xR2RML:
An R2RML Extension for the Translation of Non-Relational Databases to RDF

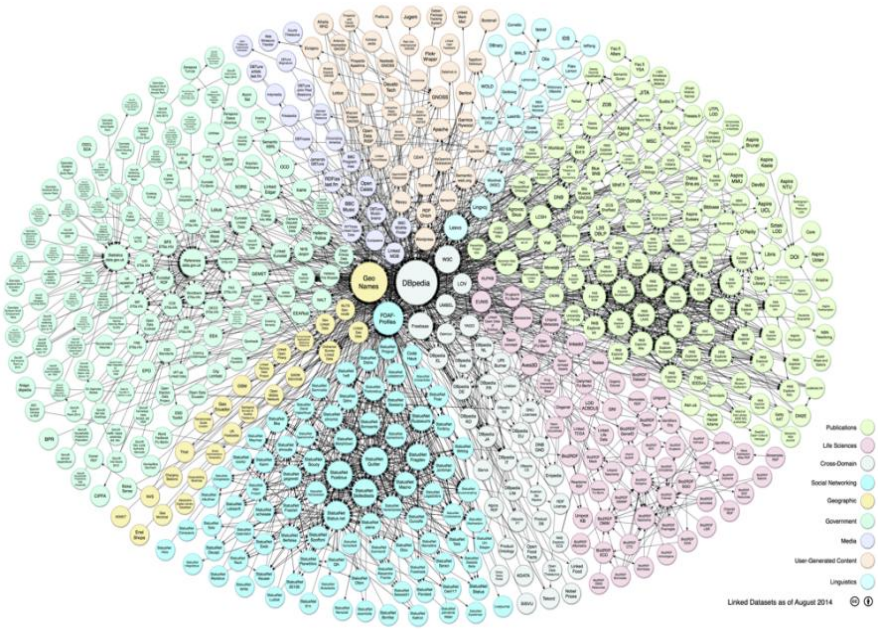
F. Michel, I3S laboratory

CrEDIBLE

Emerging web of data

- Web of data ⇨ publication/interlinking of open datasets
- Driven by data integration projects, e.g.:
  - Linking Open Data project: 1015 DS,
    83 Life Science DS,
    incl. BIO2RDF (35 DS) BIO2RDF
  - Neuroscience Information Framework NIF
    (12598 registry entries)

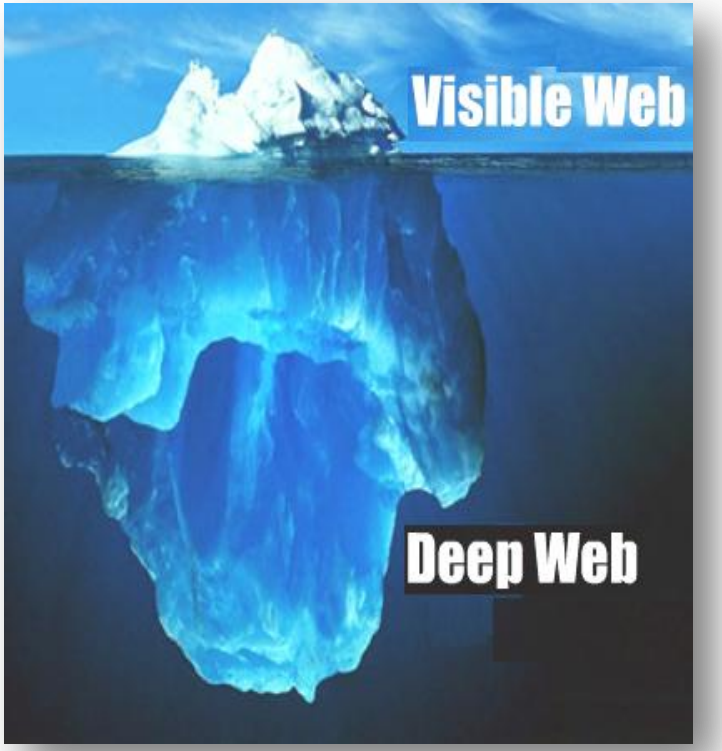
(Data: Oct. 2014)



Linked Datasets as of April 2014

Emerging web of data

- Need to access data from the Deep Web [1]
  - Strd./unstrd. data hardly indexed by search engines



- While LD projects link more and more DS, exponential data growth goes on
  - Various types of DBs: RBS, Native XML DB, NoSQL, LDAP directory, OODB...

[1] B. He, M. Patel, Z. Zhang, and K. C.-C. Chang. Accessing the deep web. Communications of the ACM, 50(5):94–101, 2007

The NoSQL trend

- Generic term encompassing different models
- NoSQL initially driven Big Data needs of major web players (G.A.F.A: Google, Apple, Facebook, Amazon + ...)
- But Big Data not longer the only use case: also used in many other situations
  - Lightweight, easy to deploy, flexible model, easy to code with
- Forecast[1]:
  - $3.4 Billion market in 2020
  - 21% growth/y 2015–2020 (CGAR)

1. http://www.marketresearchmedia.com/?p=568

Observations

- LD integration projects face the same questions:
  - How to translate ever more heterogeneous sources to RDF?
  - How to query the translated data? (ETL vs. Dynamic access)
- (Deep web) data keeps growing fast
- Migration from one-fits-all systems (RBDs) to multiple very different systems
- Need for a standardized approach to enable the mapping of heterogeneous data sources to RDF

Previous works

- Several works target specific types of data sources
  - XML, CSV/Spreadsheets, RBDs: R2RML W3C
- Integration frameworks
  - DataLift, RML
  - Commercial products: Asio Tool Suite, Anzo Data Collab. Server
- Current situation
  - No uniform mapping language for heterogeneous data bases

  - No uniform mapping language for heterogeneous data bases

  - NoSQL systems hardly addressed

R2RML – Relational To RDF Meta Language

- W3C recommendation, 2012
- Goals:
  - Describe mappings of relational entities to RDF
  - Reuse of existing ontologies
  - Operationalization not addressed
- How: *TriplesMaps* define how to generate RDF triples
  - 1 logical table ⇨ rows to process
  - 1 subject map ⇨ subject IRIs
  - N (predicate map-object map) couples
  - 1 opt. graph map ⇨ graph IRIs

Triples

R2RML

- Looks simple but very powerful
- Term maps:
  - Functions to generate RDF terms from cells
    - IRI, blank node, literal
  - 3 types:
    - Constant, e.g. target graph: `http://example.org/graph`
    - Column, e.g. a literal value: `NAME`
    - Template, e.g. subject IRI: `http://example/org/resource/{ID}`
- Handle cross-reference relationships
  - Use subjects of a TM as objects of another TM

R2RML

Study

Id	Acronym	CenterId
10	CAC2010	4

FK

Centre

Id	Name	address
4	Hospital X	...

Produced RDF:

```
<http://example.org/centre#4> a st:centre.
<http://example.org/study#10> a st:study;
    lang:has-for-name "CAC2010";
    role:has-for-agent-at <http://example.org/centre#4>.
```

R2RML mapping graph:

```
<#Centre>
    rr:logicalTable [ rr:tableName "Centre" ];
    rr:subjectMap [ rr:class st:centre;
        rr:template "http://example.org/centre#{Id}";
    ].

<#Study>
    rr:logicalTable [ rr:tableName "Study" ];
    rr:subjectMap [ rr:class st:study;
        rr:template "http://example.org/study#{Id}";
    ];
    rr:predicateObjectMap [
        rr:predicate lang:has-for-name;
        rr:objectMap [ rr:column "Acronym" ];
    ];
    rr:predicateObjectMap [
        rr:predicate role:has-for-agent-at;
        rr:objectMap [
            rr:parentTriplesMap <#Centre>;
                rr:joinCondition [
                    rr:child "CenterId";
                    rr:parent "Id"; ]; ]; ].
```

xR2RML

- Leverages R2RML, backward compatible
- Uniform language to describe mappings from most common types of DB to RDF
- Defines extensions to:
  - Allow **various query languages** and protocols
    - SQL, NoSQL DBs custom QL, HTTP GET query string, SPARQL...
  - Deal with **row-based data model** (RDB, column store, SPARQL result set), **“tree-like” data model**: collection, key-value associations (JSON, XML, Object, LDAP...)
  - Generate **RDF lists and containers** (bag, sequence, alternate) from structured values (collections, key-value associations)

xR2RML: Logical source and data element references

XML database, XQuery

```
<centres>
    <centre @Id="4">
        <name>Hostital X</name>
    </centre>
    <centre @Id="6">
        <name>Pontchaillou</name>
    </centre>
    ...
</centres>
```

xR2RML mapping

```
<#Centre>
  xrr:logicalSource [
    xrr:query "/centres/centre";


  ];
```

XQuery expression passed to the DB

rr: R2RML vocabulary
xrr: xR2RML vocabulary

xR2RML: Logical source

XML database, XQuery

```
<centres>
    <centre @Id="4">
        <name>Hostital X</name>
    </centre>
    <centre @Id="6">
        <name>Pontchaillou</name>
    </centre>
    ...
</centres>
```

rr: R2RML vocabulary
xrr: xR2RML vocabulary

xR2RML mapping

```
<#Centre>
  xrr:logicalSource [
    xrr:query "/centres/centre";


  ];
  rr:subjectMap [
    rr:class st:centre;
    rr:template
      "http://example.org/centre#{//centre/@Id}";
  ];
  rr:predicateObjectMap [
    rr:predicate lang:has-for-name;
    rr:objectMap [
      xrr:reference "//centre/name" ];
  ];
```

XPath expressions evaluated by the xR2RML processor

xR2RML: Data element references

XML database, XQuery

```
<centres>
    <centre @Id="4">
        <name>Hostital X</name>
    </centre>
    <centre @Id="6">
        <name>Pontchaillou</name>
    </centre>
    ...
</centres>
```

rr: R2RML vocabulary
xrr: xR2RML vocabulary

xR2RML mapping

```
<#Centre>
  xrr:logicalSource [
    xrr:query "/centres/centre";
    xrr:format xrr:XML;

  ];
  rr:subjectMap [
    rr:class st:centre;
    rr:template
      "http://example.org/centre#{//centre/@Id}";
  ];
  rr:predicateObjectMap [
    rr:predicate lang:has-for-name;
    rr:objectMap [
      xrr:reference "//centre/name" ];
  ];
```

Syntax of data => syntax of element references

xR2RML: Data element references

XML database, XQuery

```
<centres>
    <centre @Id="4">
        <name>Hostital X</name>
    </centre>
    <centre @Id="6">
        <name>Pontchaillou</name>
    </centre>
    ...
</centres>
```

```
rr: R2RML voca
xrr: xR2RML vo
```

xR2RML mapping

```
<#Centre>
  xrr:logicalSource [
    xrr:query "/centres/centre";
    xrr:format xrr:XML;

  ];
  rr:subjectMap [
    rr:class st:centre;
```

```
ntre#{//centre/@Id}";

r-name;

e/name" ];
```

Syntax of data => syntax of element references

Types of DB	xrr:format	xrr:reference rr:template
RDB, CSV, Column store, SPARQL result set...	xrr:Row	Column name
Native XML DB	xrr:XML	XPath expression
Document Store	xrr:JSON	JSONPath expression
...	...	...

xR2RML: embedded contents of mixed formats

Data with mixed-format content

Relational table “STAFF”, column “Name” contains JSON data:

...	Name	...
...	{ “FirstName”: “Bob”, “LastName: “Smith” }	...

xR2RML mapping

```
<#Centre>
  xrr:logicalSource [
    xrr:sourceName “STAFF";
  ];
  ...
  rr:predicateObjectMap [
    rr:predicate lang:fist-name;
    rr:objectMap [
      xrr:reference
        “Column(Name)/JSONPath($.FirstName)" ];
```

xrr:format	Syntax path constructor
xrr:Row	Column(), CSV(), TSV()
xrr:XML	XPath()
xrr:JSON	JSONPath()
...	...

xR2RML: multiple values vs. collection/container

MongoDB database (NoSQL document db)

```
{ "studyid": 10,
  "acronym": "CAC2010",
  "centres": [
    { "centreid": 4,
      "name": "Hostpial X" },
    { "centreid": 6,
      "name": "Pontchaillou" },
  ]
}
```

xR2RML mapping

```
<#Study>
  xrr:logicalSource [
    xrr:query '''db.studies.find(
      { studyid:{ $exists:true } })''';
    xrr:format xrr:JSON;
  ];
  rr:subjectMap [
    rr:class st:study;
    rr:template
      "http://example.org/study#{$.studyid}";
  ];
  rr:predicateObjectMap [
    rr:predicate st:involves;
    rr:objectMap [
      xrr:reference "$.centres.*.name" ];
  ];
```

Cartesian product between all term maps

Multiple values returned

Generated triples:

```
<http://example.org/study#10> st:involves "Hospital X".
<http://example.org/study#10> st:involves "Pontchaillou".
```

xR2RML: multiple values vs. collection/container

MongoDB database (NoSQL document db)

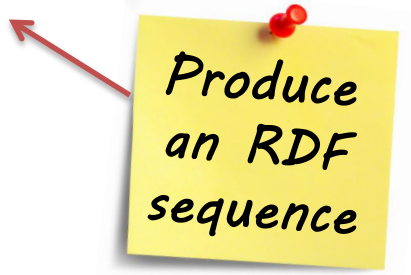
```
{ "studyid": 10,
  "acronym": "CAC2010",
  "centres": [
    { "centreid": 4,
      "name": "Hostpial X" },
    { "centreid": 6,
      "name": "Pontchaillou"}
  ]
}
```

Generated triples:

```
<http://example.org/study#10> st:involves
[ a rdf:Seq;
  rdf:_1 "Hospital X";
  rdf:_2 "Pontchaillou";
].
```

xR2RML mapping

```
<#Study>
  xrr:logicalSource [
    xrr:query '''db.studies.find(
      { studyid:{ $exists:true } })''';
    xrr:format xrr:JSON;
  ];
  rr:subjectMap [
    rr:class st:study;
    rr:template
      "http://example.org/study#{$.studyid}";
  ];
  rr:predicateObjectMap [
    rr:predicate st:involves;
    rr:objectMap [
      xrr:reference "$.centres.*.name" ];
      rr:termType xrr:RdfSeq;
  ];
```



xR2RML: multiple values vs. collection/container

MongoDB database (NoSQL document db)

```
{ "studyid": 10,
  "acronym": "CAC2010",
  "centres": [
    { "centreid": 4,
      "name": "Hostpial X" },
    { "centreid": 6,
      "name": "Pontchaillo
  ]
}
```

Generated triples:

```
<http://example.org/study
[ a rdf:Seq;
  rdf:_1 "Hospital X";
  rdf:_2 "Pontchaillou".
].
```

R2RML term types	Additional xR2RML term types
rr:IRI, rr:Literal, rr:BlankNode	xrr:RdfList, xrr:RdfSeq, xrr:RdfBag, xrr:RdfAlt

xR2RML mapping

```
<#Study>
  xrr:logicalSource [
    xrr:query '''db.studies.find(
      { studyid:{ $exists:true } })''';
    xrr:format xrr:JSON;
  ];
  rr:subjectMap [
    rr:class st:study;

                                    nple.org/study#{$.studyid}";

                                    ectMap [
                                    st:involves;

                                    ce "$.centres.*.name" ];
                                    xrr:RdfSeq;
```

Produce an RDF sequence

xR2RML: in depth parsing of structured values, produce nested collections/containers

MongoDB database (NoSQL document db)

```
{ "studyid": 10,
  "acronym": "CAC2010",
  "centres": [
    { "centreid": 4,
      "name": "Hostpial X",
      "doctors": ["John X", "Bob Y"]
    },
    { "centreid": 6,
      "name": "Pontchaillou",
      "doctors": ["Ted Z"]
    }
  ]
}
```

xR2RML mapping

```
<#Study>
  xrr:logicalSource [ ... ];
  rr:subjectMap [ ... ];

  rr:predicateObjectMap [
    rr:predicate st:examinators;
    rr:objectMap [
      xrr:reference "$.centres.*";
      rr:termType xrr:RdfList;
      xrr:nestedTermMap [
        xrr:reference "$.doctors.*";
        rr:termType xrr:RdfList;
        xrr:nestedTermMap [
          rr:datatype xsd:string;];
        ];
      ];
```

Nested term maps

Generated triples:

```
<http://example.org/study#10> st:examinators
    ( ( "John X"^^xsd:string  "Bob Y"^^xsd:string )
      ( "Ted Z"^^xsd:string )
    ).
```

xR2RML: in depth parsing of structured values, produce nested collections/containers

- From structured values (XML, JSON...): nested collections and key-value associations

  ⇨need to parse them in depth

- To RDF...

  ⇨need to generated nested lists/containers, and qualify members (data type, language tag...)

Generated triples:

```
<http://example.org/study#10> st:examinators
  ( ( "John X"^^xsd:string  "Bob Y"^^xsd:string )
    ( "Ted Z"^^xsd:string )
  ).
```

xR2RML: cross-references

MongoDB database (NoSQL document db)

Collection “studies”:

```
{ “studyid”: 10,
  “acronym”: “CAC2010”,
  “centres”: [ 4, 6 ]
}
```

Collection “centres”:

```
{ “centreid”: 4,
  “name”: “Hostpial X” },
{ “centreid”: 6,
  “name”: “Pontchaillou”}
```

Joint query between multiple-valued terms

xR2RML mapping

```
<#Centre>
  xrr:logicalSource [ ... ];
  rr:subjectMap [ ... ].

<#Study>
  xrr:logicalSource [ .. ];
  rr:subjectMap [ ... ];
  rr:predicateObjectMap [
    rr:predicate st:involves;
    rr:objectMap [
      rr:parentTriplesMap <#Centre>;
      rr:joinCondition [
        rr:child "$.centres.*";
        rr:parent “$.centreid"; ];
      rr:termType xrr:RdfSeq; ]; ].
```

Produce an RDF sequence

Generated triples:

```
<http://example.org/study#10> st:involves
[ a rdf:Seq;
  rdf:_1 <http://example.org/centre#4>;
  rdf:_2 <http://example.org/centre#6>;
].
```

Conclusions

- Data deluge keeps on ever faster
- Data not only in RDBs but also other kinds of DBs
- There exists languages to map some types of DBs to RDF (R2RML)
- There exists integration frameworks with per-source adaptors
- There is no common mapping language able to adapt to any (most) types of databases

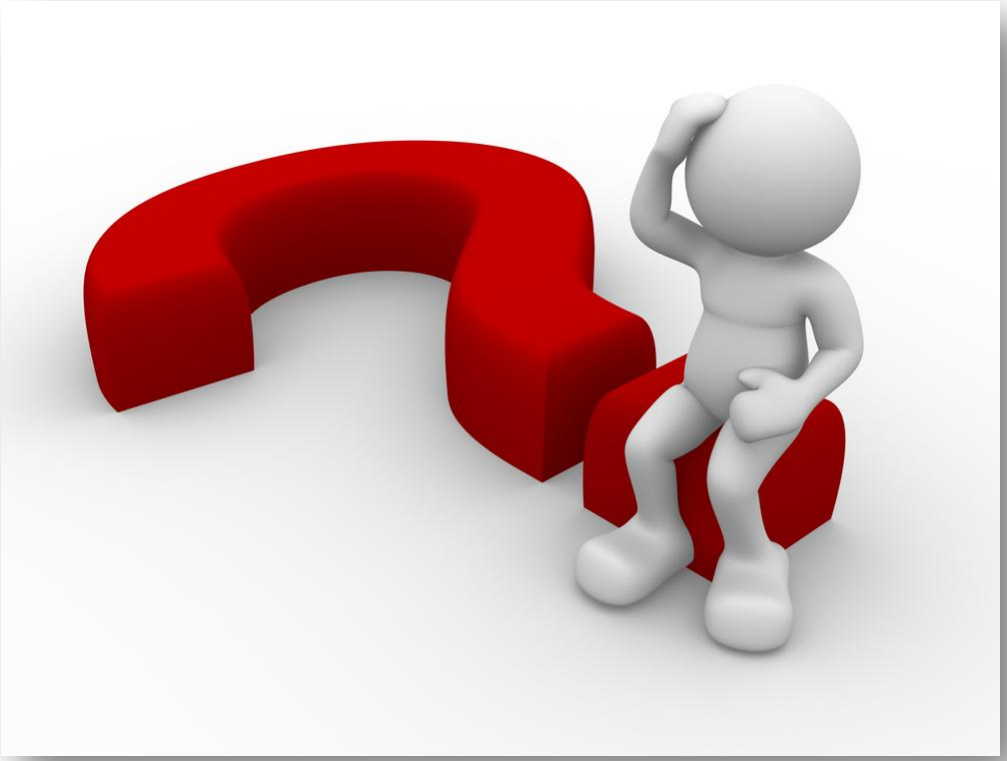
Conclusions

- xR2RML:
  - A language to map most common types of DB (RDB, NoSQL, XML...) to RDF
  - Extensible to other data models and query languages
    - Just a matter of software dev., no change required in the language
  - Allows content with mixed formats
  - Can flexibly produce RDF collections and containers
  - Modular: reuse of mapping parts across different DBs
- More details:

F. Michel, L. Djimenou, C. Faron-Zucker, and J. Montagnat. xR2RML: Non-Relational Databases to RDF Mapping Language. Research report. ISRN I3S/RR 2014-04-FR. http://hal.archives-ouvertes.fr/hal-01066663

Perspectives

- Currently developing a prototype implementation
  - Based on Morph-RDB
  - Support of RDBs with embedded XML or JSON values
  - Support one NoSQL document DB (MongDB or CouchDB)
- Prototype extensions:
  - One XML DB (BaseX or eXists)
  - One extensible column store (Cassandra)
- Open questions
  - What about graph stores?
    - Can they fit in the parsing of nested values defined for XML/JSON?
  - What about key-value stores?
    - Queriable through APIs but hardly provide query languages.



Contacts:
Franck Michel
Johan Montagnat
Catherine Faron-Zucker