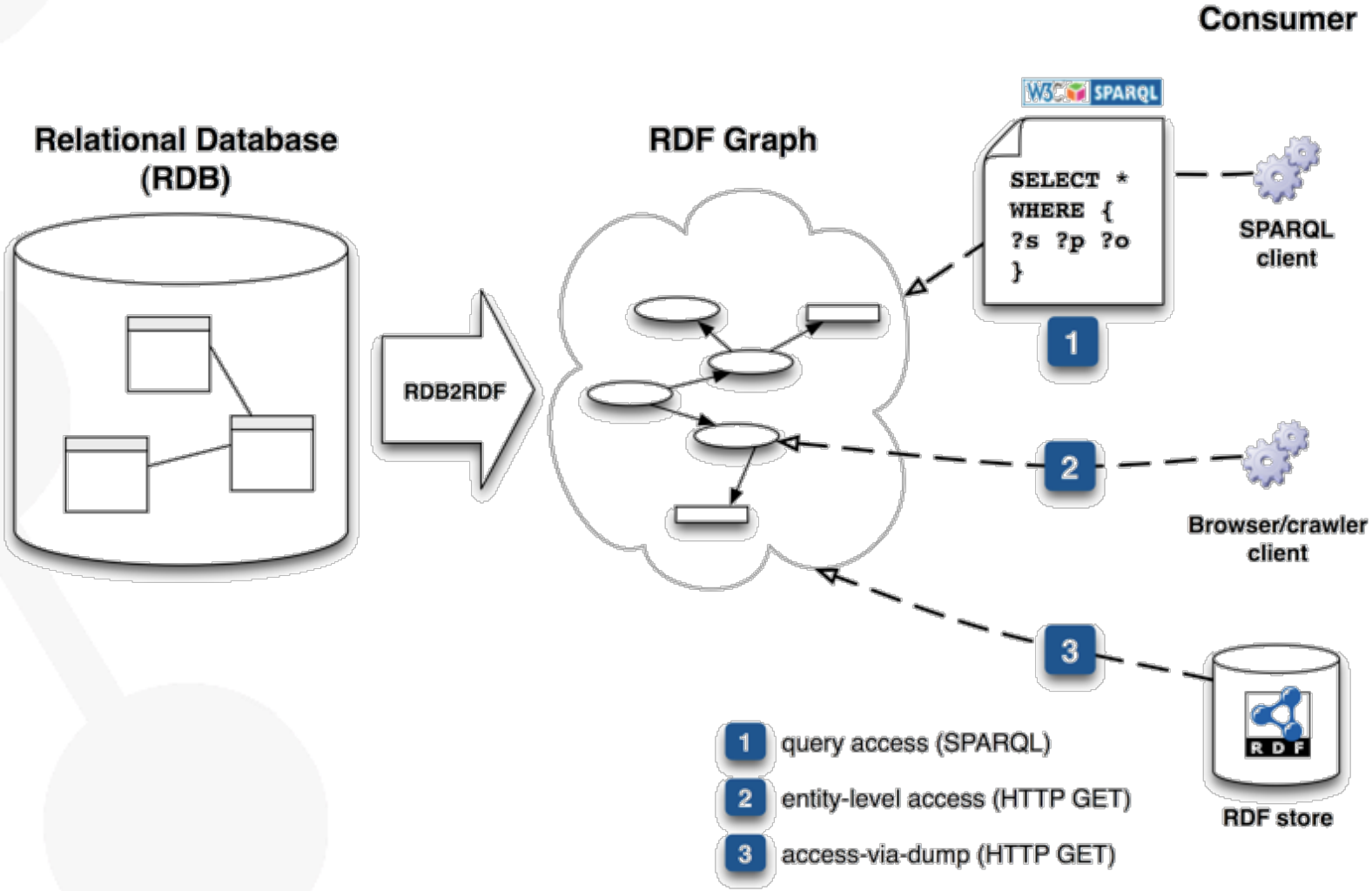
CID	NAME
100	Austin
200	Madrid





Why RDB2RDF?

- A majority of dynamic Web content is backed by relational databases (RDB), and so are many enterprise systems



RDB2RDF Requirements

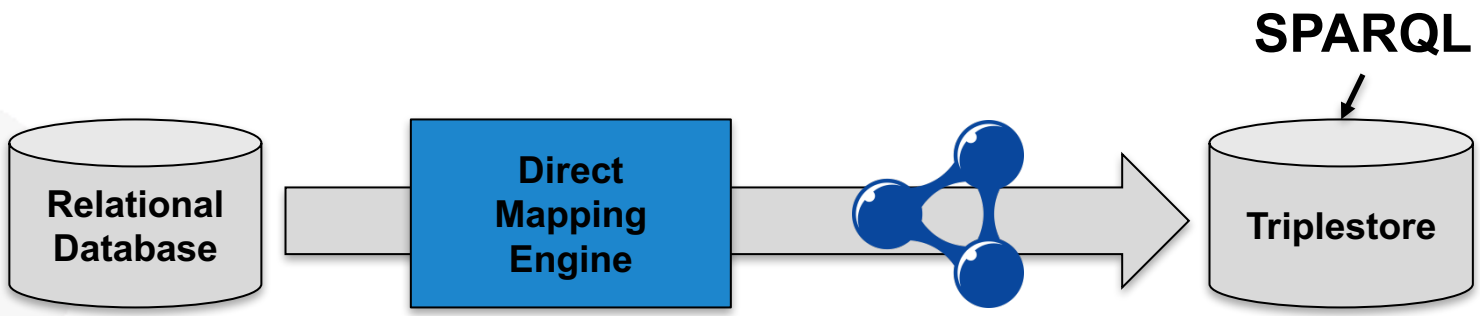
- **Core**
  - Direct + transformative mappings
  - Generation of Globally Unique Identifiers (i.e., URIs)
  - Materialize RDF (ETL) + virtual RDF views (query translation)
  - Datatypes (inc. vendor specific)
  - Rename SQL column names
  - Apply functions before mapping
  - Exposing many-to-many join tables as simple triples
  - Creating classes based on attribute values
- **Optional**
  - Named graphs, namespace declaration, static metadata (e.g., licensing, provenance)

RDB2RDF Scenarios

- **Scenario 1. Direct mapping**
  - Quick solution to export RDF
  - Potentially good for large database schemas
- **Scenario 2. R2RML**
  - Existing ontologies that allow interoperability with other sources
  - Mostly a manual effort

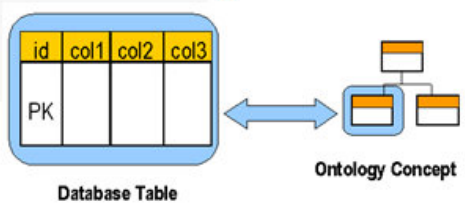
Scenario 1: Direct Mapping

- Existing table and column names are encoded into URIs
- Data is translated into RDF and loaded into an existing, Internet accessible triplestore



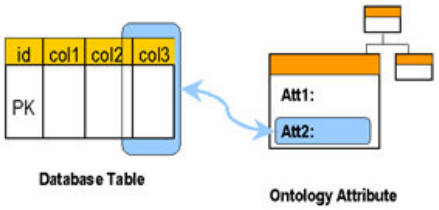
Extract **Transform** **Load**

For concepts...



A view maps exactly one concept in the ontology.

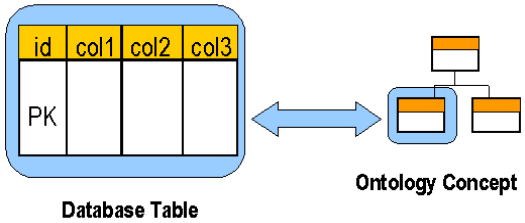
For attributes...



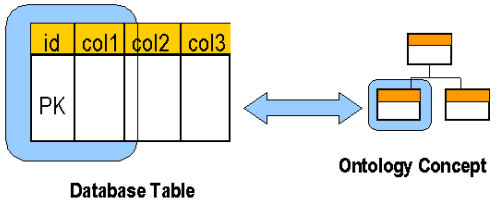
A column in a database view maps directly an attribute or a relation.

However, different mapping alternatives exist

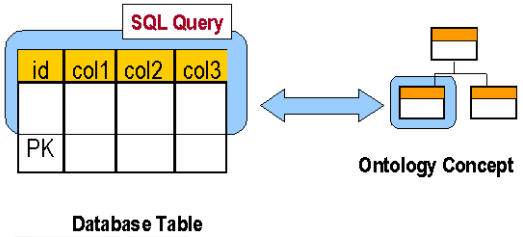
For concepts...



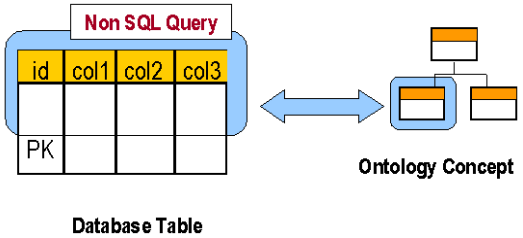
A view maps exactly one concept in the ontology.



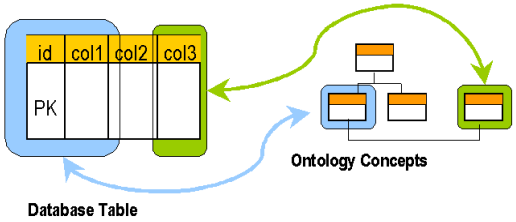
A subset of the columns in the view maps a concept in the ontology.



A subset (selection) of the records of a database view maps a concept in the ontology.

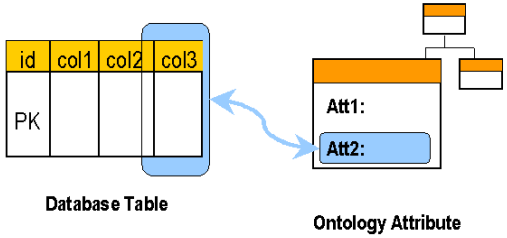


A subset of the records of a database view maps a concept in the ontology, but the selection cannot be made using SQL.

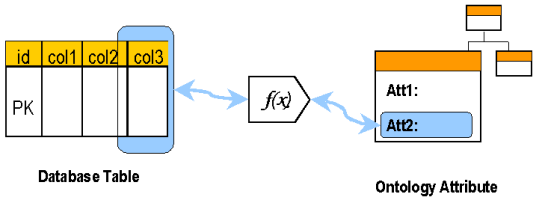


One or more concepts can be extracted from a single data field (not in 1NF).

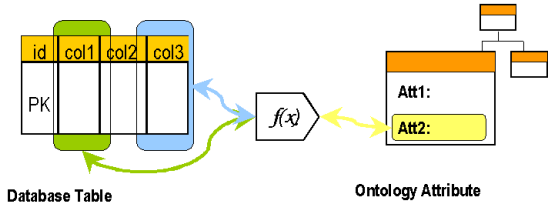
For attributes...



A column in a database view maps directly an attribute or a relation.

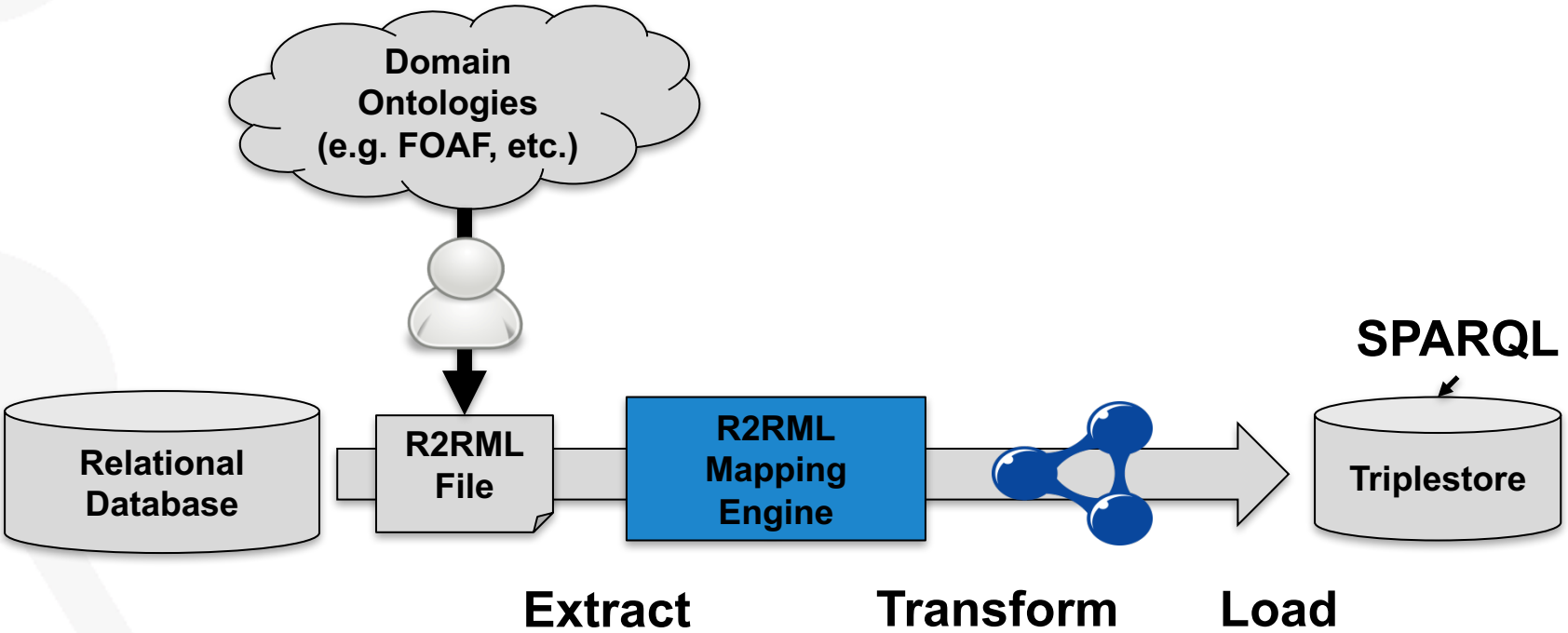


A column in a database view maps an attribute or a relation after some transformation.



A set of columns in a database view map an attribute or a relation.

Scenario 2: R2RML



W3C RDB2RDF Standards

- A Direct Mapping of Relational Data to RDF
  - Default automatic mapping of relational data to RDF
  - W3C Recommendation 27 September 2012

- R2RML: RDB to RDF Mapping Language
  - Customizable language to map relational data to RDF
  - W3C Recommendation 27 September 2012

W3C Recommendation

W3C

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>

W3C Recommendation

W3C

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

Table of Contents

Direct Mapping

- The direct mapping defines an RDF Graph representation of the data in an RDB
- It takes as input an RDB (data and schema), and generates an RDF graph (called the direct graph)

```
CREATE TABLE "Addresses" (
        "ID" INT, PRIMARY KEY("ID"),
        "city" CHAR(10),
        "state" CHAR(2)
)

CREATE TABLE "People" (
        "ID" INT, PRIMARY KEY("ID"),
        "fname" CHAR(10),
        "addr" INT,
        FOREIGN KEY("addr") REFERENCES "Addresses"("ID")
)

INSERT INTO "Addresses" ("ID", "city", "state") VALUES (18, 'Cambridge', 'MA')
INSERT INTO "People" ("ID", "fname", "addr") VALUES (7, 'Bob', 18)
INSERT INTO "People" ("ID", "fname", "addr") VALUES (8, 'Sue', NULL)
```

People

PK		→ Address(ID)
ID	fname	addr
7	Bob	18
8	Sue	NULL

Addresses

PK		
ID	city	state
18	Cambridge	MA

Direct Mapping - example

People

PK		→ Address(ID)
ID	fname	addr
7	Bob	18
8	Sue	NULL

Addresses

PK		
ID	city	state
18	Cambridge	MA

For each row in every table:
<TABLENAME/PKCOLUMNNAME=CELLVALUE> rdf:type <TABLENAME>

```
<People/ID=7> rdf:type <People> .
<People/ID=7> <People#ID> 7 .
<People/ID=7> <People#fname> "Bob" .
<People/ID=7> <People#addr> 18 .
<People/ID=7> <People#ref-addr> <Addresses/ID=18> .
<People/ID=8> rdf:type <People> .
<People/ID=8> <People#ID> 8 .
<People/ID=8> <People#fname> "Sue" .

<Addresses/ID=18> rdf:type <Addresses> .
<Addresses/ID=18> <Addresses#ID> 18 .
<Addresses/ID=18> <Addresses#city> "Cambridge" .
<Addresses/ID=18> <Addresses#state> "MA" .
```

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

Direct Mapping - example

People

PK		→ Address(ID)
ID	fname	addr
7	Bob	18
8	Sue	NULL

Addresses

PK		
ID	city	state
18	Cambridge	MA

For every cell
<TABLENAME/COLUMNNAME=CELLVALUE> <TABLENAME#COLUMNNAME> <CELLVALUE>

```
<People/ID=7> rdf:type <People> .
<People/ID=7> <People#ID> 7 .
<People/ID=7> <People#fname> "Bob" .
<People/ID=7> <People#addr> 18 .
<People/ID=7> <People#ref-addr> <Addresses/ID=18> .
<People/ID=8> rdf:type <People> .
<People/ID=8> <People#ID> 8 .
<People/ID=8> <People#fname> "Sue" .

<Addresses/ID=18> rdf:type <Addresses> .
<Addresses/ID=18> <Addresses#ID> 18 .
<Addresses/ID=18> <Addresses#city> "Cambridge" .
<Addresses/ID=18> <Addresses#state> "MA" .
```

Direct Mapping - example

People

PK		→ Address(ID)
ID	fname	addr
7	Bob	18
8	Sue	NULL

Addresses

PK		
ID	city	state
18	Cambridge	MA

For every cell in the foreign key column
<TABLENAME/PKCOLUMNNAME=CELLVALUE> <TABLENAME#ref-COLUMNNAME> <REFTABLENAME/REFPKCOLUMNAME=CELLVALUE>

```
<People/ID=7> rdf:type <People> .
<People/ID=7> <People#ID> 7 .
<People/ID=7> <People#fname> "Bob" .
<People/ID=7> <People#addr> 18 .
<People/ID=7> <People#ref-addr> <Addresses/ID=18> .
<People/ID=8> rdf:type <People> .
<People/ID=8> <People#ID> 8 .
<People/ID=8> <People#fname> "Sue" .

<Addresses/ID=18> rdf:type <Addresses> .
<Addresses/ID=18> <Addresses#ID> 18 .
<Addresses/ID=18> <Addresses#city> "Cambridge" .
<Addresses/ID=18> <Addresses#state> "MA" .
```

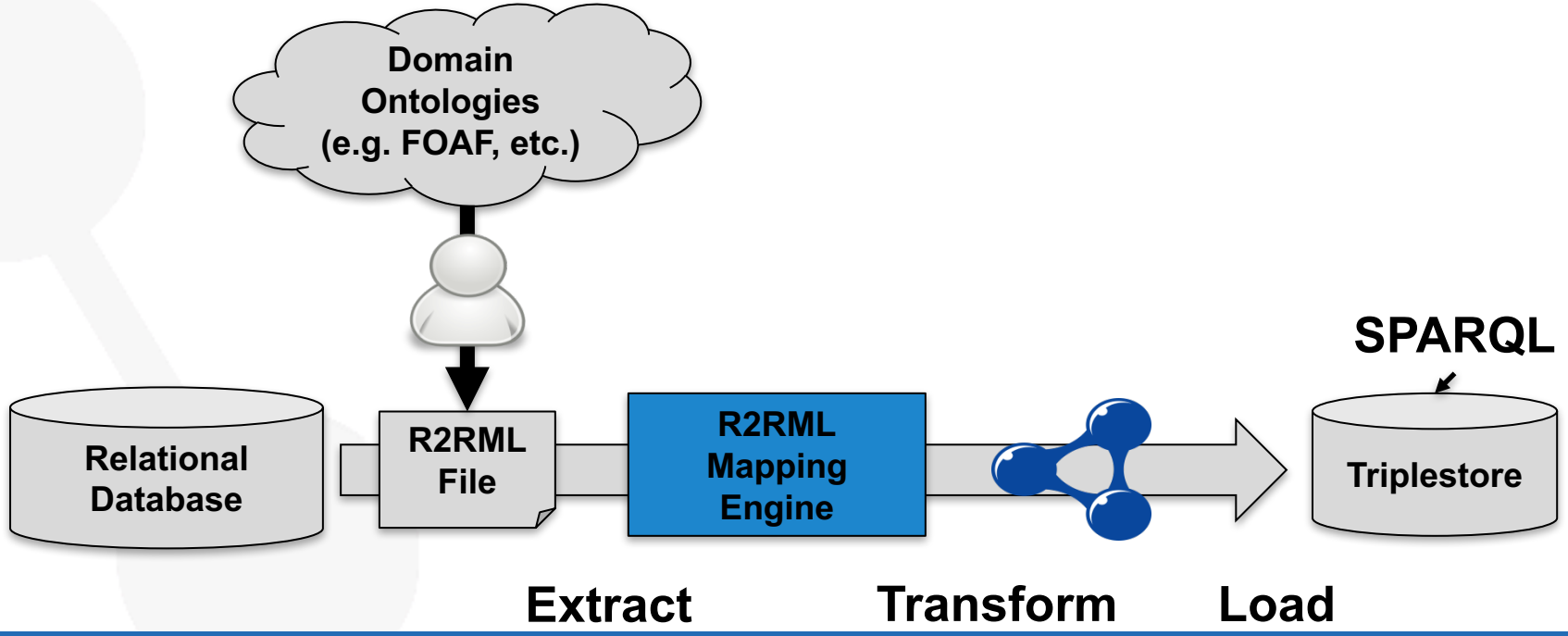
Summary: Direct Mapping

- Default and automatic mapping
- URIs are automatically generated
  - <table>
  - <table#attribute>
  - <table#ref-attribute>
  - <Table#pkAttr=pkValue>
- RDF represents the same relational schema

Table of Contents

W3C R2RML

- Input
  - Database (schema and data)
  - Target ontologies
  - Mappings between the database and target ontologies in R2RML
- Output
  - RDF graph



Example Input & Desired Output

Example Input Database

EMP

EMPNO	ENAME	JOB	DEPTNO
7369	SMITH	CLERK	10

DEPT

DEPTNO	DNAME	LOC
10	APPSERVER	NEW YORK

Desired RDF Output

```
<http://data.example.com/employee/7369> rdf:type ex:Employee.
<http://data.example.com/employee/7369> ex:name "SMITH".
<http://data.example.com/employee/7369> ex:department <http://data.example.com/department/10>.

<http://data.example.com/department/10> rdf:type ex:Department.
<http://data.example.com/department/10> ex:name "APPSERVER".
<http://data.example.com/department/10> ex:location "NEW YORK".
```

Triples Map

- R2RML Mapping Document = a finite set of Triples Maps (`rr:TriplesMap`)
- Specifies transformation rules from a logical table rows to RDF triples
  - MUST have exactly one `rr:logicalTable` property
- Generates the same subject
  - MUST have exactly one `rr:subjectMap` property
- MAY have zero or more `rr:predicateObjectMap` properties

Triples Map Example

EMP

EMPNO	ENAME	JOB	DEPTNO
7369	SMITH	CLERK	10

```
<http://data.example.com/employee/7369> rdf:type ex:Employee.
<http://data.example.com/employee/7369> ex:name "SMITH".
<http://data.example.com/employee/7369> ex:department
    <http://data.example.com/department/10>.
```

DEPT

DEPTNO	DNAME	LOC
10	APPSERVER	NEW YORK

```
<http://data.example.com/department/10> rdf:type ex:Department.
<http://data.example.com/department/10> ex:name "APPSERVER".
<http://data.example.com/department/10> ex:location "NEW YORK".
```

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

<#TriplesMap1>
   rr:logicalTable [ … ];
   rr:subjectMap [ … ];
   rr:predicateObjectMap [ … ];
   rr:predicateObjectMap [ … ];
   …
.

<#TriplesMap2>
   rr:logicalTable [ … ];
   rr:subjectMap [ … ];
   rr:predicateObjectMap [ … ];
   rr:predicateObjectMap [ … ];
   …
.
```

Logical Table

- Triples Map
  - MUST have exactly one `rr:logicalTable` property
- Logical table: tabular SQL query result that is to be mapped to RDF triples.
- Either
  - Base Table or View (`rr:table`)
  - R2RML view (`rr:query`)

Logical Table: Example

EMP

EMPNO	ENAME	JOB	DEPTNO
7369	SMITH	CLERK	10

```
<http://data.example.com/employee/7369> rdf:type ex:Employee.
<http://data.example.com/employee/7369> ex:name "SMITH".
<http://data.example.com/employee/7369> ex:department
   <http://data.example.com/department/10>.
```

DEPT

DEPTNO	DNAME	LOC
10	APPSERVER	NEW YORK

```
<http://data.example.com/department/10> rdf:type ex:Department.
<http://data.example.com/department/10> ex:name "APPSERVER".
<http://data.example.com/department/10> ex:location "NEW YORK".
```

```
<#TriplesMap1>
   rr:logicalTable [ rr:tableName "EMP" ];
   rr:subjectMap [ … ];
   rr:predicateObjectMap [ … ];
   rr:predicateObjectMap [ … ];
   …
.
```

```
<#TriplesMap2>
   rr:logicalTable [ rr:tableName "DEPT" ];
   rr:subjectMap [ … ];
   rr:predicateObjectMap [ … ];
   rr:predicateObjectMap [ … ];
   …
.
```

Subject Map and Predicate Map

- Triples Map
  - MUST have exactly one `rr:subjectMap` property
  - MAY have zero or more `rr:predicateObjectMap` properties
- Subject Map (`rr:SubjectMap`)
  - MAY have one or more `rr:class` properties
- PredicateObjectMap (`rr:PredicateObjectMap`)
  - Predicate Map (`rr:PredicateMap`)
  - Object Map (`rr:ObjectMap`)

rr:class: Example

EMP

EMPNO	ENAME	JOB	DEPTNO
7369	SMITH	CLERK	10

```
<http://data.example.com/employee/7369> rdf:type ex:Employee.
<http://data.example.com/employee/7369> ex:name "SMITH".
<http://data.example.com/employee/7369> ex:department
   <http://data.example.com/department/10>.
```

DEPT

DEPTNO	DNAME	LOC
10	APPSERVER	NEW YORK

```
<http://data.example.com/department/10> rdf:type ex:Department.
<http://data.example.com/department/10> ex:name "APPSERVER".
<http://data.example.com/department/10> ex:location "NEW YORK".
```

```
<#TriplesMap1>
   rr:logicalTable [ rr:tableName "EMP" ];
   rr:subjectMap [ …
      rr:class ex:Employee;
   ];
   rr:predicateObjectMap [
      rr:predicateMap [ … ]
      rr:objectMap [ … ]
   ];
   …
.
```

```
<#TriplesMap2>
   rr:logicalTable [ rr:tableName "DEPT" ];
   rr:subjectMap [ …
      rr:class ex:Department;
   ];
   rr:predicateObjectMap [
      rr:predicateMap [ … ]
      rr:objectMap [ … ]
   ];
   …
.
```

Term Map

- RDF Term
  - IRI
  - Literal
  - Blank Node
- Term Map (`rr:TermMap`): a function that generates an RDF term from a logical table row.

Where to use?

- Subject Map
- Predicate Map
- Object Map

How to generate?

- Constant
- Column
- Template

What type?

- IRI
- Literal
- Blank Node

Subject Map: Example

EMP

EMPNO	ENAME	JOB	DEPTNO
7369	SMITH	CLERK	10

```
<#TriplesMap1>
  rr:logicalTable [ rr:tableName "EMP" ];
  rr:subjectMap [
    rr:template "http://data.example.com/employee/{EMPNO}";
    rr:termType rr:IRI; rr:class ex:Employee
  ];
  rr:predicateObjectMap [
    rr:predicateMap [ … ] rr:objectMap [ … ]
  ];
  …
.
```

```
<http://data.example.com/employee/7369> rdf:type ex:Employee.
<http://data.example.com/employee/7369> ex:name "SMITH".
<http://data.example.com/employee/7369> ex:department <http://data.example.com/department/10>.
```

DEPT

DEPTNO	DNAME	LOC
10	APPSERVER	NEW YORK

```
<#TriplesMap2>
  rr:logicalTable [ rr:tableName "DEPT" ];
  rr:subjectMap [
    rr:template "http://data.example.com/department/{DEPTNO}";
    rr:termType rr:IRI; rr:class ex:Department
  ];
  rr:predicateObjectMap [
    rr:predicateMap [ … ] rr:objectMap [ … ]
  ];
  …
.
```

```
<http://data.example.com/department/10> rdf:type ex:Department.
<http://data.example.com/department/10> ex:name "APPSERVER".
<http://data.example.com/department/10> ex:location "NEW YORK".
```

Predicate Object Map: Example(1)

EMP

EMPNO	ENAME	JOB	DEPTNO
7369	SMITH	CLERK	10

```
<#TriplesMap1>
  rr:logicalTable [ rr:tableName "EMP" ];

  rr:subjectMap [
    rr:template "http://data.example.com/employee/{EMPNO}";
    rr:termType rr:IRI; rr:class ex:Employee;
  ];

  rr:predicateObjectMap [
    rr:predicateMap [ rr:constant ex:name; rr:termType rr:IRI ]
    rr:objectMap [ rr:column "ENAME"; rr:termType rr:Literal ]
  ];
  …
.
```

```
<http://data.example.com/employee/7369> rdf:type ex:Employee.
<http://data.example.com/employee/7369> ex:name "SMITH".
<http://data.example.com/employee/7369> ex:department <http://data.example.com/department/10>.
```

Predicate Object Map: Example (2)

DEPT

DEPTNO	DNAME	LOC
10	APPSERVER	NEW YORK

```
<#TriplesMap2>
  rr:logicalTable [ rr:tableName "DEPT" ];
  rr:subjectMap [
    rr:template “http://data.example.com/employee/{DEPTNO}”;
    rr:termType rr:IRI; rr:class ex:Department;
  ];
  rr:predicateObjectMap [
    rr:predicateMap [ rr:constant ex:name; rr:termType rr:IRI ]
    rr:objectMap [ rr:column “DNAME”; rr:termType rr:Literal ]
  ];
  rr:predicateObjectMap [
    rr:predicateMap [ rr:constant ex:location; rr:termType rr:IRI ]
    rr:objectMap [ rr:column “LOC”; rr:termType rr:Literal ]
  ];
.
```

```
<http://data.example.com/department/10> rdf:type ex:Department.
<http://data.example.com/department/10> ex:name "APPSERVER".
<http://data.example.com/department/10> ex:location "NEW YORK".
```

Almost There

Example Input Database

EMP

EMPNO	ENAME	JOB	DEPTNO
7369	SMITH	CLERK	10

DEPT

DEPTNO	DNAME	LOC
10	APPSERVER	NEW YORK

Desired RDF Output

```
<http://data.example.com/employee/7369> rdf:type ex:Employee.
<http://data.example.com/employee/7369> ex:name "SMITH".
<http://data.example.com/employee/7369> ex:department <http://data.example.com/department/10>.

<http://data.example.com/department/10> rdf:type ex:Department.
<http://data.example.com/department/10> ex:name "APPSERVER".
<http://data.example.com/department/10> ex:location "NEW YORK".
```

RefObjectMap

- Using subjects of another triples map as the value for objects
- join between logical tables
- `rr:RefObjectMap`
  - has exactly one `rr:parentTriplesMap` property
  - MAY have one or more `rr:joinCondition` properties
    - `rr:child`
    - `rr:parent`

RefObjectMap: Example

EMP

EMPNO	ENAME	JOB	DEPTID
7369	SMITH	CLERK	10

```
<http://data.example.com/employee/7369> rdf:type ex:Employee.
<http://data.example.com/employee/7369> ex:name "SMITH".
<http://data.example.com/employee/7369> ex:department
   <http://data.example.com/department/10>.
```

```
<#TriplesMap1>
  rr:logicalTable [ rr:tableName "EMP" ];
  rr:subjectMap [
    rr:template "http://data.example.com/employee/{EMPNO}";
    rr:termType rr:IRI; rr:class ex:Employee
  ];
  rr:predicateObjectMap [
    rr:predicateMap [
      rr:constant ex:department; rr:termType rr:IRI
    ]
    rr:objectMap [
      rr:parentTriplesMap <#TriplesMap2>;
      rr:joinCondition [
        rr:child "DEPTID";  rr:parent "DEPTNO";
      ];
    ]
  ];
  …
.
```

DEPT

DEPTNO	DNAME	LOC
10	APPSERVER	NEW YORK

```
<http://data.example.com/department/10> rdf:type ex:Department.
<http://data.example.com/department/10> ex:name "APPSERVER".
<http://data.example.com/department/10> ex:location "NEW YORK".
```

```
<#TriplesMap2>
  rr:logicalTable [ rr:tableName "DEPT" ];
  rr:subjectMap [
    rr:template "http://data.example.com/department/{DEPTNO}";
    rr:termType rr:IRI; rr:class ex:Department
  ];
  rr:predicateObjectMap [
    rr:predicateMap [ … ] rr:objectMap [ … ]
  ];
  …
.
```

Complete Mappings

EMP

EMPNO	ENAME	JOB	DEPTID
7369	SMITH	CLERK	10

```
<http://data.example.com/employee/7369> rdf:type
   ex:Employee.
<http://data.example.com/employee/7369> ex:name "SMITH".
<http://data.example.com/employee/7369> ex:department
   <http://data.example.com/department/10>.
```

```
<#TriplesMap1>
  rr:logicalTable [ rr:tableName "EMP" ];

  rr:subjectMap [
    rr:template "http://data.example.com/employee/{EMPNO}";
    rr:termType rr:IRI; rr:class ex:Employee; ];

  rr:predicateObjectMap [
    rr:predicateMap [ rr:constant ex:name; rr:termType rr:IRI ]
    rr:objectMap [ rr:column "ENAME"; rr:termType rr:Literal ] ];

  rr:predicateObjectMap [
    rr:predicateMap [ rr:constant ex:department; rr:termType rr:IRI
]
    rr:objectMap [
      rr:parentTriplesMap <#TriplesMap2>;
      rr:joinCondition [ rr:child "DEPTID";  rr:parent "DEPTNO"; ];
    ]
  ];.
```

DEPT

DEPTNO	DNAME	LOC
10	APPSERVER	NEW YORK

```
<http://data.example.com/department/10> rdf:type
   ex:Department.
<http://data.example.com/department/10> ex:name
   "APPSERVER".
<http://data.example.com/department/10> ex:location
   "NEW YORK".
```

```
<#TriplesMap2>
  rr:logicalTable [ rr:tableName "DEPT" ];

  rr:subjectMap [
    rr:template "http://data.example.com/employee/{DEPTNO}";
    rr:termType rr:IRI; rr:class ex:Department; ];

  rr:predicateObjectMap [
    rr:predicateMap [ rr:constant ex:name; rr:termType rr:IRI ]
    rr:objectMap [ rr:column "DNAME"; rr:termType rr:Literal ] ];

  rr:predicateObjectMap [
    rr:predicateMap [ rr:constant ex:location; rr:termType rr:IRI ]
    rr:objectMap [ rr:column "LOC"; rr:termType rr:Literal ]
  ]; .
```

Table of Contents

Installing morph-RDB

GitHub, Inc. (US) https://github.com/oeg-upm/morph-rdb/wiki

This repository Search Pull requests Issues Gist

oeg-upm / morph-rdb

Unwatch 15 Unstar 13

Code Issues 15 Pull requests 0 Projects 0 Wiki Pulse Graphs

Home

Freddy Priyatna edited this page on 30 Sep · 55 revisions

In this page we provide some information that should be useful for users who want to install and make use of morph-RDB from a linux command line.

Installing morph-RDB

In order to install morph-RDB in your computer, you must follow the following set of instructions:

1. Create a new directory for the morph-RDB files
   i. mkdir morph
   ii. cd morph
2. Download the library files used by morph-RDB
   i. wget https://github.com/oeg-upm/morph-rdb/releases/download/v3.7.5/lib.zip
   ii. unzip lib.zip
3. Download the latest morph-RDB jar file and rename it to morph.jar
   i. wget https://github.com/oeg-upm/morph-rdb/releases/download/v3.7.8/morph-rdb-dist-3.7.8.jar
   ii. unlink morph-rdb.jar
   iii. ln -s morph-rdb-dist-3.7.8.jar morph-rdb.jar

Pages 1

Home

Add a custo

Clone this wiki local

https://github.co

Clone in Desktop

https://github.com/oeg-upm/morph-rdb/wiki

Running examples with CSV/RDB

Table of Contents

Assignment

- https://goo.gl/forms/ewFN0EVkNCF8p6k42

Questions?