Tutorial on Knowledge Graph Construction

Ana Iglesias-Molina (UPM), Dylan Van Assche (Ghent U.), Enrique Iglesias (L3S Group), Julián Arenas-Guerro (UPM), Mario Scrocca (CEFRIEL), Markus Schröder (DFKI), Samaneh Jozashoori (TIB), Sergio Rodríguez-Méndez (ANU), Umutcan Şimşek (LFUI)

Organizers: **David Chaves-Fraga**, Anastasia Dimou

public-kg-construct@w3.org

@kgc_workshop

Funded by the European Union

Agenda

Morning:

- Introduction to KG construction (09:00-10:30)
- Coffee Break (10:30 - 11:00)
- Mapping Editors and Declarative Functions (11:00 - 12:30)

Afternoon:

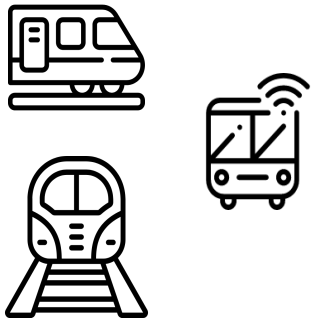
- KGC with RML: SDM-RDFizer & RocketRML (14:00-15.30)
- Coffee Break (15:30 - 16:00)
- KGC with RML: Morph-KGC and RMLStreamer (16:00-17:30)

Technical Requirements

Docker

https://docs.docker.com/engine/install/

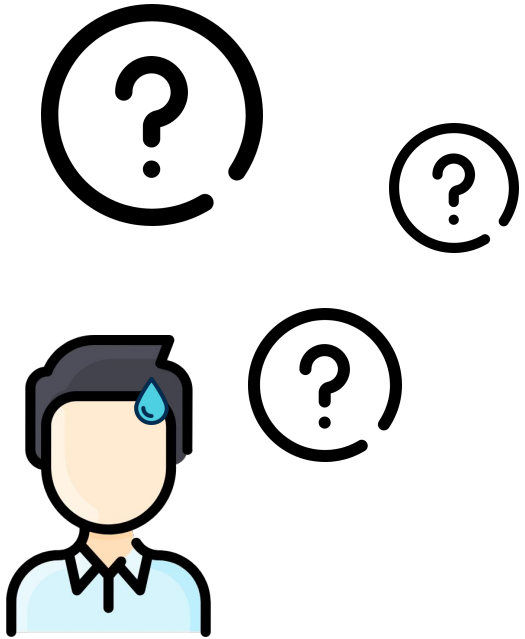
We arrive in Crete, and as knowledge engineers, we are in charge of handling Greek's **transport data** to enhance sustainable mobility



* Adapted from KGC with Declarative Mapping Rules Tutorial @ ISWC 2020 (Iglesias-Molina et al.)

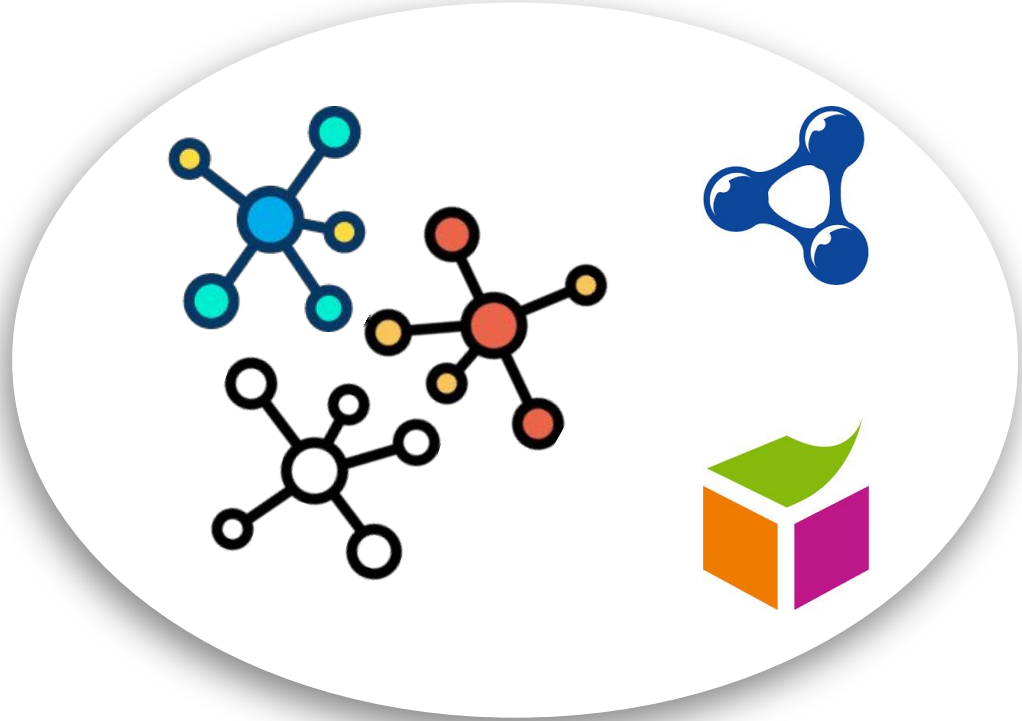
However, real transport data is **tricky**:

- **Heterogeneity**
- **Complexity**
- Data **not clean** and **normalized**
- Values, such as dates, have to be **correctly interpreted**
- Has to be **understandable** by anyone



* Adapted from KGC with Declarative Mapping Rules Tutorial @ ISWC 2020 (Iglesias-Molina et al.)

So he decided to build a **Knowledge Graph**, the solution to represent clearly the heterogeneous and make it **clear**, **accessible** and **queriable**



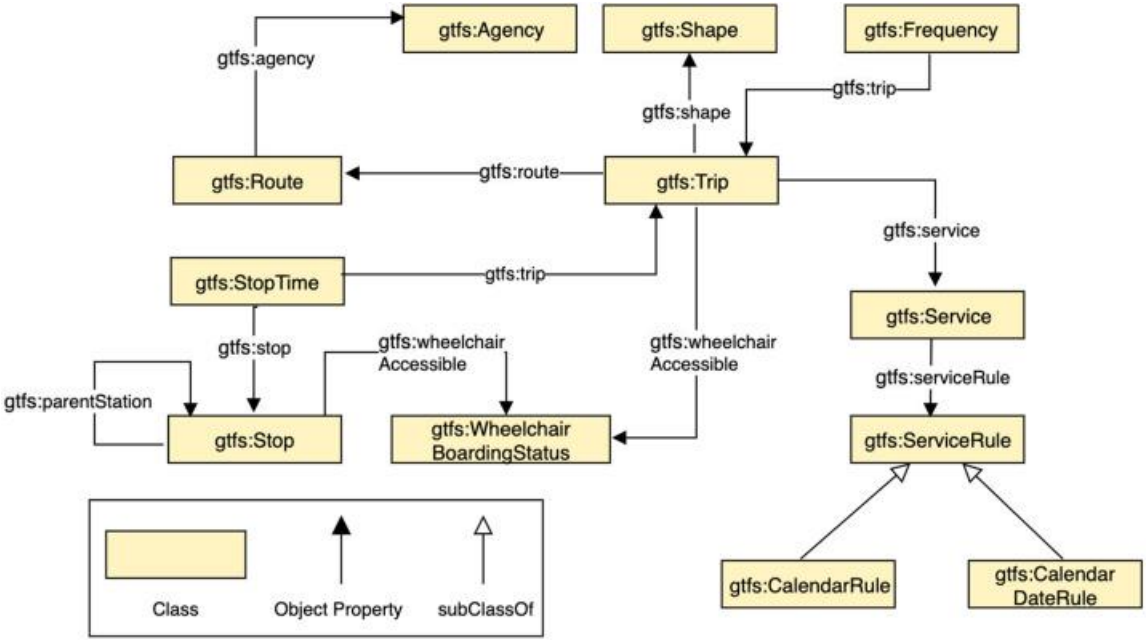
* Adapted from KGC with Declarative Mapping Rules Tutorial @ ISWC 2020 (Iglesias-Molina et al.)

The **requirements** needed for this data integration pipeline are:

- **Ontology** that models the transport domain
- **Standard declarative mapping rules**:
  - Flexibility
  - Adaptability
  - Maintainability
  - Readability
  - Reproducibility
- Ensure **materialized** KG
- **Avoid** at maximum ad-hoc and **manual** steps
- **Efficient** generation

* Adapted from KGC with Declarative Mapping Rules Tutorial @ ISWC 2020 (Iglesias-Molina et al.)

He has been given the data, so it's time to choose a suitable ontology: the **Linked General Transit Feed Specification (LinkedGTFS)**



- To be aligned with the GTFS spec[1] (de-facto standard for open transport data)
- 13 input sources (in different potential formats)
- No information about data size, join selectivity, etc.

[1] https://developers.google.com/transit/gtfs

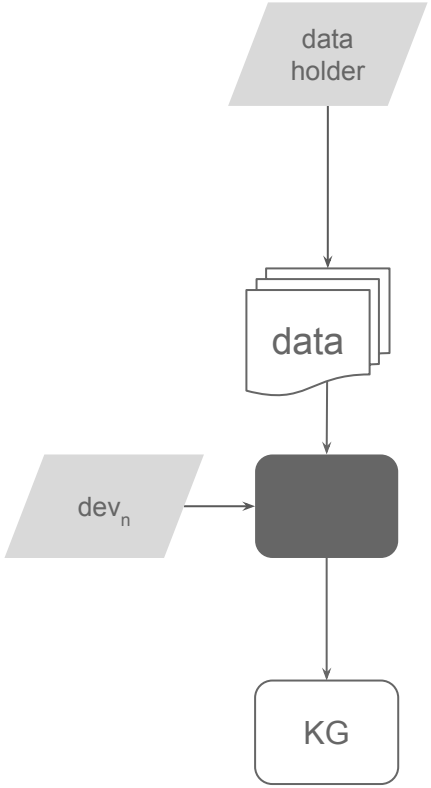
GTFS-Madrid-Bench: A benchmark for virtual knowledge graph access in the transport domain Chaves-Fraga et al. JoWS 2020

Introduction to Declarative Knowledge Graph Construction

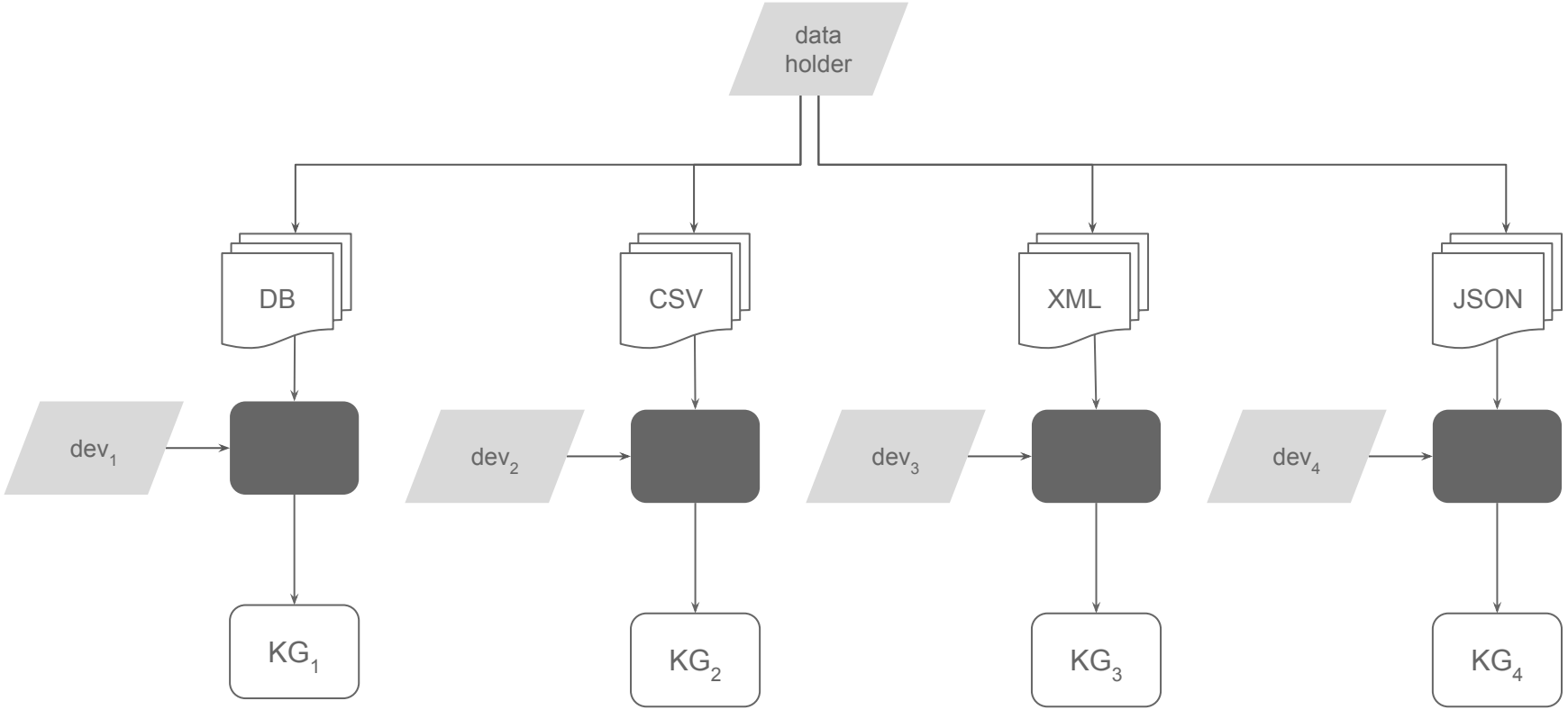
Anastasia Dimou
David Chaves-Fraga

anastasia.dimou@kuleuven.be, david.chaves@upm.es

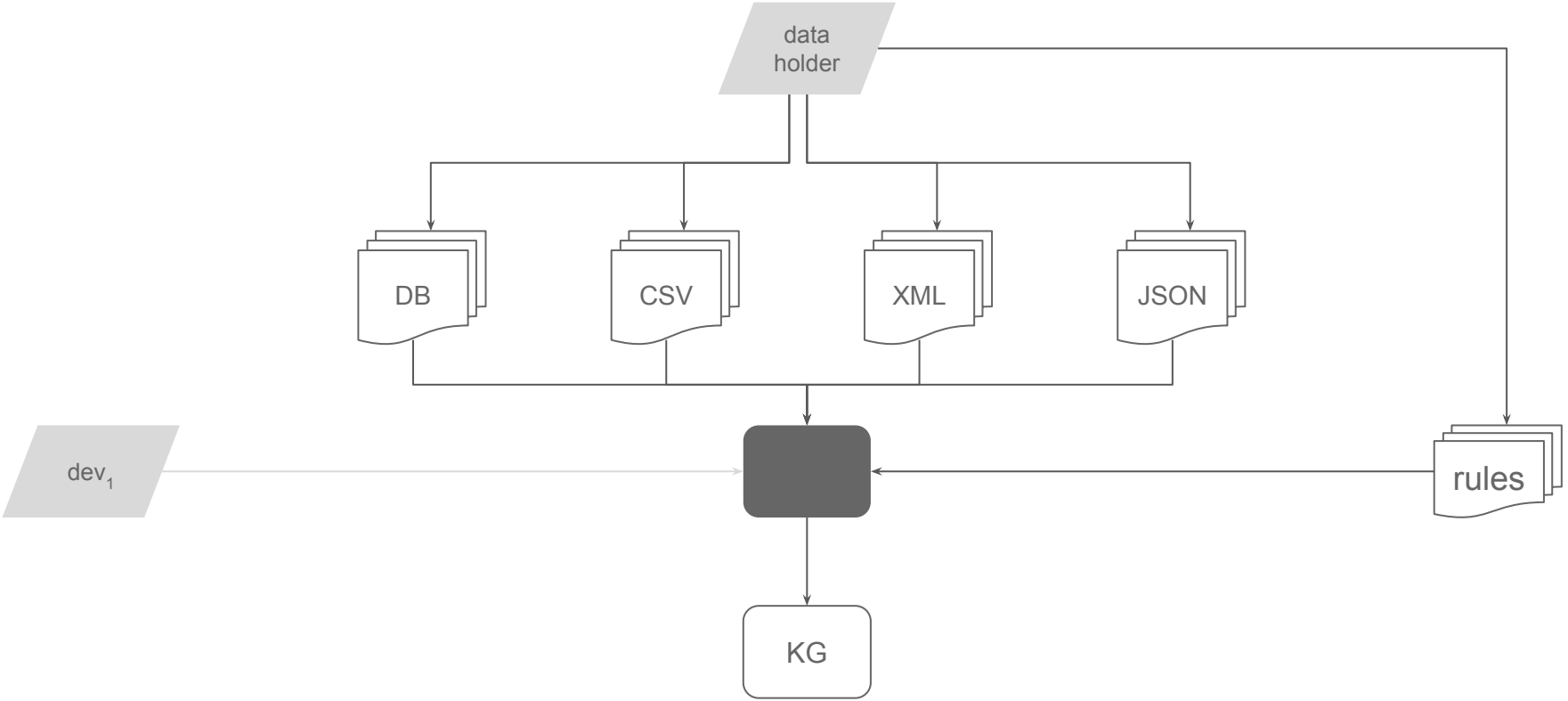
@natadimou, @dchavesf



custom dedicated script for a data owner’s data
(-) new development cycle every time a modification is needed



dedicated tool for certain format
(+) great solution if a data owner has data only in a certain format
(-) learn and maintain multiple tools if a data owner has data in different formats
(-) post-processing step to integrate

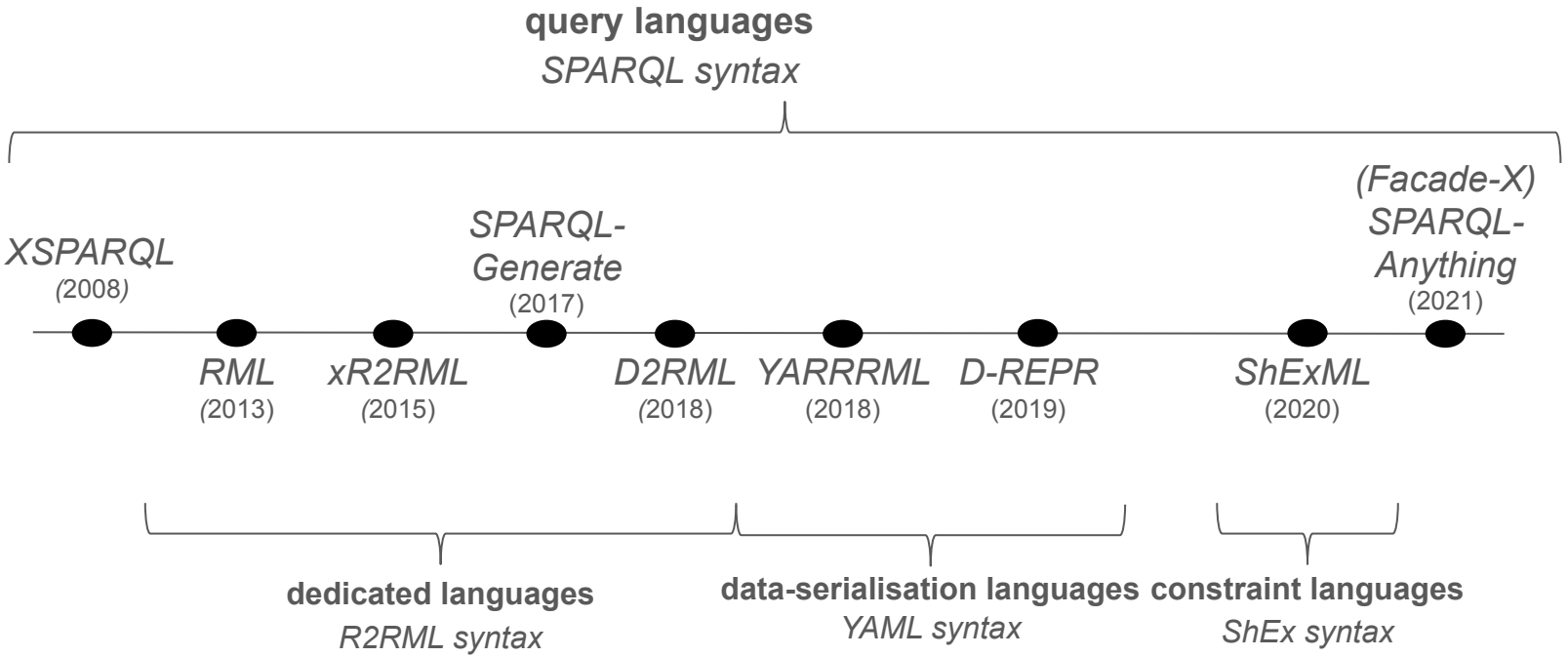


a tool for all data formats

(+) learn and maintain a single tool

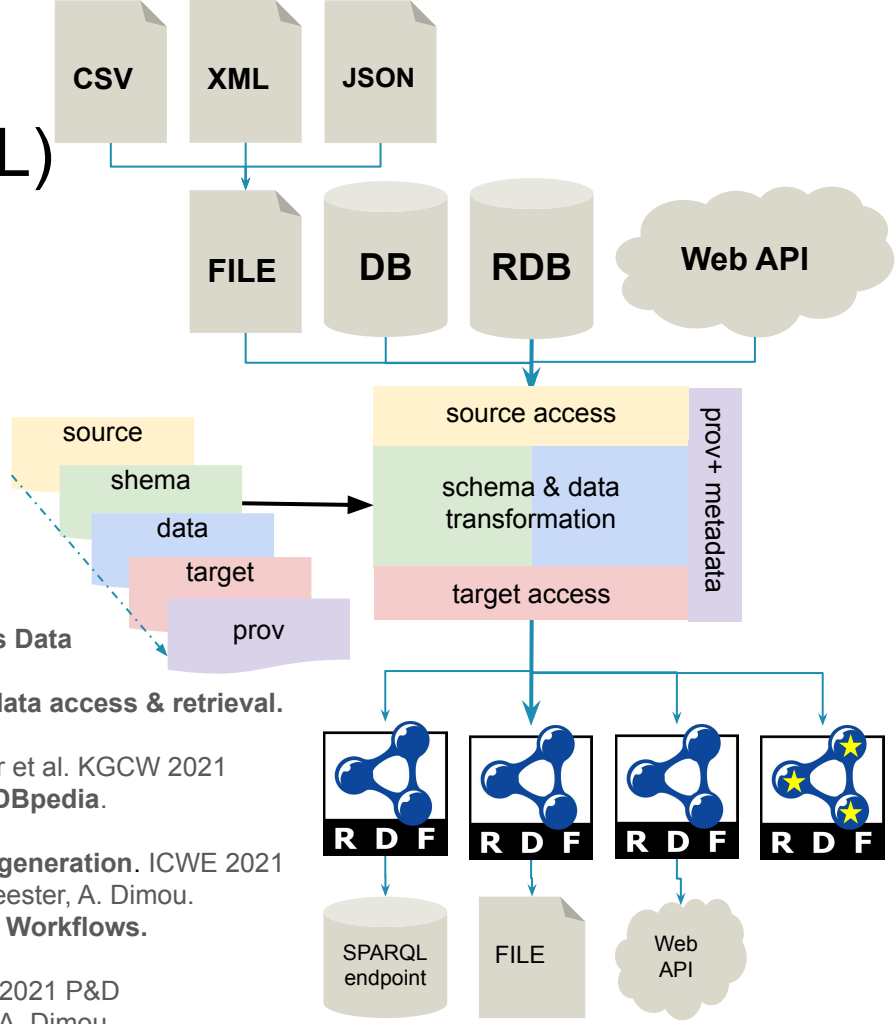
(+) configure the rules that define how a KG is generated

Declarative mapping languages - *schema* transformations



Declarative RDF graph generation from heterogeneous (semi-)structured data: a Systematic Literature Review.
D. Van Assche, T. Delva, G. Haesendonck, P. Heyvaert, B. De Meester, A. Dimou. (JWS under major revision)

RDF Mapping Language (RML)

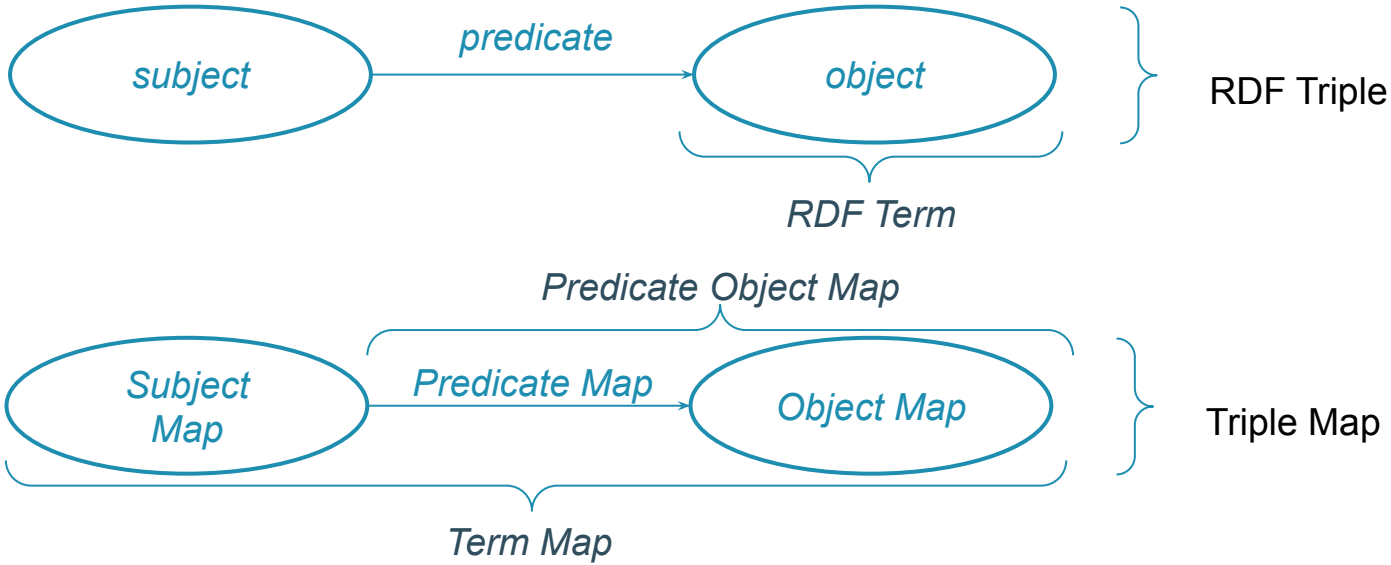


RML: A Generic Language for Integrated RDF Mappings of Heterogeneous Data
A. Dimou et al. LDOW 2014
Machine-interpretable dataset & service descriptions for heterogeneous data access & retrieval.
A. Dimou et al. SEMANTICS 2015
Mapping Spreadsheets to RDF: Supporting Excel in RML. Markus Schröder et al. KGCW 2021
Declarative data transformations for Linked Data generation: the case of DBpedia.
B.De Meester et al. ESWC 2017
Leveraging Web of Things W3C recommendations for knowledge graphs generation. ICWE 2021
D. Van Assche, G.Haesendonck, G. De Mulder, T. Delva, P. Heyvaert, B. De Meester, A. Dimou.
Automated Metadata Generation for Linked Data Generation & Publishing Workflows.
A. Dimou et al. LDOW 2016
RML-star: A declarative mapping language for RDF-star generation ISWC 2021 P&D
T. Delva, J. Arenas-Guerrero, A. Iglesias-Molina, O. Corcho, D. Chaves-Fraga, A. Dimou.

rank	name	nationality	mark	notes
1	Anzhelika Sidorova	Russia	4.95	WL,PB
2	Sandi Morris	USA	4.90	SB
3	Katerina Stefanidi	Greece	4.85	SB
4	Holly Bradshaw	UK	4.80	-
5	Alysha Newman	Canada	4.80	-
6	Angelica Bengtsson	Sweden	4.80	NR

Input:
- (semi-)structured raw data
- ontology/vocabulary terms
- mapping rules

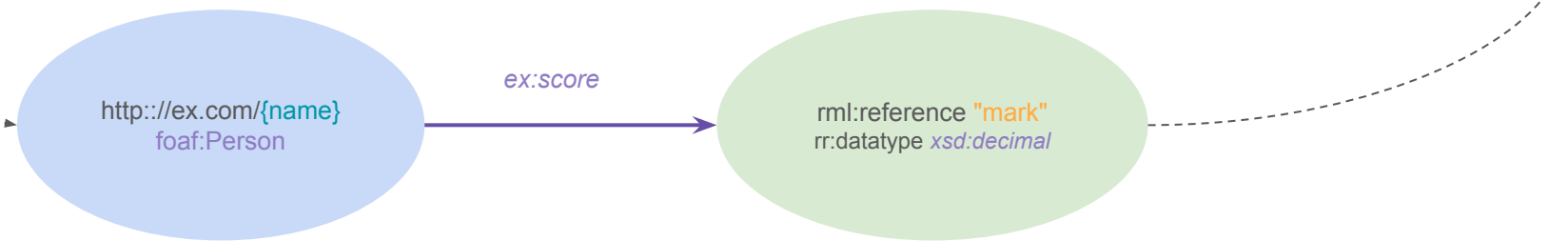
Output:
- RDF graphs



rank	name	nationality	mark	notes
1	Anzhelika Sidorova	Russia	4.95	WL,PB
2	Sandi Morris	USA	4.90	SB
3	Katerina Stefanidi	Greece	4.85	SB
4	Holly Bradshaw	UK	4.80	-
5	Alysha Newman	Canada	4.80	-
6	Angelica Bengtsson	Sweden	4.80	NR

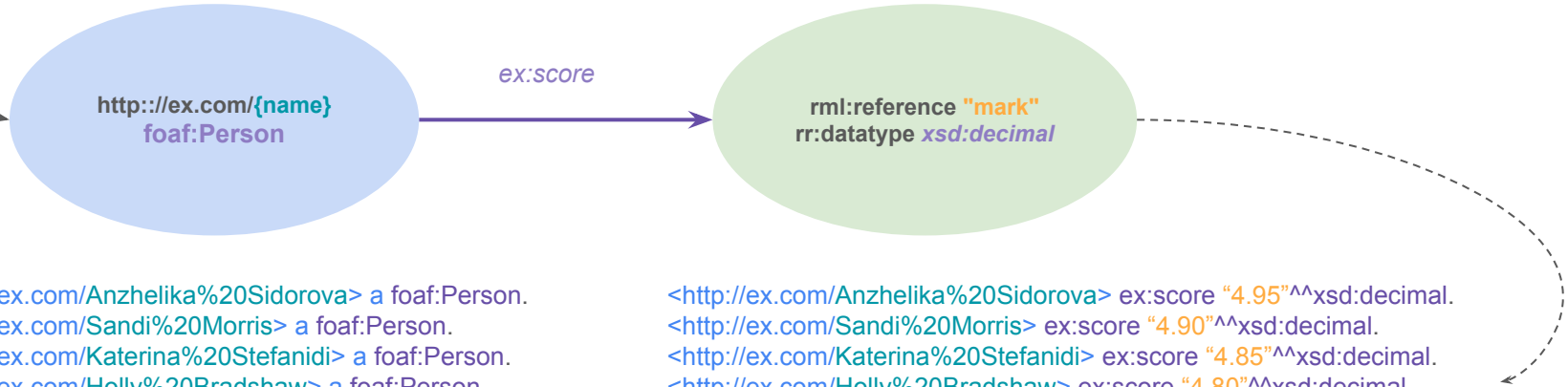
```
<#TriplesMap_1> [
  rr:subjectMap […];
  rr:predicateObjectMap [
    rr:predicateMap […];
    rr:objectMap […]; ] ].
```

<http://ex.com/Anzhelika%20Sidorova> ex:score “4.95”^^xsd:decimal.
<http://ex.com/Sandi%20Morris> ex:score “4.90”^^xsd:decimal.
<http://ex.com/Katerina%20Stefanidi> ex:score “4.85”^^xsd:decimal.
<http://ex.com/Holly%20Bradshaw> ex:score “4.80”^^xsd:decimal.
<http://ex.com/Alysha%20Newman> ex:score “4.80”^^xsd:decimal.
<http://ex.com/Angelica%20Bengtsson> ex:score “4.80”^^xsd:decimal.



rank	name	nationality	mark	notes
1	Anzhelika Sidorova	Russia	4.95	WL,PB
2	Sandi Morris	USA	4.90	SB
3	Katerina Stefanidi	Greece	4.85	SB
4	Holly Bradshaw	UK	4.80	-
5	Alysha Newman	Canada	4.80	-
6	Angelica Bengtsson	Sweden	4.80	NR

```
<#TriplesMap_1> [
  rr:subjectMap [
    rr:template "http:://ex.com/{name}";
    rr:class foaf:Person; ];
  rr:predicateObjectMap [
    rr:predicateMap [rr:constant ex:score];
    rr:objectMap [ rml:reference "mark";
                   rr:datatype xsd:decimal]; ] ].
```

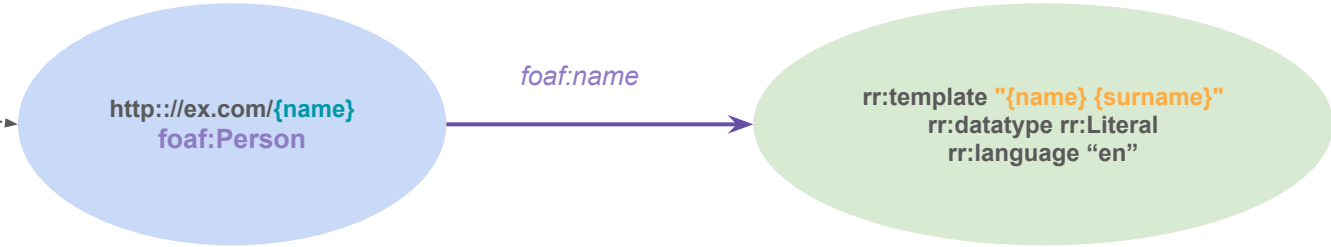


<http://ex.com/Anzhelika%20Sidorova> a foaf:Person.
<http://ex.com/Sandi%20Morris> a foaf:Person.
<http://ex.com/Katerina%20Stefanidi> a foaf:Person.
<http://ex.com/Holly%20Bradshaw> a foaf:Person.
<http://ex.com/Alysha%20Newman> a foaf:Person.
<http://ex.com/Angelica%20Bengtsson> a foaf:Person.

<http://ex.com/Anzhelika%20Sidorova> ex:score “4.95”^^xsd:decimal.
<http://ex.com/Sandi%20Morris> ex:score “4.90”^^xsd:decimal.
<http://ex.com/Katerina%20Stefanidi> ex:score “4.85”^^xsd:decimal.
<http://ex.com/Holly%20Bradshaw> ex:score “4.80”^^xsd:decimal.
<http://ex.com/Alysha%20Newman> ex:score “4.80”^^xsd:decimal.
<http://ex.com/Angelica%20Bengtsson> ex:score “4.80”^^xsd:decimal.

rank	name	surname	nationality	mark	notes
1	Anzhelika	Sidorova	Russia	4.95	WL,PB
2	Sandi	Morris	USA	4.90	SB
3	Katerina	Stefanidi	Greece	4.85	SB
4	Holly	Bradshaw	UK	4.80	-
5	Alysha	Newman	Canada	4.80	-
6	Angelica	Bengtsson	Sweden	4.80	NR

```
<#TriplesMap_1> [
  rr:subjectMap [
    rr:template "http:://ex.com/{name}";
    rr:class foaf:Person; ];
  rr:predicateObjectMap [
    rr:predicateMap [rr:constant foaf:name];
    rr:objectMap [ rr:template "{name} {surname}";
                   rr:termType rr:Literal;
                   rr:language "en"] ] ].
```




```
<http://ex.com/Anzhelika%20Sidorova> a foaf:Person.
<http://ex.com/Sandi%20Morris> a foaf:Person.
<http://ex.com/Katerina%20Stefanidi> a foaf:Person.
<http://ex.com/Holly%20Bradshaw> a foaf:Person.
<http://ex.com/Alysha%20Newman> a foaf:Person.
<http://ex.com/Angelica%20Bengtsson> a foaf:Person.
```

```
<http://ex.com/Anzhelika%20Sidorova> foaf:name “Anzhelika Sidorova”@en.
<http://ex.com/Sandi%20Morris> foaf:name "Sandi Morris"@en.
<http://ex.com/Katerina%20Stefanidi> foaf:name "Katerina Stefanidi"@en.
<http://ex.com/Holly%20Bradshaw> foaf:name "Holly Bradshaw"@en.
<http://ex.com/Alysha%20Newman> foaf:name "Alysha Newman"@en .
<http://ex.com/Angelica%20Bengtsson> foaf:name "Angelica Bengtsson"@en .
```

rank	name	surname	nationality	mark	notes
1	Anzhelika	Sidorova	Russia	4.95	WL,PB
2	Sandi	Morris	USA	4.90	SB
3	Katerina	Stefanidi	Greece	4.85	SB
4	Holly	Bradshaw	UK	4.80	-
5	Alysha	Newman	Canada	4.80	-
6	Angelica	Bengtsson	Sweden	4.80	NR

```
<countries>
  <country continent="Europe">
    <country_abb>GR</country_abb>
    <country_name country_language="en">Greece</country_name>
    <country_name country_language="nl">Griekenland</country_name>
  </country>
  <country continent="Europe">
    <country_abb>UK</country_abb>
    <country_name country_language="en">United Kingdom</country_name>
    <country_name country_language="nl">Verenigd Koninkrijk</country_name>
  </country>
  <country continent="America">
    <country_abb>CA</country_abb>
    <country_name country_language="en">Canada</country_name>
    <country_name country_language="nl">Canada</country_name>
  </country>
...
</countries>
```

```
<#TriplesMap_1> [
  rr:predicateObjectMap [
    rr:predicateMap [rr:constant ex:country];
    rr:objectMap [ rr:parentTriplesMap <#TriplesMap_2>;
                   rr:joinCondition [
                              rr:parent "country_name";
                              rr:child "nationality"]]]].
```

```
<#TriplesMap_2> [
  rml:logicalSource [
    rml:source "countries.xml";
    rml:referenceFormulation ql:XPath;
    rml:iterator "countries/country" ];
  rr:subjectMap [
    rr:template "http:://ex.com/{country_abb}"; ].
```

<http://ex.com/Anzhelika%20Sidorova> ex:country <http://ex.com/RU>.
<http://ex.com/Sandi%20Morris> ex:countrly <http://ex.com/US>.
<http://ex.com/Katerina%20Stefanidi> ex:country <http://ex.com/EL>.
<http:://ex.com/Holly%20Bradshaw> ex:country <http://ex.com/UK>.
<http:://ex.com/Alysha%20Newman> ex:country <http://ex.com/CA>.
<http:://ex.com/Angelica%20Bengtsson> ex:country <http://ex.com/SE>.

User Interfaces

Matey https://github.com/rmlio/matey

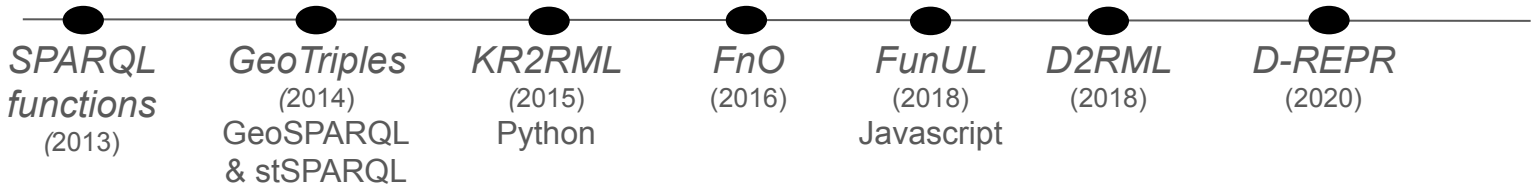
Mapeathor https://morph.oeg.fi.upm.es/tool/mapeathor, https://github.com/oeg-upm/morph-website

RMLEditor https://app.rml.io/rmleditor/, https://rml.io/tools/rmleditor/, https://github.com/RMLio/rmleditor-ce

Map-On http://semanco-tools.eu/map-on, https://github.com/arc-lasalle/Map-On

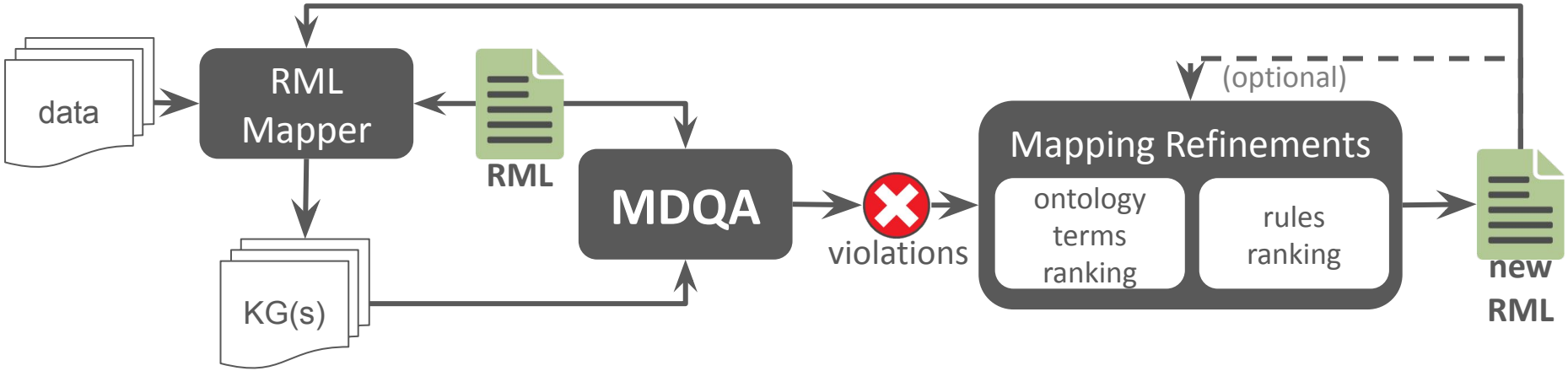
RMLx Visual Editor http://pebbie.org/mashup/rml

Declarative mapping languages - *data* transformations

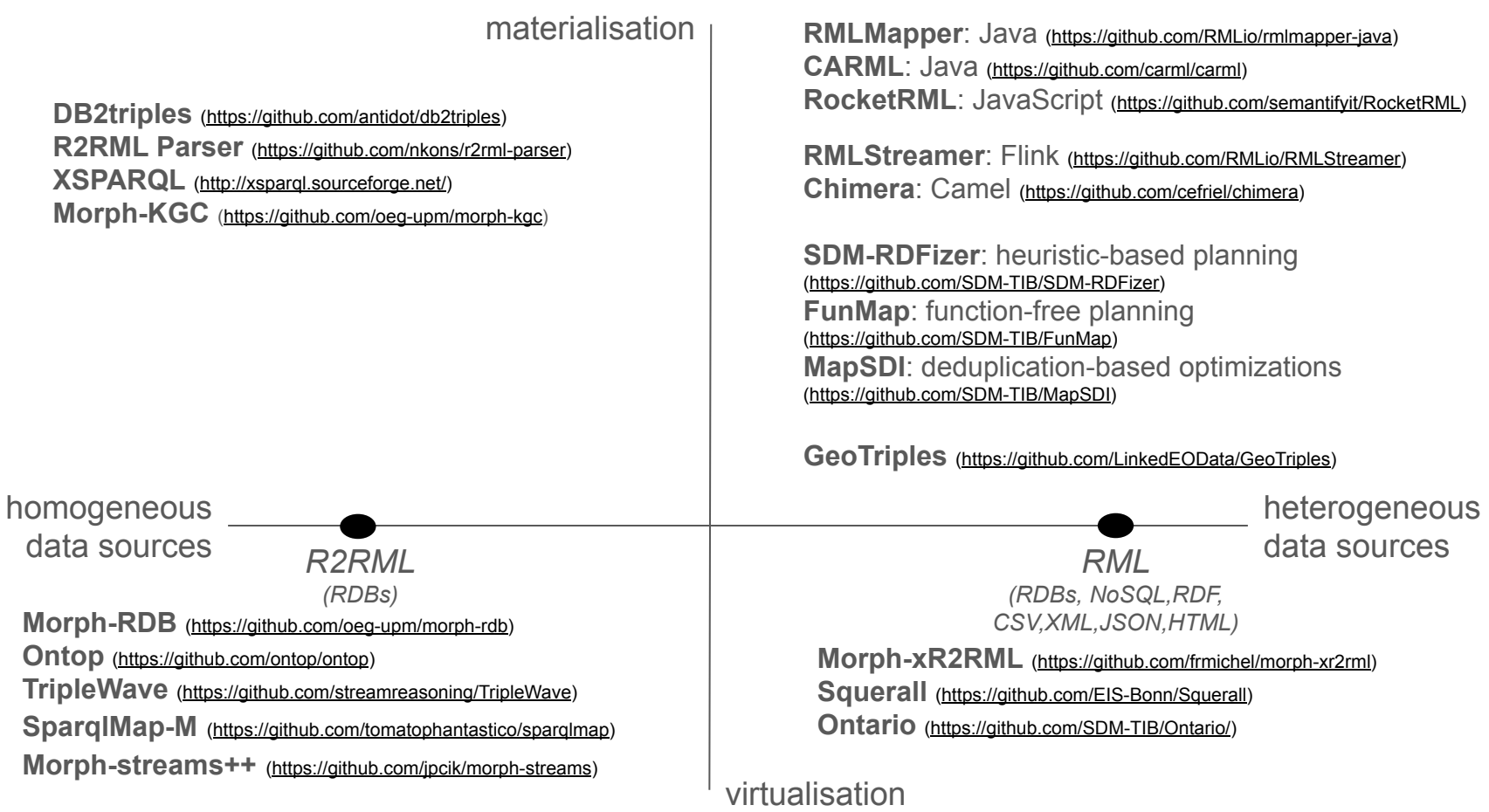


Declarative RDF graph generation from heterogeneous (semi-)structured data: a Systematic Literature Review.
D. Van Assche, T. Delva, G. Haesendonck, P. Heyvaert, B. De Meester, A. Dimou. (JWS under major revision)

Validation



Assessing and Refining Mappings to RDF to Improve Dataset Quality. A. Dimou et al. ISWC 2015
Rule-driven inconsistency resolution for knowledge graph generation rules P. Heyvaert A. Dimou, B. De Meester, R. Verborgh. SWJ 2019



Declarative RDF graph generation from heterogeneous (semi-)structured data: a Systematic Literature Review.
D. Van Assche, T. Delva, G. Haesendonck, P. Heyvaert, B. De Meester, A. Dimou. (JWS under review after major revision)

Test Cases - Implementation Reports - Benchmarks

Choose yourself the best tool for your needs!

http://rml.io/test-cases/
http://rml.io/implementation-report/

Test Case	RMLMapper	CARML	RocketRML	SDM-RDFizer	RMLStreamer	Chimera	Morph-KGC
RMLTC0000-CSV	passed	passed	passed	passed	passed	passed	passed
RMLTC0000-JSON	passed	passed	passed	passed	passed	passed	failed
RMLTC0000-MYSQL	passed	inapplicable	inapplicable	passed	inapplicable	inapplicable	passed
RMLTC0000-POSTGRESQL	passed	inapplicable	inapplicable	passed	inapplicable	inapplicable	passed
RMLTC0000-SPARQL	passed	inapplicable	inapplicable	inapplicable	inapplicable	inapplicable	inapplicable
RMLTC0000-SQLSERVER	passed	inapplicable	inapplicable	passed	inapplicable	inapplicable	inapplicable

Conformance test-cases for the RDF Mapping Language. P. Heyvaert, D. Chaves-Fraga, F. Priyatna, O. Corcho, E. Mannens, R. Verborgh, A. Dimou. KGSWC2019

Benchmarks:

GTFS: evaluate tools generating RDF graphs with RML mapping rules (https://github.com/oeg-upm/gtfs-bench)

RODI: test the quality of (semi-)automatically generated mapping rules (https://github.com/chrpin/rodi)

RODI: Benchmarking Relational-to-Ontology Mapping Generation Quality. Pinkel et al. SWJ 2016
GTFS-Madrid-Bench: A benchmark for virtual knowledge graph access in the transport domain Chaves-Fraga et al. JWS 2020

YARRRML

Dylan Van Assche

dylan.vanassche@ugent.be

@DylanVanAssche

https://dylanvanassche.be

Mapping rules are not human-friendly

RDF is great for machines but too verbose for humans

A mistake in your RDF syntax is hard to spot if you are not familiar with RDF

Harder to read than JSON or YAML

```
@prefix rr: <http://www.w3.org/ns/r2rml#>.
@prefix rml: <http://semweb.mmlab.be/ns/rml#>.
@prefix ql: <http://semweb.mmlab.be/ns/ql#>.
@prefix transit: <http://vocab.org/transit/terms/>.
@prefix xsd: <http://www.w3.org/2001/XMLSchema#>.
@prefix wgs84_pos:
<http://www.w3.org/2003/01/geo/wgs84_pos#>.
@base <http://example.com/ns#>.

<#AirportMapping> a rr:TriplesMap;
  rml:logicalSource [
    rml:source "Airport.csv" ;
    rml:referenceFormulation ql:CSV
  ];
  rr:subjectMap [
    rr:template "http://airport.example.com/{id}";
    rr:class transit:Stop
  ];
  rr:predicateObjectMap [
    rr:predicate transit:route;
    rr:objectMap [
      rml:reference "stop";
      rr:datatype xsd:int
    ]
  ];
.
```

More human-friendly mapping rules?

YAML is widely known among humans and is integrated in developer tools

YAML is easier to write than RDF

https://yaml.org

RML mapping rules in RDF Turtle

RML is widely used, but hard to write if not familiar with Turtle & RML

https://rml.io/spec

YARRRML

https://rml.io/yarrrml/spec

YARRRML is a human-friendly representation of mapping rules

YAML-based
YARRRML is a subset of YAML

Human-friendly representation
Human-friendly representation of RML mapping rules

Specification
YARRRML has its own specification

https://rml.io/yarrrml/spec/

```
<#TriplesMap_1> [
  rml:logicalSource [
    rml:source "poleVaulters.csv";
    rml:referenceFormulation ql:CSV; ]; ]
  rr:subjectMap [
    rr:template "http://ex.com/{name}"; ];
  rr:predicateObjectMap [
    rr:predicateMap [rr:constant ex:score];
    rr:objectMap [ rml:reference "mark";
                   rr:datatype xsd:decimal]; ];
  rr:predicateObjectMap [
    rr:predicateMap [rr:constant foaf:name];
    rr:objectMap [ rml:reference "name"; rr:language "en"]; ];
  rr:predicateObjectMap [
    rr:predicateMap [rr:constant ex:country];
    rr:objectMap [ rr:parentTriplesMap <#TriplesMap_2>;
                   rr:joinCondition [
                     rr:parent "country_name";
                     rr:child "nationality"] ]; ] ].
  <#TriplesMap_2> [
  rml:logicalSource [
    rml:source "countries.xml";
    rml:referenceFormulation ql:XPath;
    rml:iterator "countries/country" ];
  rr:subjectMap [
    rr:template "http://ex.com/{country_abb}";
    rr:graphMap [ rr:constant ex:CountryGraph ]; ].
```

YARRRML

https://rml.io/yarrrml/

```
mapping:
  person:
    sources:
      - [poleVaulters.csv~csv]
    subjects:
     - value: "http://ex.com/{name}"
    predicateobjects:
      - [ex:score, $(mark), xsd:decimal]
      - [foaf:name, $(name), en~lang]
      - [foaf:name, $(name) $(surname), en~lang]
      - predicates: ex:country
        objects:
          - mapping: country
            condition:
              function: equal
              parameters:
                - [str1, $(nationality), s]
                - [str2, $(country_name), o]
  country:
    subjects: http://ex.com/{country_abb}
```

Predicate Map

YARRRML is a human-friendly representation of mapping rules

Tooling
yarrrml-parser converts YARRRML
from and to RML. Matey provides a browser editor

Compatible
Works with any RML processor such as RMLMapper,
RMLStreamer, SDM-RDFizer, RocketRML, etc.

Battle-tested
Used in several projects, with more 1000 YARRRML mappings

https://github.com/rmlio/yarrrml-parser
https://rml.io/yarrrml/matey/

Matey is a browser application to write & test YARRRML rules

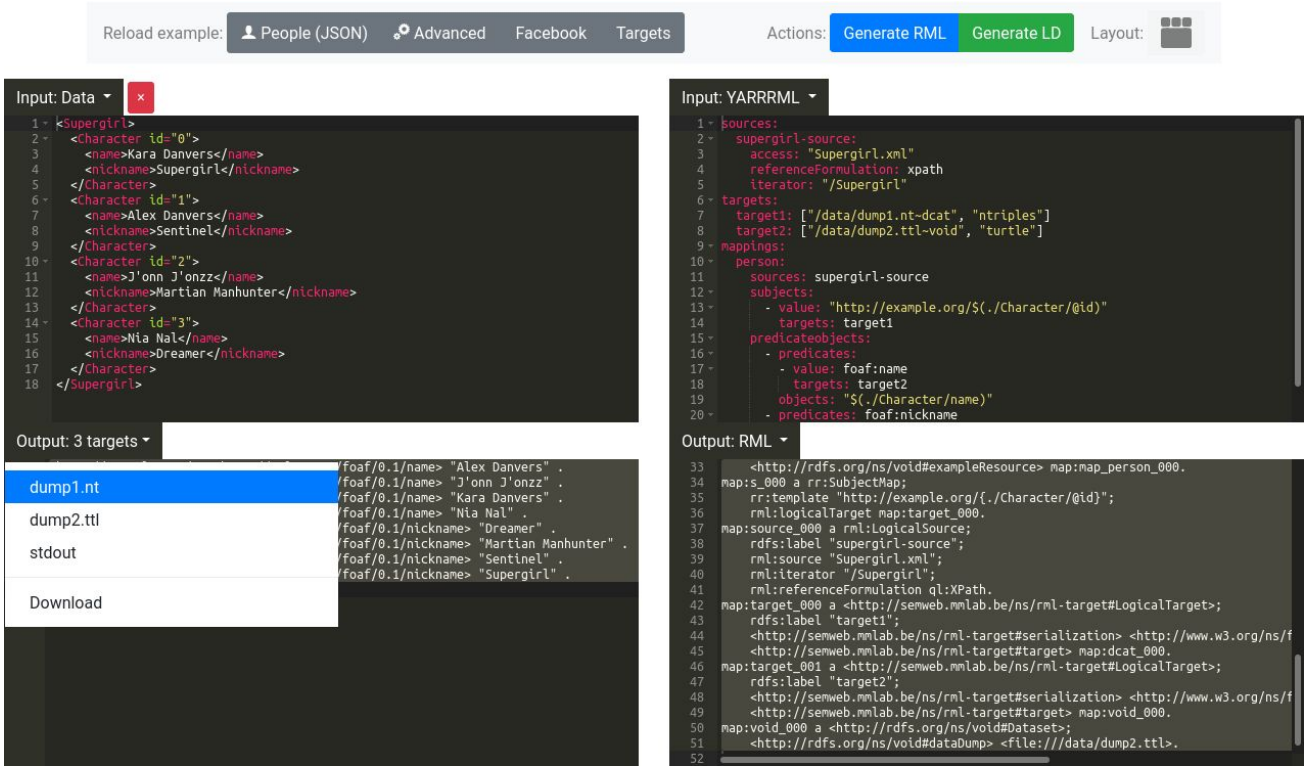
YARRRML editor
Matey is a YARRRML editor in your browser

Same tooling
Uses yarrrml-parser and RMLMapper behind the scenes,
no surprises in production!

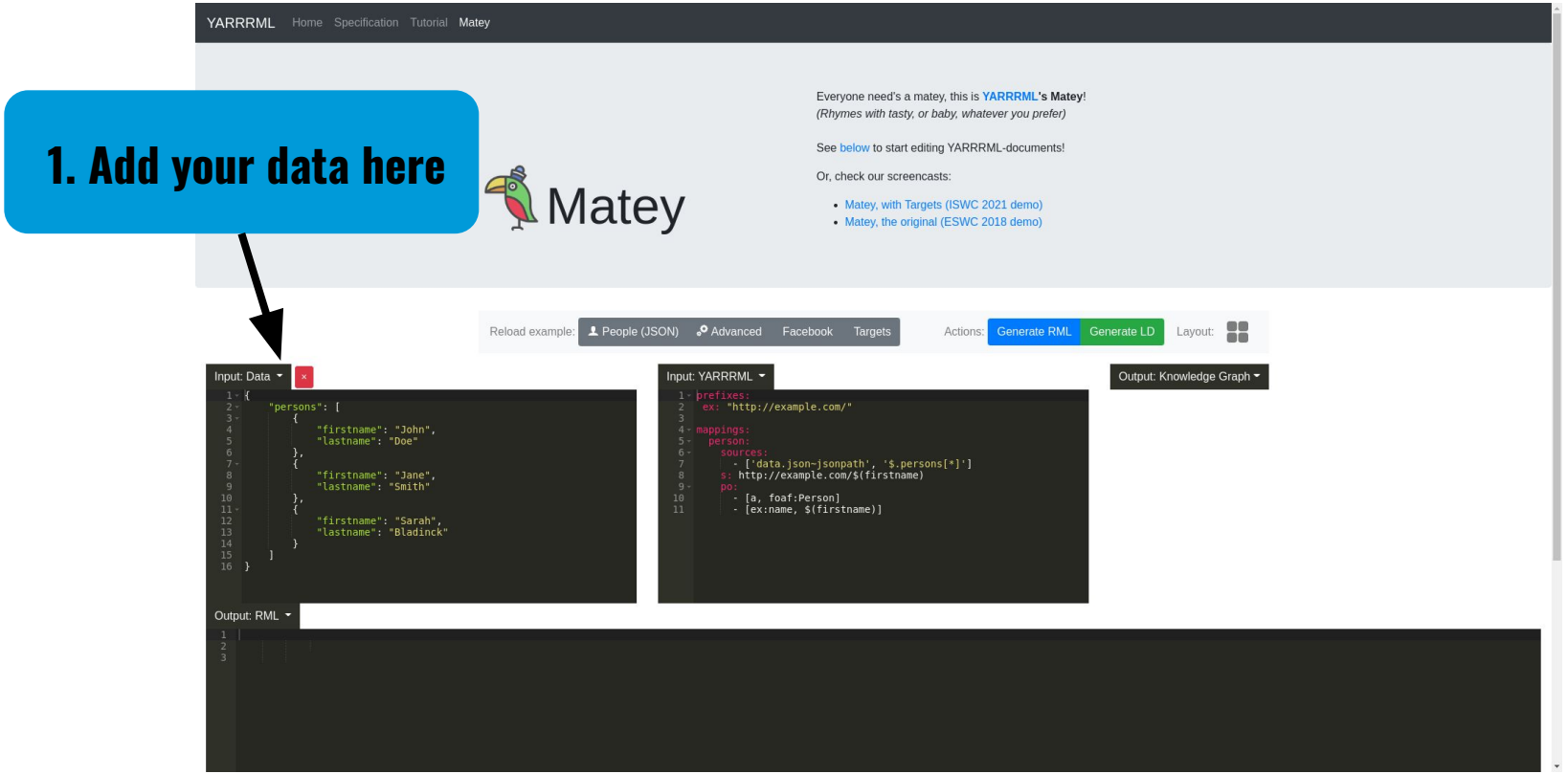
Quickly prototyping
Matey executes YARRRML rules on your data.
The generated Linked Data is directly shown in your Matey

https://github.com/rmlio/matey
https://rml.io/yarrrml/matey/

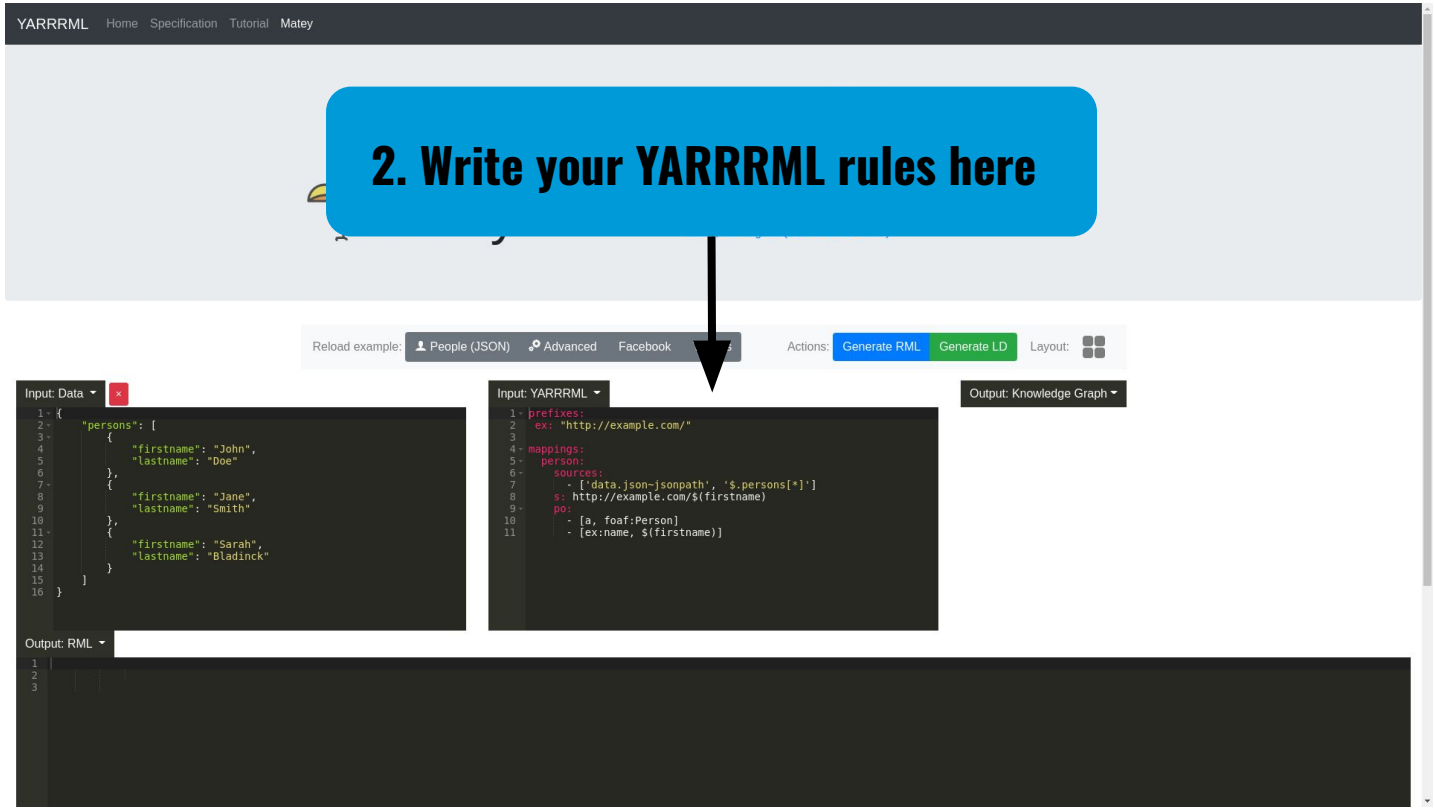
Matey UI: input data, YARRRML rules, output, and RML rules



Hands-on: demo

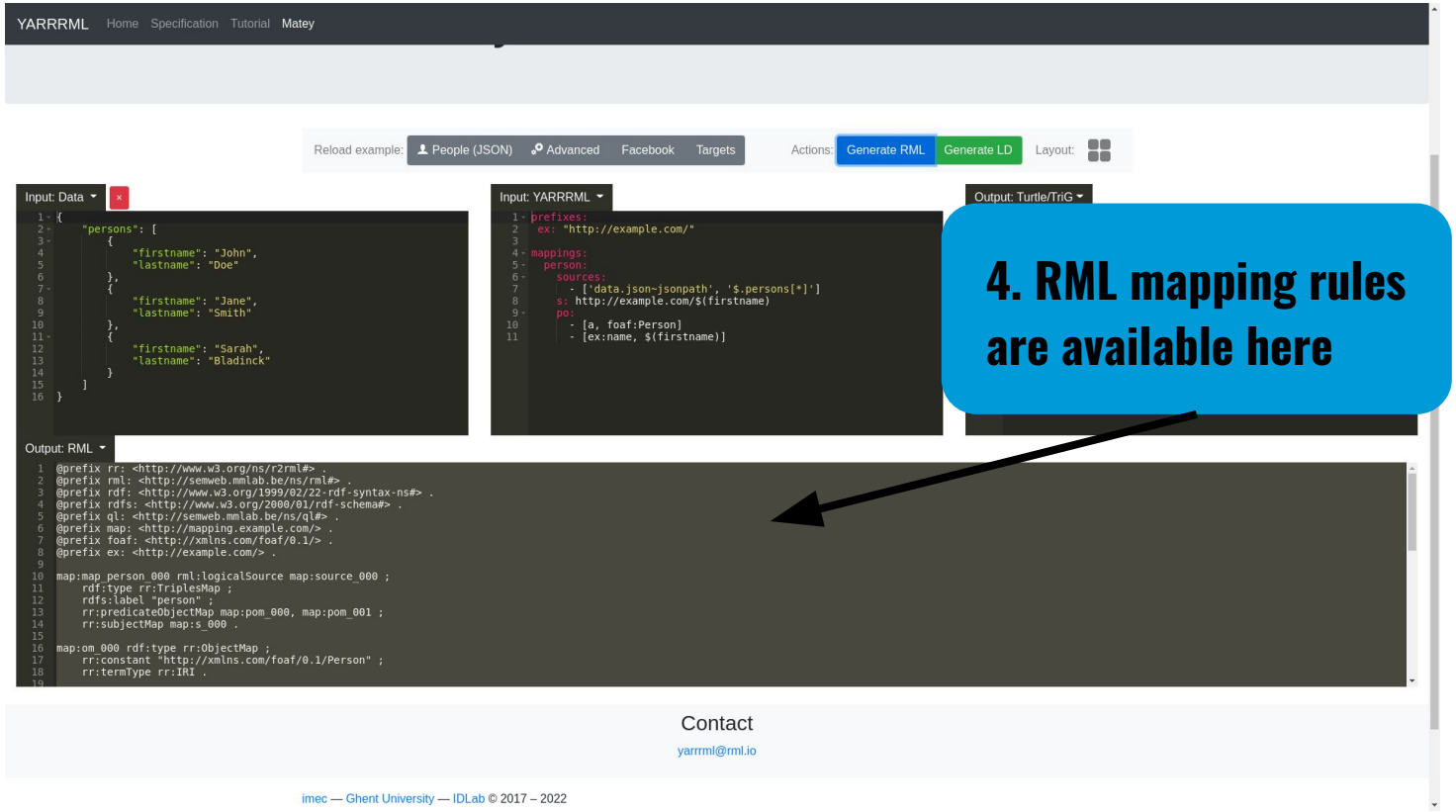


Hands-on: demo



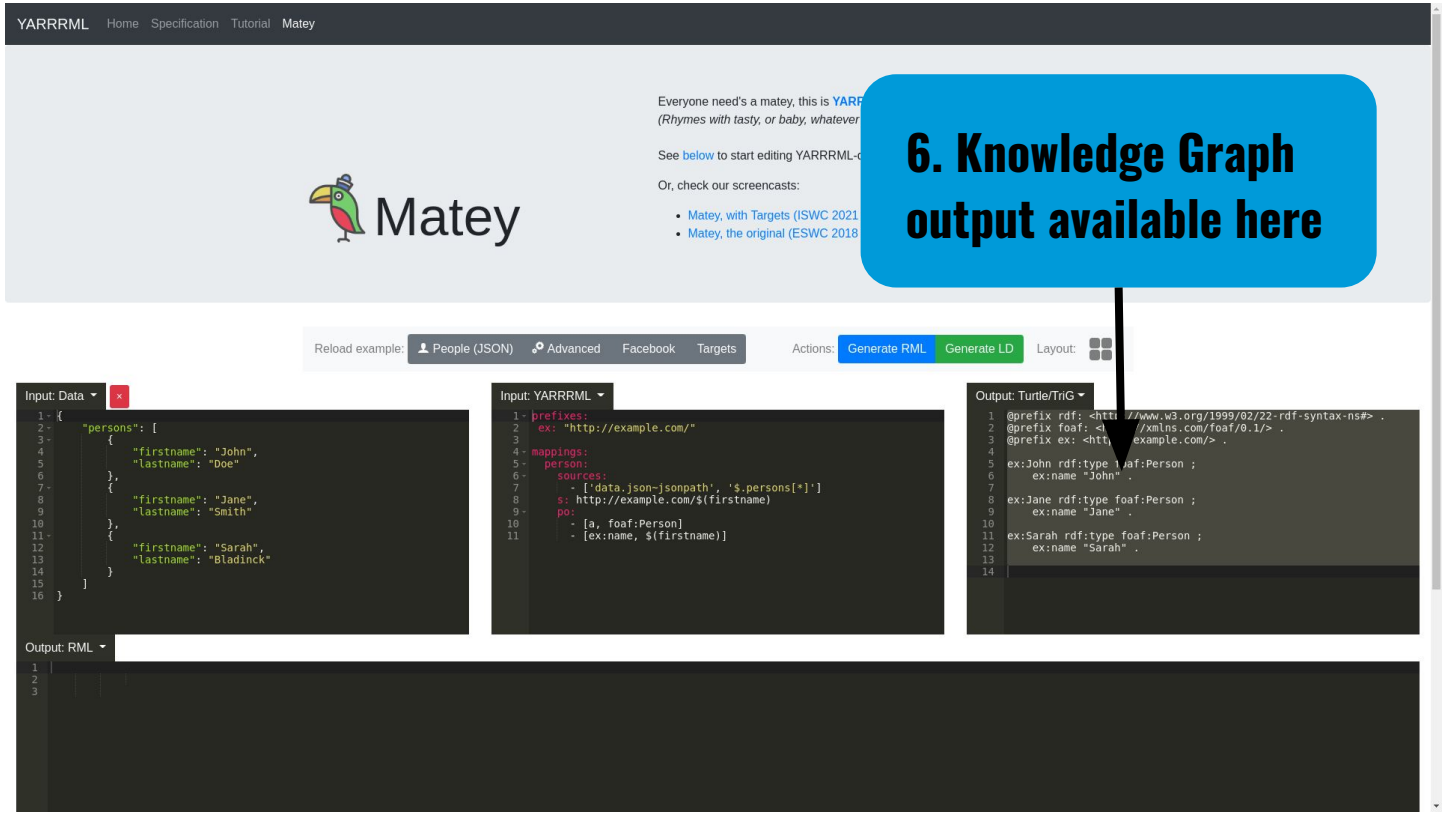
Hands-on: demo

Hands-on: demo



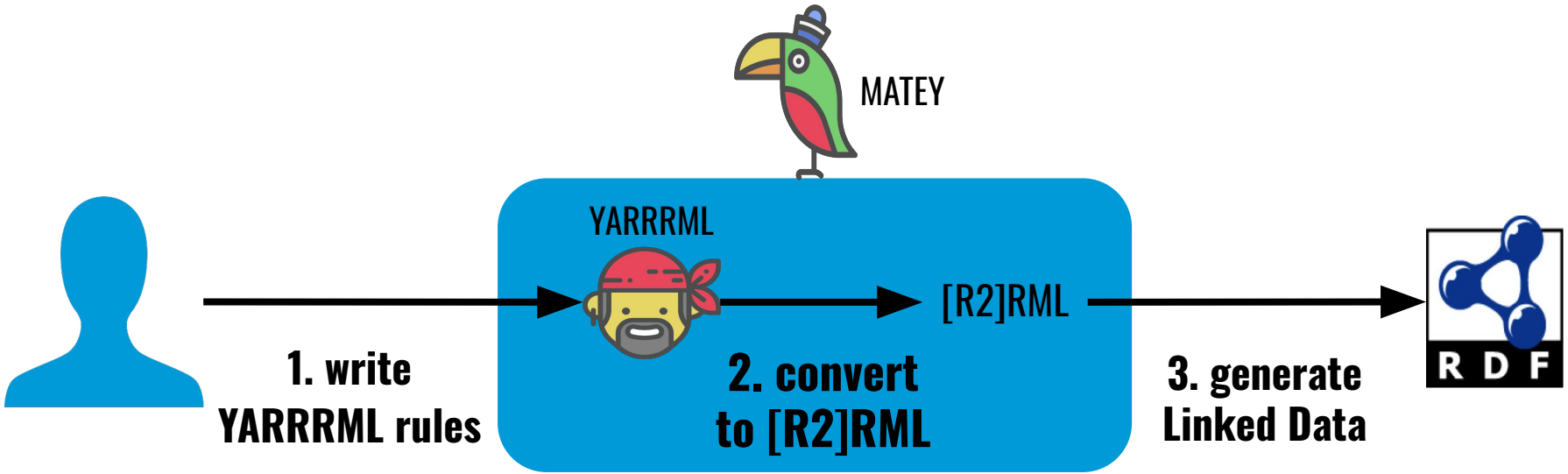
Hands-on: demo

Hands-on: demo



YARRRML & Matey hands-on!

https://rml.io/yarrrml/matey/



From Excel to RML

Markus Schröder

markus.schroeder@dfki.de

-

German
Research Center
for Artificial
Intelligence

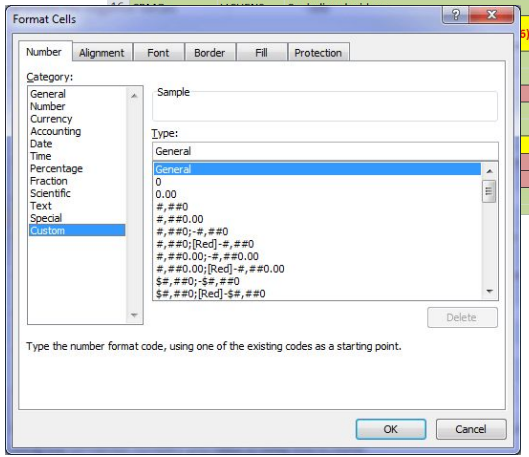
TECHNISCHE UNIVERSITÄT
KAISERSLAUTERN

Spreadsheets are not CSV files

- Well understood
- Easy and fast possibility to enter data
- Complex workbooks
- Multiple sheets
- Cells having rich meta data
  - Formats, colors, fonts, styles, borders, etc.
  - Arbitrarily arranged
  - Can lead to inconsistent and unstructured content

Found at data.gov:
[Hutten 2016_CRMO_lichen_moss_liverwort_.xlsx]

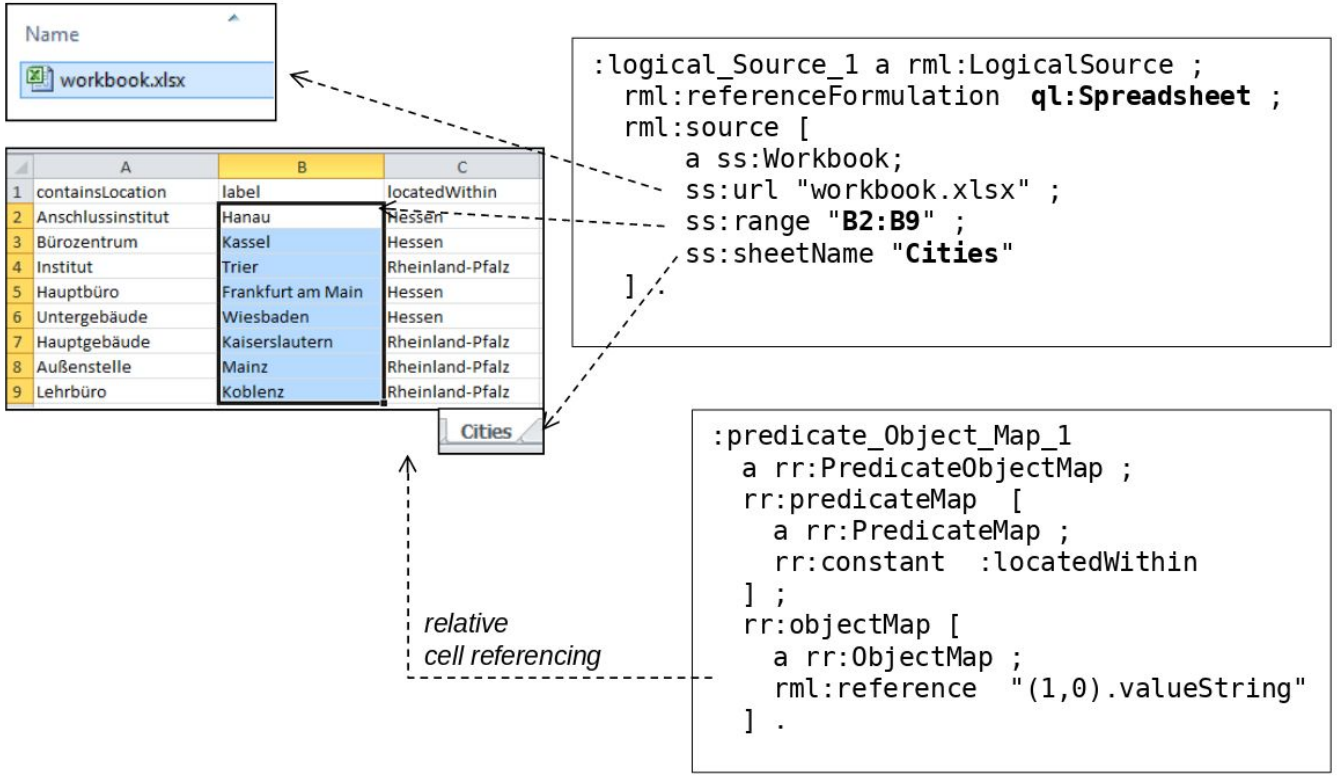
	A	B	C	D	E
1	Project_name	Class	Scientific_name	Scientific_name_Authority	Dfident
2	CRMO	LICHENS	Acarospora schleicheri	Acarospora schleicheri (Ach.) A. Massal.	-1
3	CRMO	LICHENS	Anaptychia elbursiana (Physconia thomsonii)	Anaptychia elbursiana (Szatala) Poelt	-1
4	CRMO	LICHENS	Arthonia glebosa	Arthonia glebosa Tuck.	0
5	CRMO	LICHENS	Caloplaca citrina	Caloplaca citrina (Hoffm.) Th. Fr.	-1
6			Caloplaca stillicidiorum (voucher 4229)	Caloplaca stillicidiorum (Vahl) Lynge (was not on this spreadsheet)	
7	CRMO	LICHENS	Candelaria concolor	Candelaria concolor (Dickson) Stein	-1
8			Candelaria reflexa (voucher 4299)	Candelaria reflexa (Nyl.) Lettau (was not on this spreadsheet)	?
9	CRMO	LICHENS	Cladonia chlorophaea	Cladonia chlorophaea (Flörke ex Sommerf.) Sprengel	-1
10	CRMO	LICHENS	Cladonia fimbriata	Cladonia fimbriata (L.) Fr.	-1
11	CRMO	LICHENS	Cladonia merochlorophaea	Cladonia merochlorophaea Asahina	-1
12	CRMO	LICHENS	Cladonia pocillum	Cladonia pocillum (Ach.) Grognot	-1
13	CRMO	LICHENS	Cladonia verruculosa	Cladonia verruculosa (Vainio) Ahti	-1
14			Collema sp. (voucher 4466)	Collema sp. F. H. Wigg. (was not on this spreadsheet)	
15	CRMO	LICHENS	Collema tenax	Collema tenax (Sw.) Ach.	-1
				Cyphelium lucidum (Th. Fr.) Th. Fr.	-1
				Dermatocarpon intestiniforme (Korber) Hasse (was not on this spreadsheet)	
				Dermatocarpon miniatum (L.) W. Mann	-1
				Dermatocarpon reticulatum H. Magn.	-1
				Dermatocarpon rivulorum (Arnold) Dalla Torre & Sarnth.	0
				Diploschistes diacapsis (Ach.) Lumbsch	-1
				Endocarpon pusillum Hedwig	-1
				Lecanora cenisia Ach. (was not on this spreadsheet)	
				Lecanora novomexicana H. Magn.	0
				Lecidea atrobrunnea (Ramond ex Lam. & DC.) Schaerer	0
				Lecidoma demissum (Rutstr.) Gotth. Schneider & Hertel	-1
				Leptogium californicum Tuck.	1



Benefits of native support in RML

- Eliminates extraneous conversion efforts: No pre-processing and transformation needed
- All aspects of a spreadsheet can be exploited
- Eases mapping rule communication
- For RML practitioners no extra language to learn

How it is used



- Logical source for spreadsheets
- Sheet and range selection

- Cell reference
- Access cell meta data

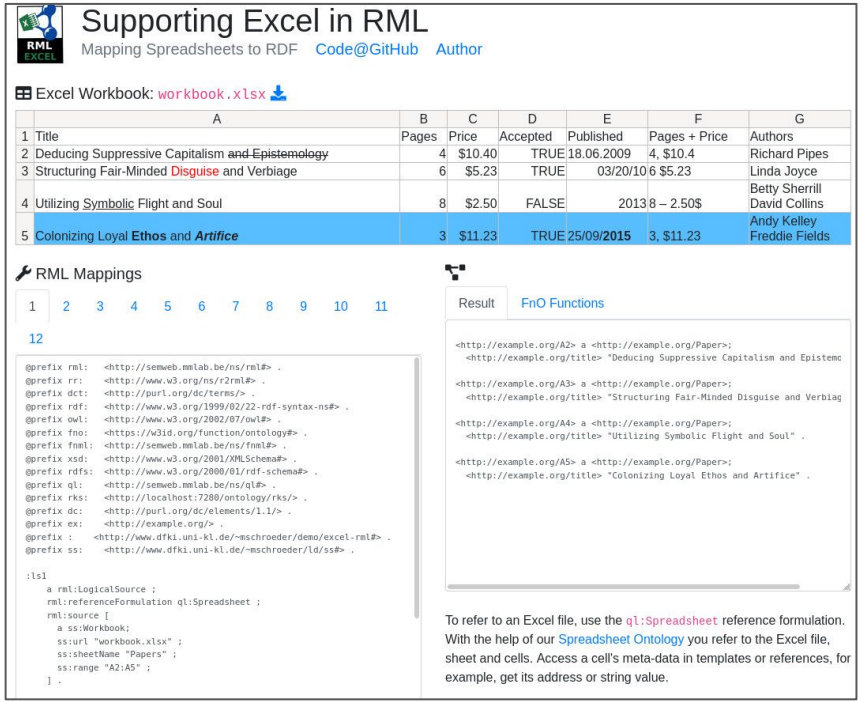
Accessible Meta Data

- Location
- Values
- Appearance

Variable Name	Description	Example
`address`	Location in a sheet	B3
`column`	Column index 0-based	1
`row`	Row index 0-based	2
`valueNumeric`	Floating point value	7.3
`valueInt`	Integer value	7
`valueBoolean`	Boolean value	true
`valueFormula`	Formula	`SUM(F5:F9)`
`valueError`	Possible error code, e.g. in case of division by zero	7
`valueString`	Plain text value	This is bold.
`valueRichText`	Formatted text in HTML syntax	This is bold.
`value`	String representation regardless of the data type	"7"
`json`	JSON representation to retrieve several meta data at once	`{"address": "E2", "cellType": "string", "valueNumeric": 0.0, "valueString": "18.06.2009", ...}`
`backgroundColor`	Background color in hexadecimal RGB value	`#FFFFFF`
`foregroundColor`	Foreground color in hexadecimal RGB value	`#000000`
`fontColor`	Font color in hexadecimal RGB value	`#FF0000`
`fontName`	Font name	Arial
`fontSize`	Font size	12

Excel in RML Demo

- Website
- Online Demo
- Paper
- Code



Spread2RML: Predicting RML Mappings on Messy Spreadsheets

These RML rules are automatically generated by the system:

	A	B	C	D	E
1	number	is recent	invalid from	hasEditor	valid from & planned valid from
2	2	Yes	11/13/2000	Alexander White	09/30/2020 *02/10/2020*
3	3	Yes	2016-07-30	Oliver Rodriguez	07/13/2016 *05/18/2016*
4	3	No	10/20/2012	Oliver Ramirez	2012-09-10 *05/12/2012*

Mixed data formats

C

```
<ex:C> rr:predicate gl:validFrom ;
  rr:objectMap [
    fnml:functionValue [
      rr:predicateObjectMap [
        rr:predicate  fno:executes ;
        rr:object     <java:parseDate>
      ] ;
      rr:predicateObjectMap [ (…)
        rr:objectMap [
          rml:reference "(2,0).json"
        ]
      ]
    ] ;
    rr:datatype xsd:date ;
    rr:termType rr:Literal   ] .
```

Entity mentions

D

```
<ex:D> rr:predicate gl:hasEditor ;
  rr:objectMap [
    fnml:functionValue [
      rr:predicateObjectMap [
        rr:predicate  fno:executes ;
        rr:object     <java:entityLinking>
      ] ; (…)
    ] ;
    rr:termType rr:IRI   ] .
```

Mixed style usages

E

```
<ex:E> rr:predicate gl:plannedValidFrom ;
  rr:objectMap [
    fnml:functionValue [
      rr:predicateObjectMap [
        rr:predicate  fno:executes ;
        rr:object     <java:getEntitiesByColor>
      ] ; (…)
      rr:predicateObjectMap [
        rr:predicate  (…) ;
        rr:objectMap [
          rr:constant "#ff0000"
        ]
      ] ; (…)
    ] ;
    rr:datatype xsd:date ;
    rr:termType rr:Literal   ] .
```

FnO functions to cope with messy spreadsheet data

- Literals can also be represented as character strings
  - `parseNumber`
  - `parseBoolean`
  - `parseDate` & `parseDateTime`

L
Limit of Quantification
1
0.84
0.766 ppb
0.864
N/A
0.4
6.7
N/A
N/A
N/A
.34 pg

E	F
Year Sampled 2006	Year sampled 2007
x	x
x	--
x	x
x	--

G
date read
4.16.15
4.16.15
4.16.15

- Named entities are mentioned in texts
  - `entityLinking`

Chemical class
Perfluorinated chemical
Plasticizer
Perfluorinated chemical
Perfluorinated chemical
Perfluorinated chemical
Alkylphenol

D	E
Taxonomic Applicability[1]	Adverse Outcome
fish, bird	early life stage mortality
fish, bird	early life stage mortality
fish, bird, amphibian	reproductive dysfunction

- Colored or typographical emphasized text
  - `getEntitiesByColor`
  - `getEntitiesByTag`
  - `getEntitiesByUnformatted`

C
Scientific_name
Acarospora schleicheri
Anaptychia elbursiana (Physconia thomsonii)
Arthonia glebosa
Caloplaca citrina
Caloplaca stillicidiorum (voucher 4229)
Candelaria concolor
Candelaria reflexa (voucher 4299)

Appeals Adjudicated (formerly Appeals Decided) **(#65)**
Number of appeals in BVA's pending inventory **(#561)**
Percent of appeals decided with at least one remanded issue **(#709)**
Hearings held **(#712)**
Number of issues decided (adjudicated) **(#778)**

How is it done: RML object map templates

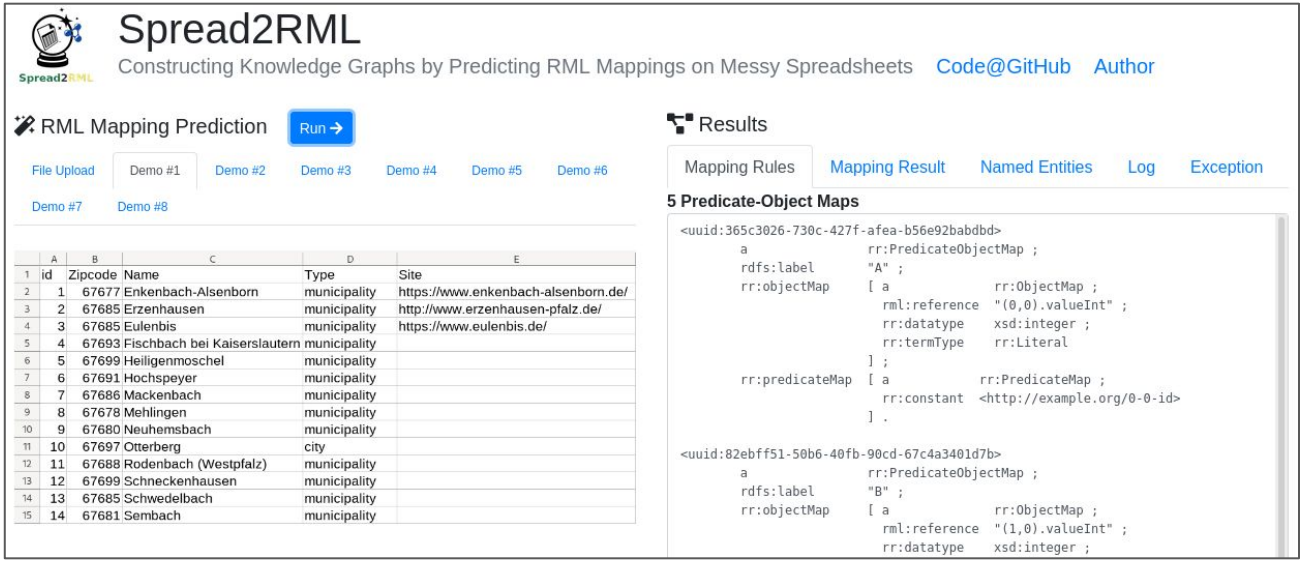
Object Map Template	Heuristic	Rank	Reference	Term Type	Datatype	FnO Function

Formatted Text	-	-	valueRichText	*child dependent*	*child dependent*	`getEntitiesByTag`
						`getEntitiesByColor`
						`getEntitiesByUnformatted`

Integer as String	$\lvert\{c \in S \mid int(c)\} \cup \{c \in N \mid dp(c) = 0\}\rvert \div \lvert C\rvert$	2	json	Literal	xsd:integer	`parseNumber`
Decimal as String (“.” decimal point)	$\lvert\{c \in S \mid dec(c, \text{“.”})\} \cup \{c \in N \mid dp(c) > 0\}\rvert \div \lvert C\rvert$	1	json	Literal	xsd:decimal	`parseNumber`
Decimal as String (“,” decimal point)	$\lvert\{c \in S \mid dec(c, \text{“,”})\} \cup \{c \in N \mid dp(c) > 0\}\rvert \div \lvert C\rvert$	1	json	Literal	xsd:decimal	`parseNumber`
Date as String	$\lvert\{c \in S \mid date(c)\} \cup N\rvert \div \lvert C\rvert$	2	json	Literal	xsd:date	`parseDate`
DateTime as String	$\lvert\{c \in S \mid datetime(c)\} \cup N\rvert \div \lvert C\rvert$	3	json	Literal	xsd:dateTime	`parseDateTime`
Integer List as String	$\lvert\{\hat{s} \in \hat{S} \mid int(\hat{s})\}\rvert \div \lvert\hat{S}\rvert$	0	json	Literal	xsd:integer	`parseNumber`
Boolean as String	if $\lvert\{str(c) \mid c \in S\}\rvert \leq 2$ then …	4	json	Literal	xsd:boolean	`parseBoolean`
String	$1 - dup(C)$	0	value	Literal	xsd:string	-
Single Entity	$dup(C)$	3	valueString	IRI	-	`entityLinking`
Multiple Entities	$dup(\hat{S} \cup N \cup B)$	4	valueString	IRI	-	`entityLinking`
Native Boolean	$\lvert B\rvert \div \lvert C\rvert$	0	valueBoolean	Literal	xsd:boolean	-
Native Integer	$\lvert\{c \in N \mid dp(c) = 0\}\rvert \div \lvert C\rvert$	3	valueInt	Literal	xsd:integer	-
Native Decimal	$\lvert\{c \in N \mid dp(c) > 0\}\rvert \div \lvert C\rvert$	4	valueNumeric	Literal	xsd:decimal	-
Numeric with Data Format	$\forall \delta \in D$ $\lvert\{c \in N \mid df(c) = \delta\}\rvert \div \lvert C\rvert$	5	json	Literal	xsd:date xsd:dateTime	`parseDate` `parseDateTime`

Spread2RML Demo

- Website
- Online Demo
- Paper
- Code



	A	B	C	D	E
1	id	Zipcode	Name	Type	Site
2	1	67677	Enkenbach-Alsenborn	municipality	https://www.enkenbach-alsenborn.de/
3	2	67685	Erzenhausen	municipality	http://www.erzenhausen-pfalz.de/
4	3	67685	Eulenbis	municipality	https://www.eulenbis.de/
5	4	67693	Fischbach bei Kaiserslautern	municipality	
6	5	67699	Heiligenmoschel	municipality	
7	6	67691	Hochspeyer	municipality	
8	7	67686	Mackenbach	municipality	
9	8	67678	Mehlingen	municipality	
10	9	67680	Neuhemsbach	municipality	
11	10	67697	Otterberg	city	
12	11	67688	Rodenbach (Westpfalz)	municipality	
13	12	67699	Schneckenhausen	municipality	
14	13	67685	Schwedelbach	municipality	
15	14	67681	Sembach	municipality	

Coffee Break

public-kg-construct@w3.org

kgc_workshop

MEL + TNNT

Metadata Extractor & Loader +
The NLP-NER Toolkit

Sergio José Rodríguez Méndez

Sergio.RodriguezMendez@anu.edu.au

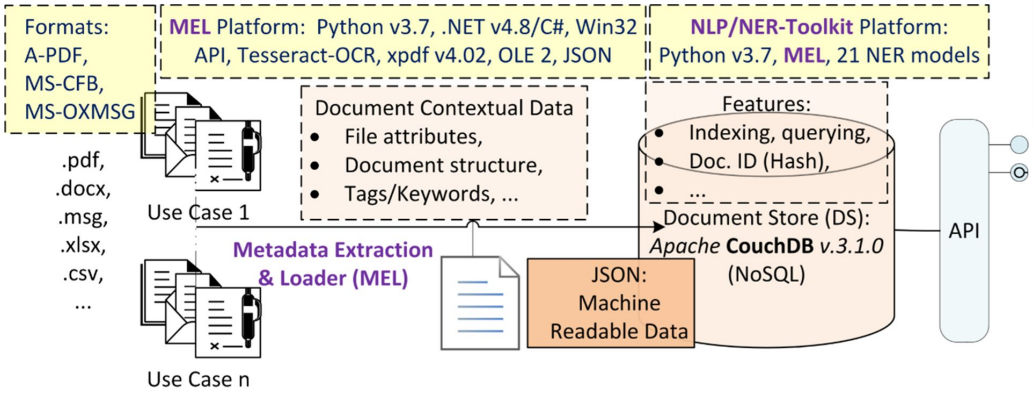
@sjrm_142857

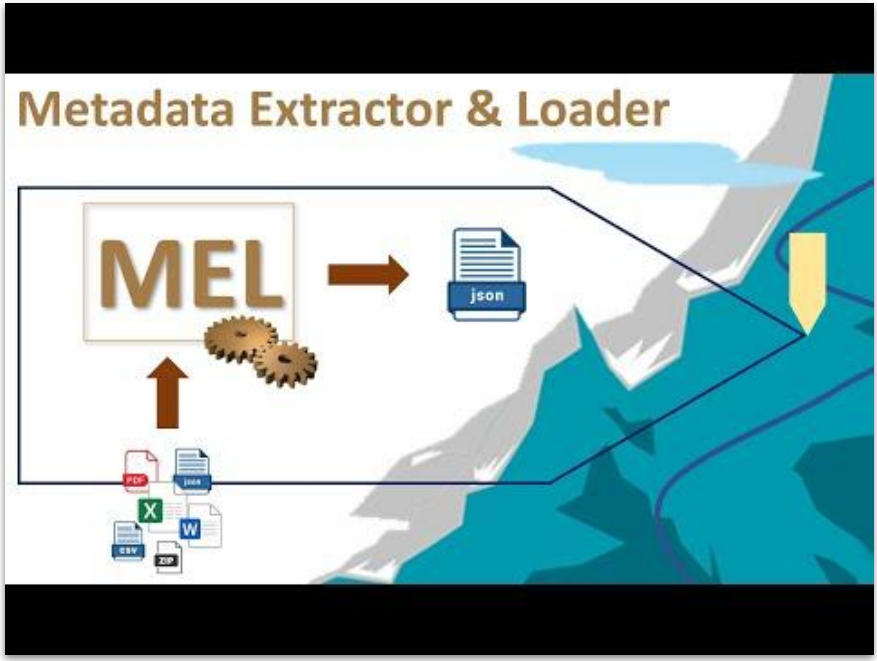
Australian National University

School of Computing

MEL: Metadata Extractor & Loader

- Automates the metadata and content extraction from over 20 different file formats.
- Basic text analysis: pattern matching and keyword frequency.
- Machine-readable output: JSON.
- Integrated with TNNT.





1-minute teaser video

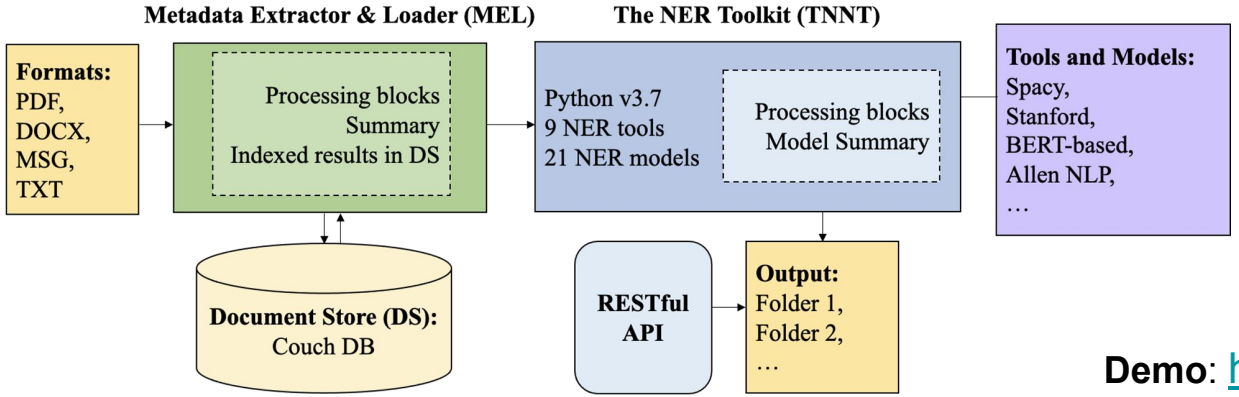
Demo: https://bit.ly/3kTOLJa

Repo : https://w3id.org/kgcp/MEL-TNNT

TNNT: The NLP-NER Toolkit

- Automates the Named-Entity Recognition (NER) task from MEL outputs.
- 21 integrated SoTA NER models, which can identify up to 18 categories.
- Different models processed sequentially based on the settings.
- Hybrid processing data flow either from/to a document store or via direct processing from the file system.
- Integrated summary of the NER results.

#	Tool	No. Integrated Models
1	NLTK [1]	1
2	Stanford NER [2]	3
3	Spacy	3
4	Stanza	1
5	Flair	5
6	Allen NLP	2
7	Polyglot	1
8	Deeppavlov	4
9	Bert-based [3]	1



Demo: https://bit.ly/3waal1A

Repo : https://w3id.org/kgcp/MEL-TNNT

Mapeathor

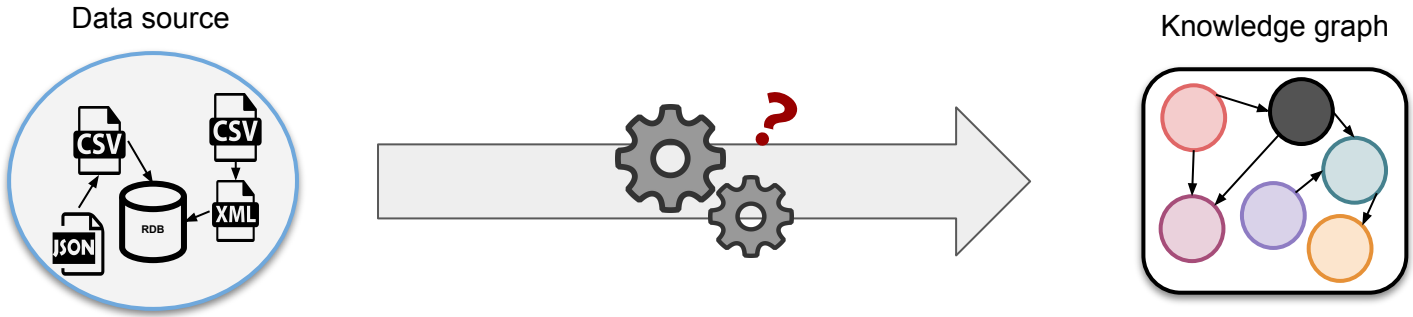
Ana Iglesias-Molina

ana.iglesiasm@upm.es

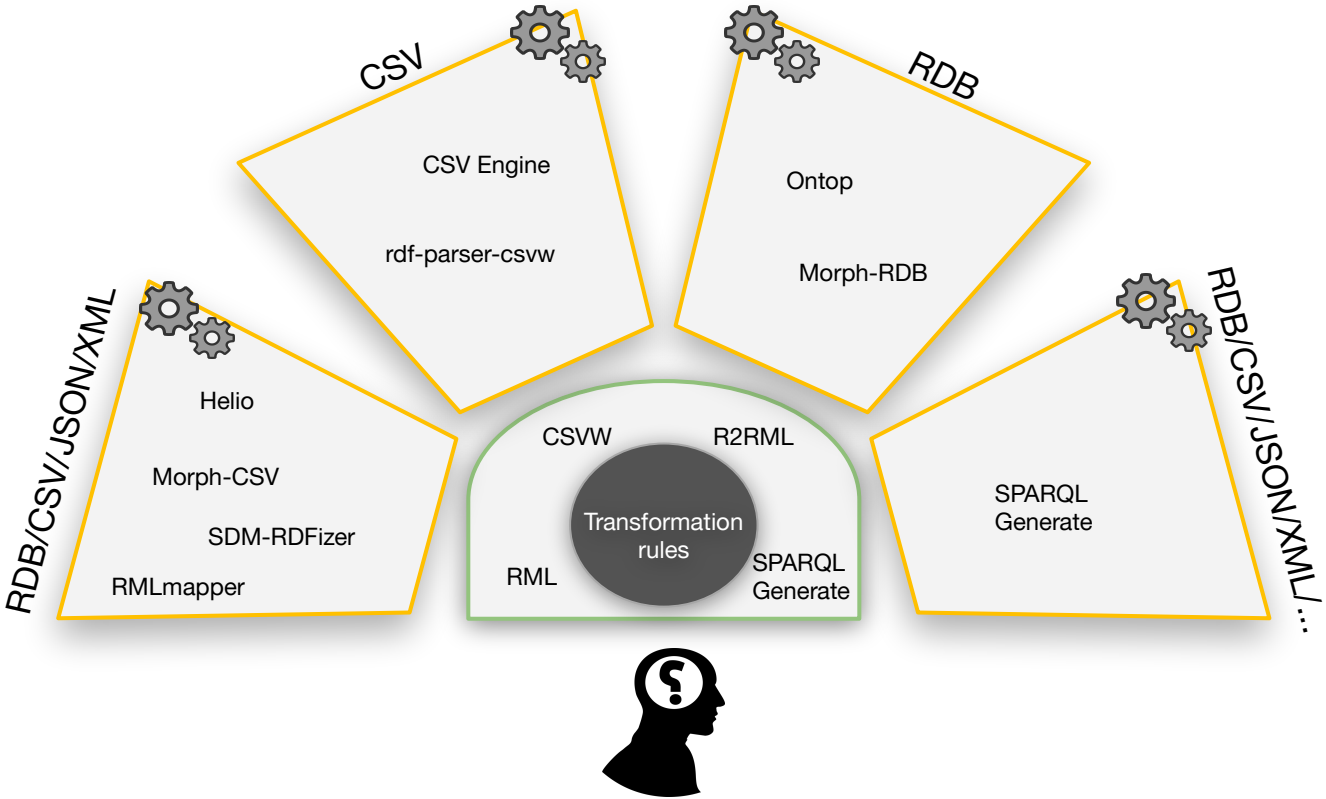
@_aieme

As we have just seen…

- Writing mapping rules in RDF-based languages is not an easy task
  - Verbosity
  - Implies knowing or learning RDF
- But not only that!
  - There are several mapping languages widely used
  - Each language is processed usually by different tools with different capabilities
  - In some cases, this leads to lack of interoperability



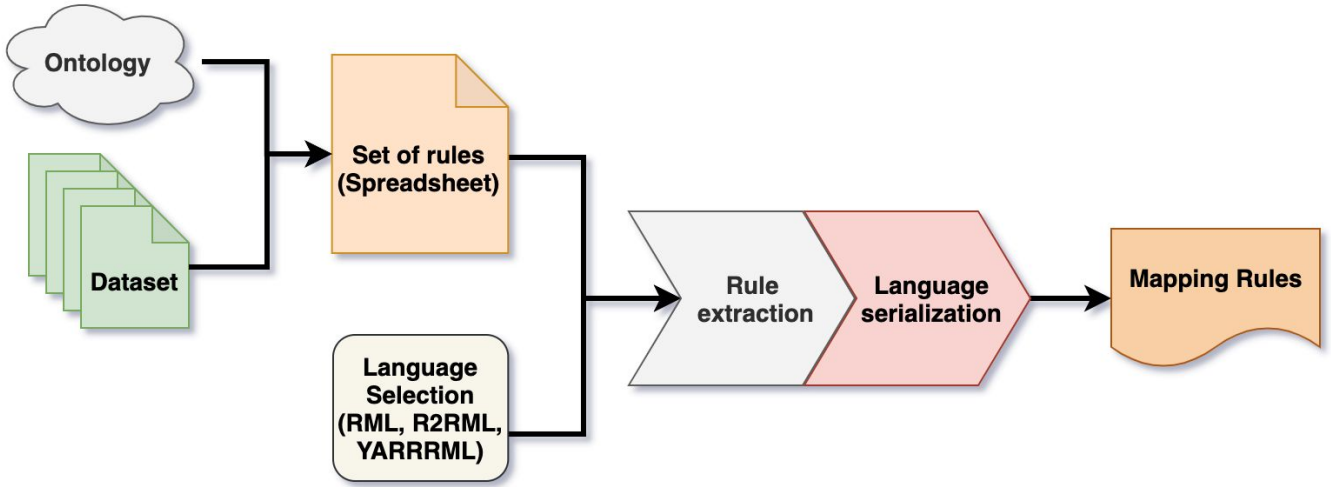
Mapping languages and tools excerpt



Mapeathor: Mapping rules as spreadsheets

- Represents mapping rules in spreadsheets
  - No need to learn the syntax of the languages
  - Improves rule visualization
  - Takes advantages of spreadsheets functionalities
  - Guides writing
- Translates to R2RML, RML and YARRRML
  - More languages in the future
- Increasing adoption
  - Used in several projects and by some companies

Mapeathor: Mapping rules as spreadsheets



Spreadsheet design

- The shared common characteristics of languages to transform data into RDF are expressed in **5 sheets**

(a) Prefix sheet

Prefix	URI
ex	http://ex.com/
rdfs	http://www.w3.org/2000/01/rdf-schema#

(b) Subject sheet

ID	Class	URI
PERSON	ex:Person	http://ex.com/Person/{ID}
SPORT	ex:Sport	http://ex.com/Sport/{ID}

(c) Source sheet

ID	Feature	Value
PERSON	source	data/people.csv
PERSON	format	CSV
SPORT	source	data/sports.csv
SPORT	format	CSV

(e) Function sheet

FunctionID	Feature	Value
<Fun1>	fno:executes	sql:concat
<Fun1>	ex:param_concat1	{ID}
<Fun1>	ex:param_concat2	<Fun2>
<Fun2>	fno:executes	sql:upper
<Fun2>	ex:param_upper	{sport}

(d) Predicate_Object sheet

ID	Predicate	Object	DataType	ReferenceID	InnerRef	OurterRef
PERSON	ex:name	{name}	string			
PERSON	ex:birthdate	{birthdate}	date			
PERSON	ex:sport			SPORT	{SportID}	{ID}
SPORT	ex:name	{sport}	string			
SPORT	ex:code	{ID}	integer			
SPORT	ex:comment	<Fun1>				

Spreadsheet design: Prefixes

Prefix	URI
rr	http://www.w3.org/ns/r2rml#
ex	http://example.com/
rml	http://semweb.mmlab.be/ns/rml#
fno	https://w3id.org/function/ontology#

```
@prefix rr: <http://www.w3.org/ns/r2rml#> .
@prefix ex: <http://example.com/> .
@prefix rml: <http://semweb.mmlab.be/ns/rml#> .
@prefix fno: <http://w3id.org/function/ontology#> .

<SPORT>
  rml:logicalSource [
    rml:source "data/sports.csv" ;
    rml:referenceFormulation ql:CSV ;
  ];
  rr:subjectMap [
    rr:class ex:Sport;
    rr:template "http://ex.com/Sport/{ID}";
  ];
  rr:predicateObjectMap [
    rr:predicateMap    [ rr:constant ex:name ];
    rr:objectMap    [ rml:reference "Sport" ];
  ];
```

```
@prefix rr: <http://www.w3.org/ns/r2rml#> .
@prefix ex: <http://example.com/> .
@prefix rml: <http://semweb.mmlab.be/ns/rml#> .
@prefix fno: <http://w3id.org/function/ontology#> .

<PERSON>
  rml:logicalSource [
    rml:source "data/people.csv" ;
    rml:referenceFormulation ql:CSV ;
  ];
  rr:subjectMap [
    rr:class ex:Person;
    rr:template "http://ex.com/Person/{ID}";
  ];
  rr:predicateObjectMap [
    rr:predicateMap    [ rr:constant ex:name ];
    rr:objectMap    [ rml:reference "Name" ];
  ];
  rr:predicateObjectMap [
    rr:predicateMap    [ rr:constant ex:birthdate ];
    rr:objectMap    <DATE_FUN>;
  ];
  rr:predicateObjectMap [
    rr:predicateMap    [ rr:constant ex:sport ];
    rr:objectMap    [ rr:parentTriplesMap <SPORT>;
      rr:joinCondition [ rr:child "SportID"; rr:parent "ID"; ];
    ];
  ]; .
```

Spreadsheet design: Source

ID	Feature	Value
PERSON	source	data/people.csv
PERSON	format	CSV
SPORT	source	data/sports.csv
SPORT	format	CSV

```
@prefix rr: <http://www.w3.org/ns/r2rml#> .
@prefix ex: <http://example.com/> .
@prefix rml: <http://semweb.mmlab.be/ns/rml#> .
@prefix fno: <http://w3id.org/function/ontology#> .

<SPORT>
  rml:logicalSource [
    rml:source "data/sports.csv" ;
    rml:referenceFormulation ql:CSV ;
  ];
  rr:subjectMap [
    rr:class ex:Sport;
    rr:template "http://ex.com/Sport/{ID}";
  ];
  rr:predicateObjectMap [
    rr:predicateMap    [ rr:constant ex:name ];
    rr:objectMap    [ rml:reference "Sport" ];
  ];
```

```
@prefix rr: <http://www.w3.org/ns/r2rml#> .
@prefix ex: <http://example.com/> .
@prefix rml: <http://semweb.mmlab.be/ns/rml#> .
@prefix fno: <http://w3id.org/function/ontology#> .

<PERSON>
  rml:logicalSource [
    rml:source "data/people.csv" ;
    rml:referenceFormulation ql:CSV ;
  ];
  rr:subjectMap [
    rr:class ex:Person;
    rr:template "http://ex.com/Person/{ID}";
  ];
  rr:predicateObjectMap [
    rr:predicateMap    [ rr:constant ex:name ];
    rr:objectMap    [ rml:reference "Name" ];
  ];
  rr:predicateObjectMap [
    rr:predicateMap    [ rr:constant ex:birthdate ];
    rr:objectMap    <DATE_FUN>;
  ];
  rr:predicateObjectMap [
    rr:predicateMap    [ rr:constant ex:sport ];
    rr:objectMap    [ rr:parentTriplesMap <SPORT>;
      rr:joinCondition [ rr:child "SportID"; rr:parent "ID"; ];
    ];
  ]; .
```

people.csv

sports.csv

Spreadsheet design: Subject

ID	Class	URI
PERSON	ex:Person	http://ex.com/Person/{ID}
SPORT	ex:Sport	http://ex.com/Sport/{ID}

```
@prefix rr: <http://www.w3.org/ns/r2rml#> .
@prefix ex: <http://example.com/> .
@prefix rml: <http://semweb.mmlab.be/ns/rml#> .
@prefix fno: <http://w3id.org/function/ontology#> .

<SPORT>
  rml:logicalSource [
     rml:source "data/sports.csv" ;
     rml:referenceFormulation ql:CSV ;
  ];
  rr:subjectMap [
     rr:class ex:Sport;
     rr:template "http://ex.com/Sport/{ID}";
  ];
  rr:predicateObjectMap [
     rr:predicateMap    [ rr:constant ex:name ];
     rr:objectMap    [ rml:reference "Sport" ];
  ];
```

```
@prefix rr: <http://www.w3.org/ns/r2rml#> .
@prefix ex: <http://example.com/> .
@prefix rml: <http://semweb.mmlab.be/ns/rml#> .
@prefix fno: <http://w3id.org/function/ontology#> .

<PERSON>
  rml:logicalSource [
     rml:source "data/people.csv" ;
     rml:referenceFormulation ql:CSV ;
  ];
  rr:subjectMap [
     rr:class ex:Person;
     rr:template "http://ex.com/Person/{ID}";
  ];
  rr:predicateObjectMap [
     rr:predicateMap    [ rr:constant ex:name ];
     rr:objectMap    [ rml:reference "Name" ];
  ];
  rr:predicateObjectMap [
     rr:predicateMap    [ rr:constant ex:birthdate ];
     rr:objectMap    <DATE_FUN>;
  ];
  rr:predicateObjectMap [
     rr:predicateMap    [ rr:constant ex:sport ];
     rr:objectMap    [ rr:parentTriplesMap <SPORT>;
       rr:joinCondition [ rr:child "SportID"; rr:parent "ID"; ];
     ];
  ]; .
```

<Person/1>
ex:Person

<Person/2>
ex:Person

<Sport/2>
ex:Sport

<Sport/1>
ex:Sport

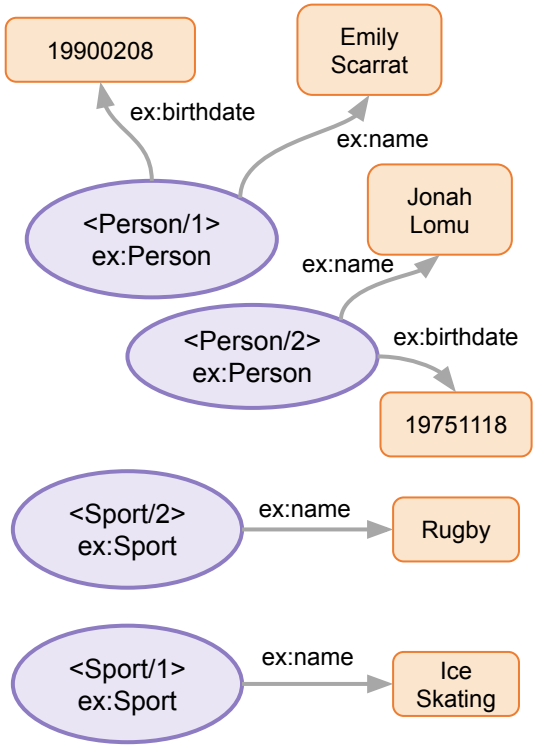
Spreadsheet design: Predicate Object

ID	Predicate	Object	DataType	ReferenceID	InnerRef	OuterRef
PERSON	ex:name	{Name}	string			
PERSON	ex:birthdate	<DATE_FUN>				
PERSON	ex:sport					
SPORT	ex:name	{Sport}	string			

```
@prefix rr: <http://www.w3.org/ns/r2rml#> .
@prefix ex: <http://example.com/> .
@prefix rml: <http://semweb.mmlab.be/ns/rml#> .
@prefix fno: <http://w3id.org/function/ontology#> .

<SPORT>
  rml:logicalSource [
     rml:source "data/sports.csv" ;
     rml:referenceFormulation ql:CSV ;
  ];
  rr:subjectMap [
     rr:class ex:Sport;
     rr:template "http://ex.com/Sport/{ID}";
  ];
  rr:predicateObjectMap [
     rr:predicateMap     [ rr:constant ex:name ];
     rr:objectMap     [ rml:reference "Sport" ];
  ];
```

```
    rml:referenceFormulation ql:CSV ;
  ];
  rr:subjectMap [
     rr:class ex:Person;
     rr:template "http://ex.com/Person/{ID}";
  ];
  rr:predicateObjectMap [
     rr:predicateMap     [ rr:constant ex:name ];
     rr:objectMap     [ rml:reference "Name" ];
  ];
  rr:predicateObjectMap [
     rr:predicateMap     [ rr:constant ex:birthdate ];
     rr:objectMap     <DATE_FUN>;
  ];
  rr:predicateObjectMap [
     rr:predicateMap     [ rr:constant ex:sport ];
     rr:objectMap     [ rr:parentTriplesMap <SPORT>;
       rr:joinCondition [ rr:child "SportID"; rr:parent "ID"; ];
     ];
  ]; .
```



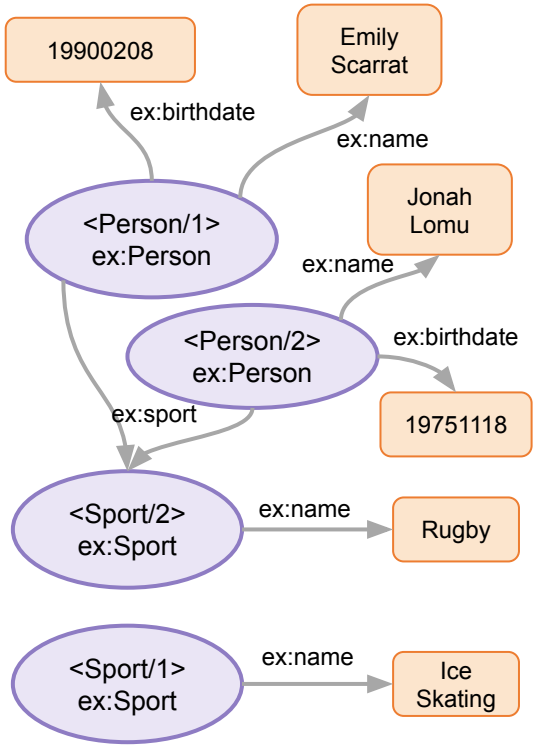
Spreadsheet design: Predicate Object

ID	Predicate	Object	DataType	ReferenceID	InnerRef	OuterRef
PERSON	ex:name	{Name}	string			
PERSON	ex:birthdate	<DATE_FUN>				
PERSON	ex:sport			SPORT	{SportID}	{ID}
SPORT	ex:name	{Sport}	string			

```
@prefix rr: <http://www.w3.org/ns/r2rml#> .
@prefix ex: <http://example.com/> .
@prefix rml: <http://semweb.mmlab.be/ns/rml#> .
@prefix fno: <http://w3id.org/function/ontology#> .

<SPORT>
  rml:logicalSource [
     rml:source "data/sports.csv" ;
     rml:referenceFormulation ql:CSV ;
  ];
  rr:subjectMap [
     rr:class ex:Sport;
     rr:template "http://ex.com/Sport/{ID}";
  ];
  rr:predicateObjectMap [
     rr:predicateMap     [ rr:constant ex:name ];
     rr:objectMap     [ rml:reference "Sport" ];
  ];
```

```
> .
> .
     rml:referenceFormulation ql:CSV ;
  ];
  rr:subjectMap [
     rr:class ex:Person;
     rr:template "http://ex.com/Person/{ID}";
  ];
  rr:predicateObjectMap [
     rr:predicateMap     [ rr:constant ex:name ];
     rr:objectMap     [ rml:reference "Name" ];
  ];
  rr:predicateObjectMap [
     rr:predicateMap     [ rr:constant ex:birthdate ];
     rr:objectMap     <DATE_FUN>;
  ];
  rr:predicateObjectMap [
     rr:predicateMap     [ rr:constant ex:sport ];
     rr:objectMap     [ rr:parentTriplesMap <SPORT>;
       rr:joinCondition [ rr:child "SportID"; rr:parent "ID"; ];
     ];
  ]; .
```

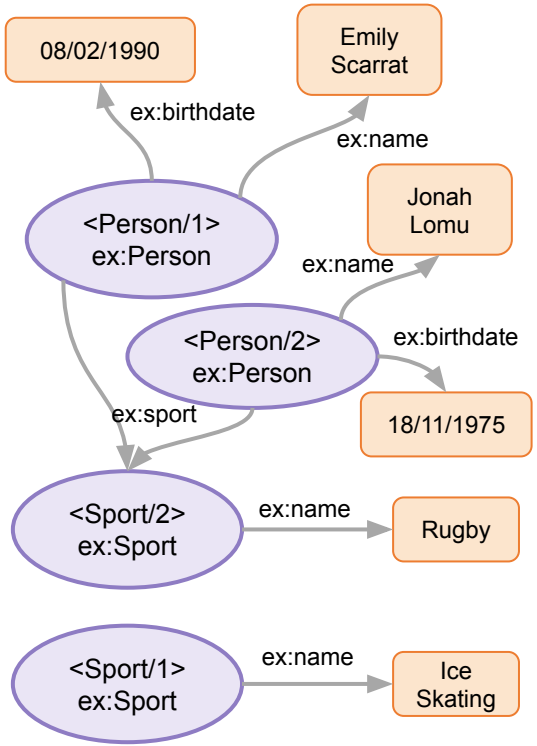


Spreadsheet design: Functions

FunctionID	Feature	Value
<DATE_FUN>	fno:executes	grel:toDate
<DATE_FUN>	ex:valueParam1	{Birthdate}
<DATE_FUN>	ex:valueParam1	“yyyyMMdd”
<DATE_FUN>	ex:valueParam2	“d/M/y”

```
@prefix rr: <http://www.w3.org/ns/r2rml#> .
@prefix ex: <http://example.com/> .
@prefix rml: <http://semweb.mmlab.be/ns/rml#> .
@prefix fno: <http://w3id.org/function/ontology#> .

<DATE_FUN>
 rml:logicalSource [
    rml:source "data/people.csv" ;
    rml:referenceFormulation ql:CSV ;
 ];
 rr:predicateObjectMap [
    rr:predicateMap    fno:executes;
    rr:objectMap    [ rml:constant grel:toDate ];
 ];
 rr:predicateObjectMap [
    rr:predicateMap    grel:valueParam1;
    rr:objectMap    [ rml:reference “Birthdate” ];
 ];
 rr:predicateObjectMap [
    rr:predicateMap    grel:valueParam2;
    rr:objectMap    [ rml:constant “yyyyMMdd” ];
 ];
 rr:predicateObjectMap [
    rr:predicateMap    grel:valueParam3;
    rr:objectMap    [ rml:constant “d/M/y” ];
 ];
```



How to run Mapeathor

https://github.com/oeg-upm/mapeathor

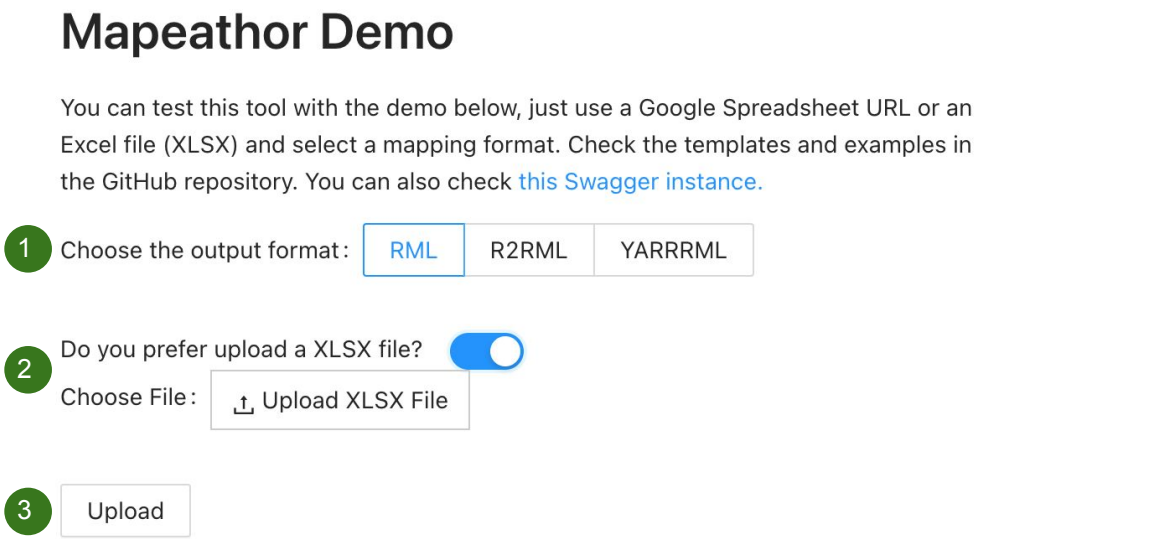
https://pypi.org/project/mapeathor/

Try it online → https://morph.oeg.fi.upm.es/demo/mapeathor

Hands-on time!

1. Create the mapping. Resources in: https://github.com/kg-construct/eswc-dkg-tutorial-2022/tree/main/material/mapeathor

2. Run it! →



https://morph.oeg.fi.upm.es/demo/mapeathor

How did you find this tool?

- We are running a survey assessing the easiness of use of Mapeathor and the spreadsheet template
- By filling the following form, you can help us improve it:
  - https://forms.gle/N2RajsjCinzsoSK26

Thanks!

Dragoman

Samaneh Jozashoori

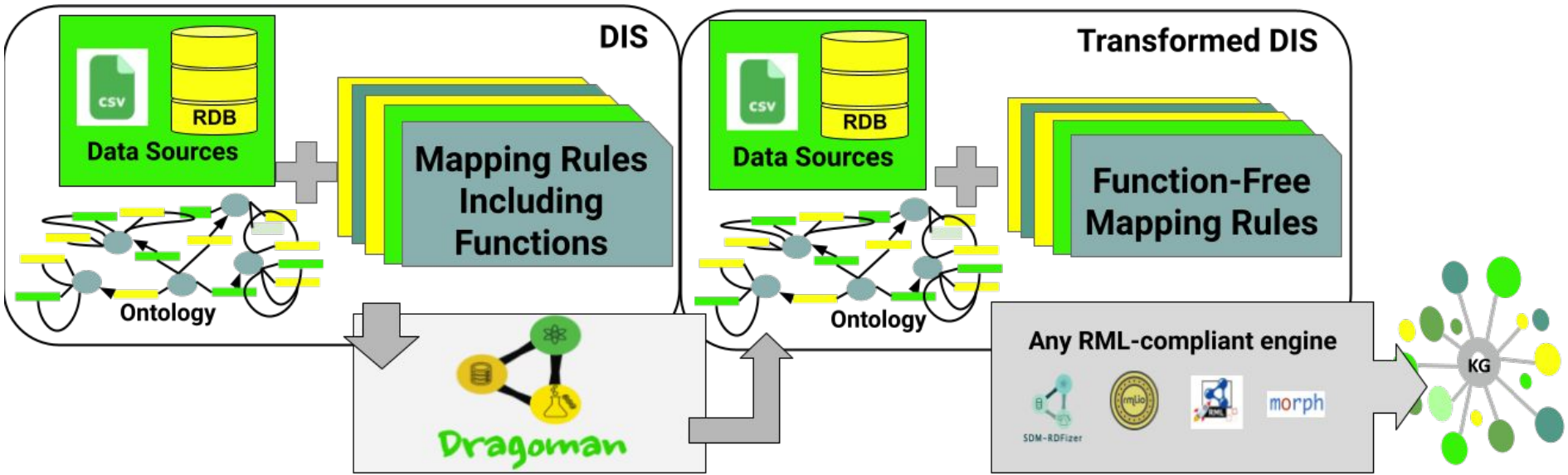
Authors

Samaneh Jozashoori, Enrique Iglesias, Maria-Esther Vidal

samaneh.jozashoori@gmail.com
samaneh.jozashoori@tib.eu

@samstwitting

Knowledge Graph Creation Pipeline Incl. Dragoman



R2RML Triples Maps- Abstract Description

Given **DIS=<O**,**S**,**M>**, mapping rules in M are defined as safe horn clauses

$$body(\overline{X}) : -head(\overline{Y})$$

$body(\overline{X})$: conjunctive query over the alphabet of the data sources S with variables in $\overline{X}$

$head(\overline{Y})$ conjunction of predicates representing classes and properties in O with variables in $\overline{Y}$

$\overline{Y}$ subset of $\overline{X}$

R2RML Triples Maps- Abstract Description

- Concept Mapping Assertions
- Role Mapping Assertions
  - Single-Source Role Mapping Assertions
  - Multi-Sources Role Mapping Assertions
- Attribute Mapping Assertions

-> Read more: “**Scaling Up Knowledge Graph Creation to Large and Heterogeneous Data Sources**” https://arxiv.org/pdf/2201.09694.pdf
-> Read more: “**MapSDI: A Scaled-Up Semantic Data Integration Framework for Knowledge Graph Creation**” https://link.springer.com/chapter/10.1007/978-3-030-33246-4_4

Optimization and Transformation: Source-based Projection

```
<TriplesMap1>
  rml:logicalSource [ rml:source "S1.csv";
    rml:referenceFormulation ql:CSV ];
  rr:subjectMap
    [rr:template "http://ex.com/{Att2}"];
  rr:predicateObjectMap [ rr:predicate ex:p1;
    rr:objectMap [ rml:reference "Att5" ]].
```

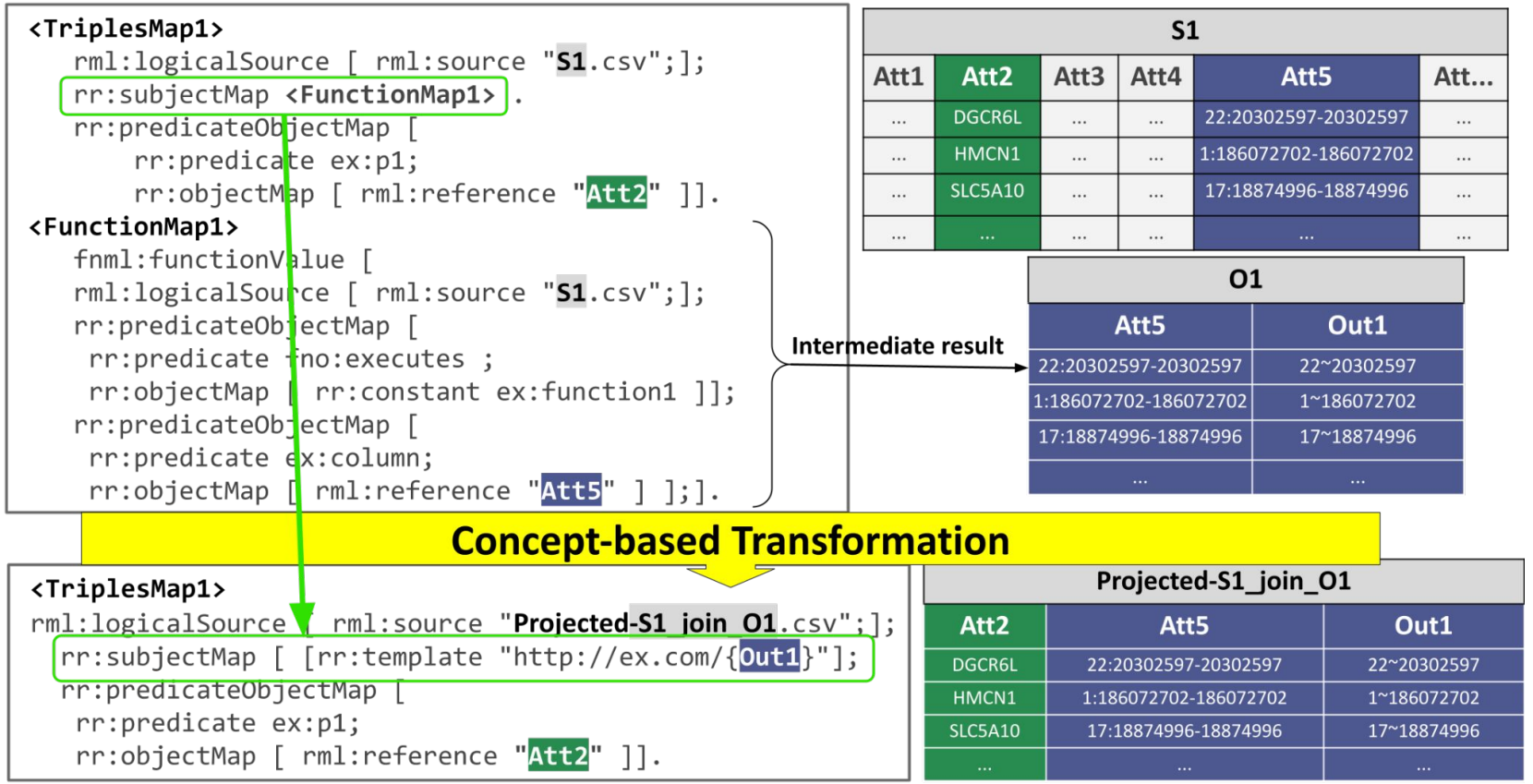
S1					
Att1	Att2	Att3	Att4	Att5	...
...	DGCR6L	...	...	22:20302597-20302597	...
...	HMCN1	...	...	1:186072702-186072702	...
...	SLC5A10	...	...	17:18874996-18874996	...
...	...	...	...	...	...

Source-based Projection

```
<TriplesMap1>
  rml:logicalSource [ rml:source "Projected-S1.csv";
    rml:referenceFormulation ql:CSV ];
  rr:subjectMap [rr:template "http://ex.com/{Att2}"];.
  rr:predicateObjectMap [rr:predicate ex:p1;
     rr:objectMap [ rml:reference "Att5" ]].
```

Projected-S1	
Att2	Att5
DGCR6L	22:20302597-20302597
HMCN1	1:186072702-186072702
SLC5A10	17:18874996-18874996
...	...

Optimization and Transformation: Concept-based Transformation



Optimization and Transformation: Role-based Transformation

```
<TriplesMap1>
  rml:logicalSource [ rml:source "S1.csv";
       rml:referenceFormulation ql:CSV ];
  rr:subjectMap [rr:template "http://ex.com/{Att2}"];
  rr:predicateObjectMap [
     rr:predicate ex:p1;
     rr:objectMap <FunctionMap1> ];
<FunctionMap1>
  fnml:functionValue [
   rml:logicalSource [ rml:source "S1.csv";];
   rr:predicateObjectMap [
     rr:predicate fno:executes ;
     rr:objectMap [ rr:constant ex:function1 ]];
   rr:predicateObjectMap [
     rr:predicate ex:column;
     rr:objectMap [ rml:reference "Att5" ] ];];
  rr:termType rr:IRI.
```

Out1

Role-based Transformation

```
<TriplesMap1>
  rml:logicalSource [ rml:source "Projected-S1.csv";
     rml:referenceFormulation ql:CSV ];
  rr:subjectMap [rr:template "http://ex.com/{Att2}"];
  rr:predicateObjectMap [
     rr:predicate ex:p1;
     rr:objectMap [
      rr:parentTriplesMap <TriplesMap2> ;
      rr:joinCondition [ rr:child "Att5";
                         rr:parent "Att5"];];].
<TriplesMap2>
     rml:logicalSource [ rml:source "O1.csv";
         rml:referenceFormulation ql:CSV ];
     rr:subjectMap [rr:template
"http://ex.com/{Out1}"];
```

S1

Att1	Att2	Att3	Att4	Att5	Att...
...	DGCR6L	...	...	22:20302597-20302597	...
...	HMCN1	...	...	1:186072702-186072702	...
...	SLC5A10	...	...	17:18874996-18874996	...
...	...	...	...	...	...

Projected-S1

Att2	Att5
DGCR6L	22:20302597-20302597
HMCN1	1:186072702-186072702
SLC5A10	17:18874996-18874996
...	...

O1

Att5	Out1
22:20302597-20302597	22~20302597
1:186072702-186072702	1~186072702
17:18874996-18874996	17~18874996
...	...

Optimization and Transformation: Attribute-based Transformation

```
<TriplesMap1>
  rml:logicalSource [ rml:source "S1.csv";
       rml:referenceFormulation ql:CSV ];
  rr:subjectMap [rr:template "http://ex.com/{Att2}"];
  rr:predicateObjectMap [
     rr:predicate ex:p1;
     rr:objectMap <FunctionMap1> ];
<FunctionMap1>
  fnml:functionValue [
   rml:logicalSource [ rml:source "S1.csv";];
   rr:predicateObjectMap [
     rr:predicate fno:executes ;
     rr:objectMap [ rr:constant ex:function1 ]];
   rr:predicateObjectMap [
     rr:predicate ex:column;
     rr:objectMap [ rml:reference "Att5" ] ];];
  rr:termType rr:Literal.
```

S1

Att1	Att2	Att3	Att4	Att5	Att...
...	DGCR6L	...	...	22:20302597-20302597	...
...	HMCN1	...	...	1:186072702-186072702	...
...	SLC5A10	...	...	17:18874996-18874996	...
...	...	...	...	...	...

Attribute-based Transformation

```
<TriplesMap1>
 rml:logicalSource [rml:source"Projected-S1_join_O1.csv";
    rml:referenceFormulation ql:CSV ];
  rr:subjectMap [rr:template "http://ex.com/{Att2}"];
  rr:predicateObjectMap [
    rr:predicate ex:p1;
    rr:objectMap [
       rml:reference "Out1" ];].
```

Projected-S1_join_O1

Att2	Att5	Out1
DGCR6L	22:20302597-20302597	22~20302597
HMCN1	1:186072702-186072702	1~186072702
SLC5A10	17:18874996-18874996	17~18874996
...	...	...

Optimization and Transformation: Composite-function-based Transformation

```
<TriplesMap1>
  rml:logicalSource [ rml:source "S1.csv";
          rml:referenceFormulation ql:CSV ];
  rr:subjectMap [rr:template "http://ex.com/{Att1}"];
  rr:predicateObjectMap [
    rr:predicate ex:p1;
    rr:objectMap <FunctionMap1> ];
<FunctionMap1>
  fnml:functionValue [
  rml:logicalSource [ rml:source "S1.csv";
          rml:referenceFormulation ql:CSV ];
  rr:predicateObjectMap [
    rr:predicate fno:executes ;
    rr:objectMap [ rr:constant ex:function1 ]];
  rr:predicateObjectMap [
    rr:predicate ex:column;
    rr:objectMap [ rml:reference "Att2" ] ];].
  rr:predicateObjectMap [
    rr:predicate ex:column;
    rr:objectMap <FunctionMap2> ];].

<FunctionMap2>
   fnml:functionValue [
   rml:logicalSource [ rml:source "S1.csv";
        rml:referenceFormulation ql:CSV ];
   rr:predicateObjectMap [
     rr:predicate fno:executes ;
     rr:objectMap [ rr:constant ex:function2 ]];
   rr:predicateObjectMap [
     rr:predicate ex:column;
     rr:objectMap [ rml:reference "Att5" ] ];].
```

S1

Att1	Att2	Att3	Att4	Att5	...
COSV50619134	DGCR6L	...	...	22:20302597-20302597	...
COSV54901969	HMCN1	...	...	1:186072702-186072702	...
COSV58755801	SLC5A10	...	...	17:18874996-18874996	...
...	...	...	...	...	...

Intermediate result

O1

Out2	Att2	Out1
22~20302597	DGCR6L	DGCR6L~22~20302597
1~186072702	HMCN1	HMCN11~186072702
17~18874996	SLC5A10	SLC5A1017~18874996
...	...	...

Intermediate result

O2

Att5	Out2
22:20302597-20302597	22~20302597
1:186072702-186072702	1~186072702
17:18874996-18874996	17~18874996
...	...

Composite-function-based Transformation

```
<TriplesMap1>
 rml:logicalSource [ rml:source "Projected-S1_join_O1 .csv";
          rml:referenceFormulation ql:CSV ];
 rr:subjectMap [rr:template "http://ex.com/{Att1}"];
 rr:predicateObjectMap [
        rr:predicate ex:p1;
        rr:objectMap [ rml:reference "Out1" ]].
```

Projected-S1_join_O1

Att1	Out2	Att2	Out1
COSV50619134	22~20302597	DGCR6L	DGCR6L~22~20302597
COSV54901969	1~186072702	HMCN1	HMCN11~186072702
COSV58755801	17~18874996	SLC5A10	SLC5A1017~18874996
...	...	...	...

Features:

- Is equipped by an optimization planner for different mapping constructions
- Able to interpret and execute composite functions
- Able to interpret the list of outputs (which is a limitation in current RML language)
- Execute functions and transforms the data integration system (DIS) into a function-free DIS efficiently

Application:

- Users can easily add their own library of functions without requiring to understand the implementation of the tool e.g. EABlock functions

https://dl.acm.org/doi/abs/10.1145/3477314.3507132

Demo: https://tib.eu/cloud/s/ikjiHyf8RNrEHSY

- Dragoman can be adopted by any RML-compliant knowledge graph creation pipeline

https://github.com/SDM-TIB/Dragoman

Hands-on:

```
> git clone https://github.com/SDM-TIB/ESWC-Demo-Dragoman
> cd ESWC-Demo-Dragoman
```

```
> mkdir dragoman
> cd dragoman
> git clone https://github.com/SDM-TIB/ESWC-Demo-Dragoman.git
> cd ESWC-Demo-Dragoman
> docker network create eswc2022
> docker-compose up -d
> docker ps
```

Now you should be able to see the dragoman container

Running Dragoman with the sample DIS

```
> docker exec -it dragoman python3 /app/run_translator.py /data/configfile.ini
```

```
> cd output
> less mapping_sample-GTFS_transferred_mapping.ttl
> less mapping_sample-GTFS_PROJECT1.csv
> less mapping_sample-GTFS_PROJECT2.csv
```

Checking the output

Lunch Break


public-kg-construct@w3.org


kgc_workshop

J2RM

Ontology-based JSON-to-RDF Mappings

Sergio José Rodríguez Méndez

Sergio.RodriguezMendez@anu.edu.au

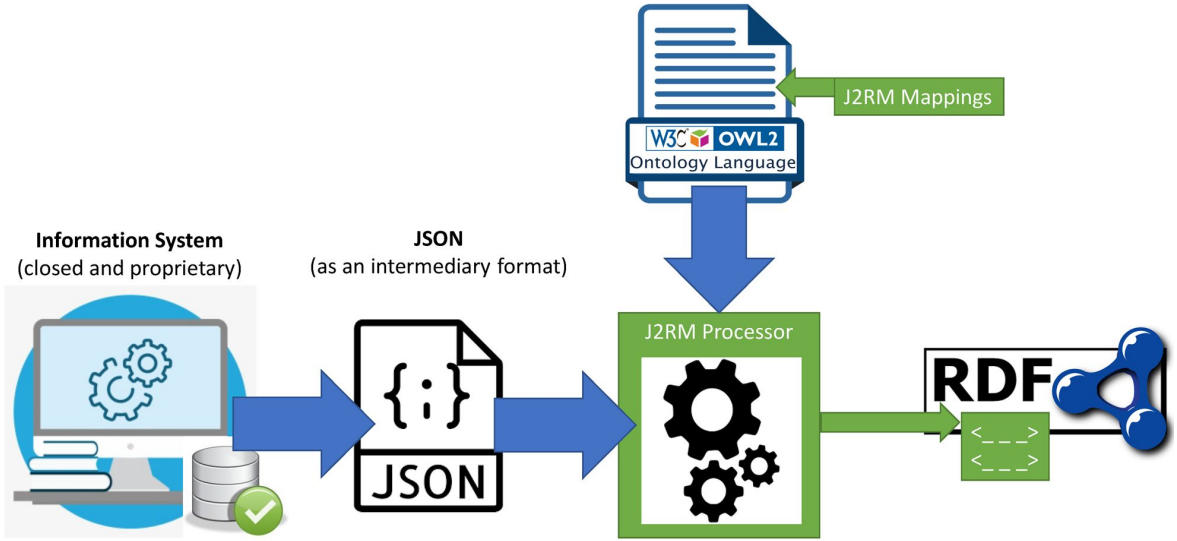
@sjrm_142857

Australian National University

School of Computing

J2RM: JSON-to-RDF Mappings

- JSON Pointer-based mappings embedded in the ontology file as annotation properties.
  - Classes and individuals: *identifiers*.
  - Object properties: *connection between identifiers.*
  - Datatype properties: *connect identifiers with values*.
- Traverses the input JSON structure: automatic distinction between objects and arrays.
- Primitives for text operations: URI generation, rdfs:label, etc.
- Support of multiples targets: *SPARQL endpoints* and *named graphs*.



Demo: https://bit.ly/38YI2Ls

Repo : https://w3id.org/kgcp/J2RM

SDM-RDFizer

Enrique Iglesias

Enrique Iglesias

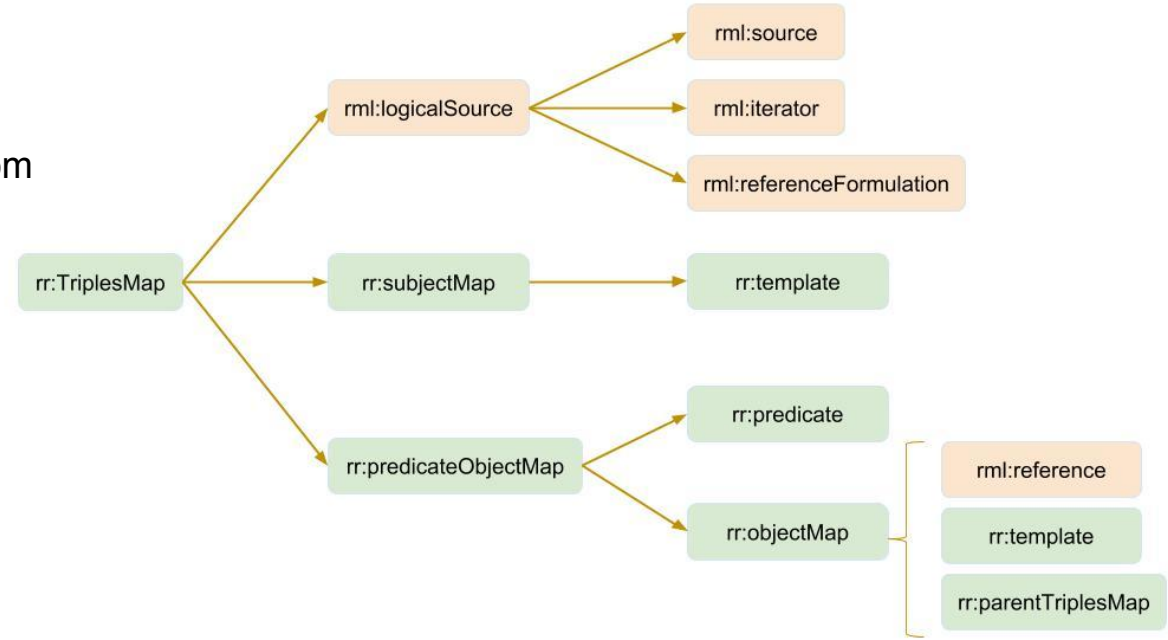
iglesias@l3s.de

@TIB_SDM

RDF Mapping Languages: Triple-based

Mapping language defined on top of R2RML

- Enables the collection of data from data sources in various formats
  - XML, JSON, CSV, RDB



RML Operators

```
<TriplesMap1>
  rml:logicalSource [ rml:source "dataSource1" ;
                      rml:referenceFormulation ql:CSV ];
  rr:subjectMap [
    rr:template "http://example.org/Gene/{Gene name}";
    rr:class ex:Gene];

  rr:predicateObjectMap [
    rr:predicate ex:geneLabel;
     rr:objectMap [ rml:reference "Gene name" ] ];

  rr:predicateObjectMap [
    rr:predicate ex:gene_isRelatedTo_sample;
    rr:objectMap [
      rr:parentTriplesMap <TriplesMap2> ] ].

<TriplesMap2>
  rml:logicalSource [ rml:source "dataSource1" ;
                      rml:referenceFormulation ql:CSV];
  rr:subjectMap [
    rr:template "http://example.org/Sample/{ID_sample}";
    rr:class ex:Sample] ;

  rr:predicateObjectMap [
    rr:predicate ex:sample_isTakenFrom_tumor;
    rr:objectMap [
      rr:parentTriplesMap <TriplesMap3>;
      rr:joinCondition [ rr:child "ID_sample";
                         rr:parent "ID_sample" ;];].

<TriplesMap3>
  rml:logicalSource [ rml:source "dataSource2";
                      rml:referenceFormulation ql:JSONPath
                      rml:iterator "$.[*]"];
  rr:subjectMap [
    rr:template "http://example.org/Tumor/{ID_tumor}";
    rr:class ex:Tumor ] .
```

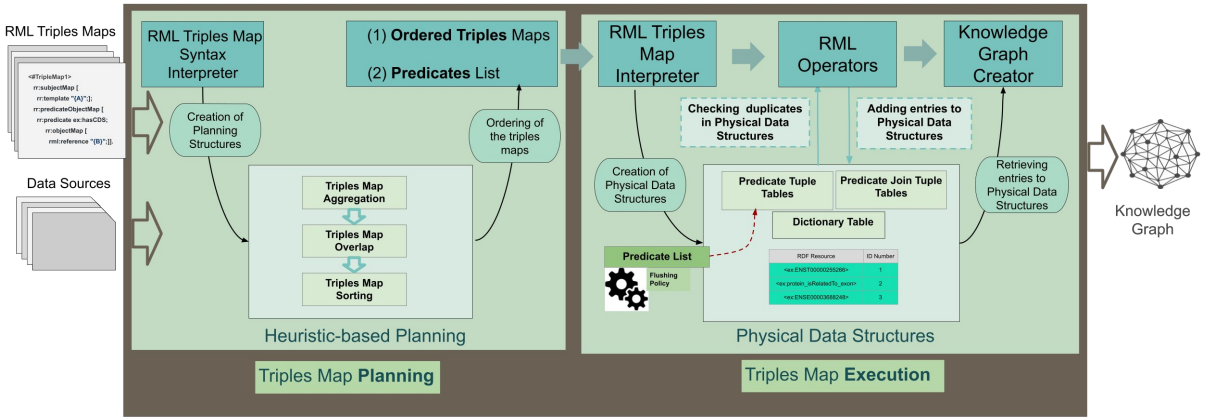
Simple Object Map (SOM) evaluates predicate object map in **triples maps**

Object Reference Map (ORM) implements a **reference** between **two triples maps**

Object Join Map (OJM) implements a **join condition** between two triples maps

Enrique Iglesias, Samaneh Jozashoori, David Chaves-Fraga, Diego Collarana, Maria-Esther Vidal. **SDM-RDFizer: An RML Interpreter for the Efficient Creation of RDF Knowledge Graphs.** ACM CIKM 2020.

DECLARATIVE KNOWLEDGE GRAPH CREATION



Knowledge Graph Creation Framework

Physical Data Structures avoid the generation of duplicated triples

- **Predicate Tuple Table (PTT):** for each predicate **p**, stores all the triples generated so far
- **Predicate Join Table (PJTT):** stores the subjects of the triples generated by a join.
- **Dictionary Table (DT):** encodes RDF resources with an identification number.
- **Predicate List (PL):** Determines which predicates should be flushed from the PTT.
- **Ordered Triples Maps List (OTML):** Determines the order of execution of the Triples Maps.

SDM-RDFizer implements **three physical operators**:

- Simple Object Map (SOM)
- Object Reference Map (ORM)
- Object Join Map (OJM)

Iglesias et al. SDM-RDFizer: An RML Interpreter for the Efficient Creation of RDF Knowledge Graphs. ACM CIKM 2020.

Organized Triples Maps List (OTML)

```
<TriplesMap1>
   rml:logicalSource [ rml:source "dataSource1"
                       rml:referenceFormulation ql:CSV];
    rr:subjectMap [
      rr:template "http://example.org/Gene/{Gene Name}";
      rr:class ex:Gene];

<TriplesMap2>
   rml:logicalSource [ rml:source "dataSource1"
                       rml:referenceFormulation ql:CSV];
    rr:subjectMap [
      rr:template "http://example.org/Sample/{ID_sample}";
      rr:class ex:Sample];

<TriplesMap3>
   rml:logicalSource [ rml:source "dataSource2"
                       rml:referenceFormulation ql:JSONPath];
    rr:subjectMap [
      rr:template "http://example.org/Tumor/{ID_tumor}";
      rr:class ex:Tumor];
```

Organized Triples Maps List

- Groups the triples maps by data source

Hash table:

- **Outer Key** format
- **Inner Key** data source
- **Value** list of triples maps.

Outer Key	Inner Key	Value

Organized Triples Maps List (OTML)

```
<TriplesMap1>
   rml:logicalSource [ rml:source "dataSource1"
                       rml:referenceFormulation ql:CSV];
    rr:subjectMap [
      rr:template "http://example.org/Gene/{Gene Name}";
      rr:class ex:Gene];

<TriplesMap2>
   rml:logicalSource [ rml:source "dataSource1"
                       rml:referenceFormulation ql:CSV];
    rr:subjectMap [
      rr:template "http://example.org/Sample/{ID_sample}";
      rr:class ex:Sample];

<TriplesMap3>
   rml:logicalSource [ rml:source "dataSource2"
                       rml:referenceFormulation ql:JSONPath];
    rr:subjectMap [
      rr:template "http://example.org/Tumor/{ID_tumor}";
      rr:class ex:Tumor];
```

Organized Triples Maps List

- Groups the triples maps by data source

Hash table:

- **Outer Key** format
- **Inner Key** data source
- **Value** list of triples maps.

Outer Key	Inner Key	Value
CSV	dataSource1	[TriplesMap1, TriplesMap2]
JSON	dataSource2	[TriplesMap3]

Organized Triples Maps List (OTML)

Organized Triples Maps List

- Groups the triples maps by data source

Sorting Heuristic:

- RML mapping rules are reordered in a way that the **most selective** mapping rules are evaluated first while **non-selective rules** are considered at the end. In other words, we seek to execute the data source with the less amount of triples maps first so that the data source can be unmounted as soon as possible.

Outer Key	Inner Key	Value
CSV	dataSource1	[TriplesMap1, TriplesMap2]
JSON	dataSource2	[TriplesMap3]

Outer Key	Inner Key	Value
JSON	dataSource2	[TriplesMap3]
CSV	dataSource1	[TriplesMap1, TriplesMap2]

Predicates List (PL)

Predicates List
- Groups the triples maps by predicate

Hash table:
- **Key** encoding **predicate**
- **Value** list of triples maps.

```
<TriplesMap1>
  rml:logicalSource [ rml:source "dataSource1" ];
   rr:subjectMap [
    rr:template "http://example.org/Gene/{Gene Name}";
    rr:class ex:Gene];
   rr:predicateObjectMap [
    rr:predicate ex:geneLabel;
    rr:objectMap [ rml:reference "Gene Name" ] ];
   rr:predicateObjectMap [
    rr:predicate ex:gene_isRelatedTo_sample;
    rr:objectMap [ rr:parentTriplesMap  <TriplesMap2>] ].
```

Key	Value

Predicates List (PL)

Predicates List
- Groups the triples maps by predicate

Hash table:
- **Key** encoding **predicate**
- **Value** list of triples maps.

```
<TriplesMap1>
  rml:logicalSource [ rml:source "dataSource1" ];
   rr:subjectMap [
    rr:template "http://example.org/Gene/{Gene Name}";
    rr:class ex:Gene];
   rr:predicateObjectMap [
    rr:predicate ex:geneLabel;
    rr:objectMap [ rml:reference "Gene Name" ] ];
   rr:predicateObjectMap [
    rr:predicate ex:gene_isRelatedTo_sample;
    rr:objectMap [ rr:parentTriplesMap  <TriplesMap2>] ].
```

Key	Value
rdf:type_ex:Gene	[TriplesMap1]
ex:geneLabel	[TriplesMap1]
ex:gene_isRelatedTo_sample	[TriplesMap1]

Predicate Tuple Table (PTT)

Predicate Tuple Table (PTT)

- stores RDF triples for each predicate generated so far
- **Key** encoding **subject** and **object**

```
<TriplesMap1>
   rml:logicalSource [ rml:source "dataSource1" ];
    rr:subjectMap [
      rr:template "http://example.org/Gene/{Gene Name}";
      rr:class ex:Gene];
    rr:predicateObjectMap [
      rr:predicate ex:geneLabel;
      rr:objectMap [ rml:reference "Gene Name" ] ];
```

<http://example.org/Gene/PHF12_ET00000268756> <ex:geneLabel> “PHF12_ET00000268756”.
<http://example.org/Gene/ALDH3A1_ET00000395555> <ex:geneLabel> “ALDH3A1_ET00000395555”.

Key
encode(http://example.org/Gene/PHF12_ET00000268756, PHF12_ET00000268756)
encode(http://example.org/Gene/ALDH3A1_ET00000395555, ALDH3A1_ET00000395555)

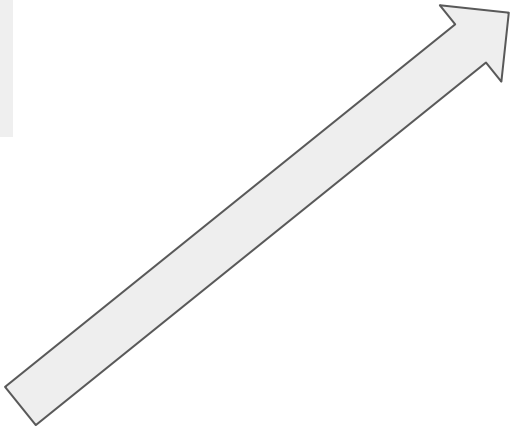
PTT ex:geneLabel

Predicate Tuple Table (PTT)

```
<TriplesMap1>
  rml:logicalSource [ rml:source "dataSource1" ];
   rr:subjectMap [
     rr:template "http://example.org/Gene/{Gene Name}";
     rr:class ex:Gene];
   rr:predicateObjectMap [
     rr:predicate ex:geneLabel;
     rr:objectMap [ rml:reference "Gene Name" ] ];
```

Key	Value
ex:geneLabel	[TriplesMap1]

Predicate List



Key	Value
ex:geneLabel	[]

Key	Value

PTT ex:geneLabel

Predicate Join Tuple Table (PJTT)

```
<TriplesMap2>
   rml:logicalSource [ rml:source "dataSource1" ];
   rr:subjectMap [
     rr:template
"http://example.org/Sample/{ID_sample}";
     rr:class ex:Sample] ;
   rr:predicateObjectMap [
     rr:predicate ex:sample_isTakenFrom_tumor;
     rr:objectMap [
       rr:parentTriplesMap <TriplesMap3>;
       rr:joinCondition [ rr:child "ID_sample";
                          rr:parent "ID_sample" ;];].

  <TriplesMap3>
    rml:logicalSource [ rml:source "dataSource2" ];
    rr:subjectMap [
      rr:template "http://example.org/Tumor/{ID_tumor}";
      rr:class ex:Tumor ] .
```

Predicate Join Tuple Table (PJTT)

- stores values generated during execution of a join condition between two RML triple maps.

Index Hash table to the Source S2 of the parentTM:

- **Key** encoding of each of **value(s)** of the **attributes** in the **join condition**
- **Value** set with the subject values in S2 associated with the values of the attributes in the hash key

Object Join Map (OJM)

Predicate Join Tuple Table (PJTT)

```
<TriplesMap2>
   rml:logicalSource [ rml:source "dataSource1" ];
   rr:subjectMap [
     rr:template
"http://example.org/Sample/{ID_sample}";
     rr:class ex:Sample] ;
   rr:predicateObjectMap [
     rr:predicate ex:sample_isTakenFrom_tumor;
     rr:objectMap [
       rr:parentTriplesMap <TriplesMap3>;
       rr:joinCondition [ rr:child "ID_sample";
                          rr:parent "ID_sample" ;];].

  <TriplesMap3>
   rml:logicalSource [ rml:source "dataSource2" ];
   rr:subjectMap [
     rr:template "http://example.org/Tumor/{ID_tumor}";
     rr:class ex:Tumor ] .
```

dataSource1

Gene Name	ID_sample
ALDH3A1_ET00000395555	2193351
PHF12_ET00000268756	2193351
PHF12_ET00000268756	2196270

dataSource2

ID_sample	ID_tumor
2193351	1455465
2193351	2064548
2196270	2061629

Predicate Join Tuple Table (PJTT)

```
<TriplesMap2>
   rml:logicalSource [ rml:source "dataSource1" ];
   rr:subjectMap [
     rr:template
"http://example.org/Sample/{ID_sample}";
     rr:class ex:Sample] ;
   rr:predicateObjectMap [
     rr:predicate ex:sample_isTakenFrom_tumor;
     rr:objectMap [
       rr:parentTriplesMap <TriplesMap3>;
       rr:joinCondition [ rr:child "ID_sample";
                          rr:parent "ID_sample" ;];].
  <TriplesMap3>
   rml:logicalSource [ rml:source "dataSource2" ];
   rr:subjectMap [
     rr:template "http://example.org/Tumor/{ID_tumor}";
     rr:class ex:Tumor ] .
```

dataSource2

ID_sample	ID_tumor
2193351	1455465
2193351	2064548
2196270	2061629

TripleMap2_ID_sample	
[2193351]	[1455465,2064548]
[2196270]	[2061629]

JPTT TripleMap2_ID_sample

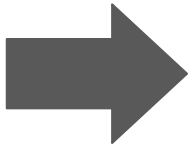
Dictionary Table

Dictionary Table
- Encodes each RDF resource with an identification number

Hash table:
- **Key** RDF resource
- **Value** identification number in base 36

Key
encode(http://example.org/Gene/PHF12_ET00000268756_PHF12_ET00000268756)
encode(http://example.org/Gene/ALDH3A1_ET00000395555_ALDH3A1_ET00000395555)

PTT ex:geneLabel



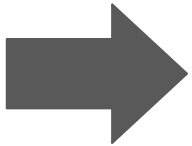
Dictionary Table

Key	Value
http://example.org/Gene/PHF12_ET00000268756	1
ex:geneLabel	2
“PHF12_ET00000268756”	3
http://example.org/Gene/ALDH3A1_ET00000395555	4
“ALDH3A1_ET00000395555”	5

Dictionary Table

Dictionary Table
- Encodes each RDF resource with an identification number

Hash table:
- **Key** RDF resource
- **Value** identification number in base 36



Dictionary Table

Key	Value
http://example.org/Gene/PHF12_ET00000268756	1
ex:geneLabel	2
“PHF12_ET00000268756”	3
http://example.org/Gene/ALDH3A1_ET00000395555	4
“ALDH3A1_ET00000395555”	5

Key
encode(http://example.org/Gene/PHF12_ET00000268756_PHF12_ET00000268756)
encode(http://example.org/Gene/ALDH3A1_ET00000395555_ALDH3A1_ET00000395555)

PTT ex:geneLabel

Key
1_3
4_5

PTT 2

Physical Operators

Simple Object Map (SOM):

Triples Map **tm1** defines predicate **p** on logical source **S**

and **tm1** subjectMap is **f1(att1)**

and **tm1** objectMap for **p** is **f2(att2)**

For each row in S

a. Create an RDF triple **t=(f1(row.att1),p,f2(row.att1))**
b. If **encode(f1(row.att1),f2(row.att1))** does not belong to the PPT for **p**
   i. **Add encode(f1(row.att1),f2(row.att1))** to PPT for **p**
   ii. **Output (f1(row.att1),p,f2(row.att1))** to the KG

<http://example.org/Gene/ALDH3A1_ET00000395599> <ex:geneLabel> “ALDH3A1_ET00000395599”.

+

Key
encode(http://example.org/Gene/PHF12_ET00000268756, PHF12_ET00000268756)
encode(http://example.org/Gene/ALDH3A1_ET00000395555, ALDH3A1_ET00000395555)

Key
encode(http://example.org/Gene/PHF12_ET00000268756, PHF12_ET00000268756)
encode(http://example.org/Gene/ALDH3A1_ET00000395555, ALDH3A1_ET00000395555)
encode(http://example.org/Gene/ALDH3A1_ET00000395599, ALDH3A1_ET00000395599)

Physical Operators

Object Reference Map (ORM):

Triples Map **tm1 refers** to Triples Map **tm2** to define predicate **p** on logical source **S**:

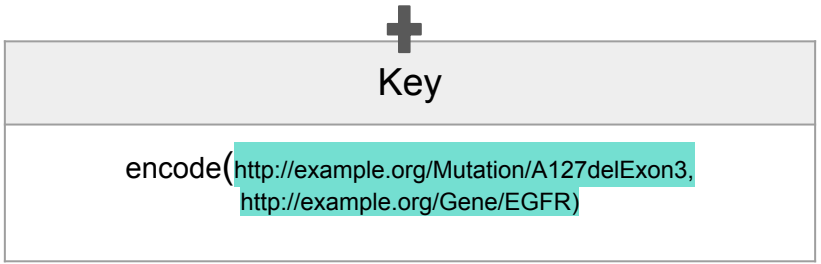
and subject of **tm1** is defined as **f1(att1)**

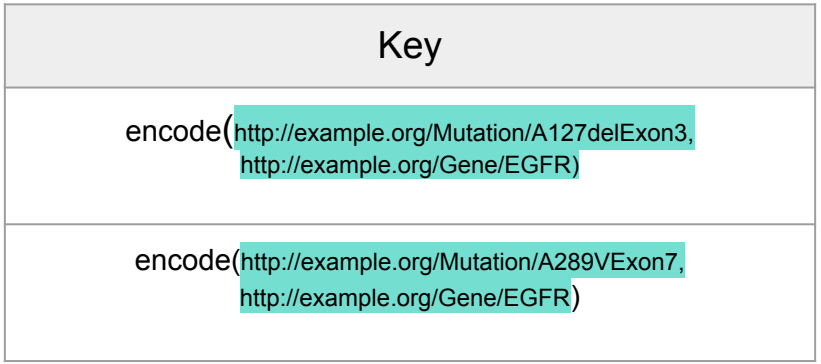
and subject of **tm2** is defined as **f2(att2)**

For each row in S

a. Create an RDF triple **t=(f1(row.att1),p,f2(row.att1))**
b. If **encode(f1(row.att1),f2(row.att1))** does not belong to the PPT for **p**
   i. **Add encode(f1(row.att1),f2(row.att1))** to PPT for **p**
   ii. **Output (f1(row.att1),p,f2(row.att1))** to the KG

<http://example.org/Mutation/A289VExon7> <ex:isMutation> <http://example.org/Gene/EGFR>





Physical Operators- OJM

```
<TriplesMap2>
  rml:logicalSource [ rml:source "dataSource1" ];
  rr:subjectMap [
    rr:template
"http://example.org/Sample/{ID_sample}";
    rr:class ex:Sample] ;
  rr:predicateObjectMap [
    rr:predicate ex:sample_isTakenFrom_tumor;
    rr:objectMap [
      rr:parentTriplesMap <TriplesMap3>;
      rr:joinCondition [ rr:child "ID_sample";
                         rr:parent "ID_sample" ;];].
<TriplesMap3>
  rml:logicalSource [ rml:source "dataSource2" ];
  rr:subjectMap [
    rr:template "http://example.org/Tumor/{ID_tumor}";
    rr:class ex:Tumor ] .
```

dataSource1

Gene Name	ID_sample
ALDH3A1_ET00000395555	2193351
PHF12_ET00000268756	2193351
PHF12_ET00000268756	2196270

dataSource2

ID_sample	ID_tumor
2193351	1455465
2193351	2064548
2196270	2061629

For each row1 in dataSource1

If there is an entry in the attributes of the join condition

JPTT TripleMap2_ID_sample

TripleMap2_ID_sample	
[2193351]	[1455465,2064548]
[2196270]	[2061629]

Then extract the values associated with the entry and generate the corresponding entries in PPT

Key
encode(http://example.org/Sample/2193351,http://example.org/Tumor/1455465)
encode(http://example.org/Sample/2193351,http://example.org/Tumor/2064548)
encode(http://example.org/Sample/2196270,http://example.org/Tumor/2061629)

Empirical Evaluation

SDM-RDFizer Version	Description
SDM-RDFizer **v3.4**	This version of the SDM-RDFizer only uses the PTT and PJTT tables.
SDM-RDFizer^{-} naive RML operators	This version of the SDM-RDFizer implements the PTT and PJTT tables not as dictionaries but as arrays.
SDM-RDF+HDT	This version is the SDM-RDFizer **v3.4** with data compression (Dictionary Table)
SDM-RDF+HDT+Flush+Order (**v4.0**)	The current standard version of the SDM-RDFizer. This version all the defined data structures and operators.

Conclusions:

For simple mapping with data sources with low duplicate rate, it is preferable to use SDM-RDFizer+HDT.

- Ordering causes too much overhead.

For complex mapping and data sources with high duplicate rates, it is preferable to use SDM-RDFizer+HDT+Flush+Order.

- By keeping the PTT as small as possible, the removal of duplicate is quicker.

Future Work:

Flushing policies for PJTT.

Storing PTT into RDB.

Extend proposed strategies to take into consideration observational data.

SDM-RDFizer as a Resource

Availability

TIB SDM

TIB LEIBNIZ INFORMATION CENTRE FOR SCIENCE AND TECHNOLOGY UNIVERSITY LIBRARY

zenodo

Apache license 2.0 license

Utility

Docker image

Docker Hub

docker

Documentation in TIB SDM

Video

Adoption and Usability

Several European and national funded projects are already using the Knowledge Graph Creation Pipeline

iASiS Big Data for Precision Medicine

BIGMedilytics

CLARIFY

P4-LUCAT

CoyPu

ImProViT

PLATOON

TIB SDM https://github.com/SDM-TIB

https://www.youtube.com/watch?v=DpH_57M1uOE

Hands-on:

Demo Repository

https://github.com/SDM-TIB/ESWC-Demo-SDM-RDFizer

Build Docker Container

docker-compose up -d

Run the SDM-RDFizer

docker exec -it sdmrdfizer python3 -m rdfizer -c configfile.ini

Download results from Container

docker exec -it sdmrdfizer cat graph/ESWC-DEMO.nt > ESWC-DEMO.nt

Thanks for your attention!

Enrique Iglesias


iglesias@l3s.de


@TIB_SDM

RocketRML

Umutcan Simsek

https://github.com/semantifyit/RocketRML


umutcan.simsek@sti2.at


@umutsims

What

- In-memory RML mapper written in Typescript
  - Fits well if your application is based on Javascript, Node.js…
  - Also published as an NPM package https://www.npmjs.com/package/rocketrml

- Available under CC-BY-SA 4.0 license

- Being actively maintained and used in commercial projects
  - e.g., Online GmbH uses it for building the German Tourism Knowledge Graph

Why and Why not

- What is cool about it
  - Particularly high performance with JSON sources
  - Supports faster implementations (with C++) of XPath
  - Supports JSONPath-Plus with the ^ operator for backwards traversal
    - very useful for mapping nested objects
  - Supports ad-hoc Javascript functions in FunctionMaps
  - Performance improvements via caching of JOINs

- What is not so cool about it
  - Limited support of the RML specification
    - e.g. No named graph support
  - No Native YARRRML support (planned feature)
  - Only CSV, XML and JSON sources are allowed

How: Hands on

- Option 1: RocketRML in browser: https://semantifyit.github.io/rml/
- Option 2: via an HTTP API: https://github.com/sumutcan/cost-tutorial-2022
  - Courtesy of ONLIM
- Option 3: via RocketRML Docker container: https://github.com/semantifyit/RocketRML

Coffee Break

public-kg-construct@w3.org

kgc_workshop

Chimera

Composable Pipelines for Semantic Data Transformation

Mario Scrocca (Cefriel)

mario.scrocca@cefriel.com

@mario_scrock

Cefriel

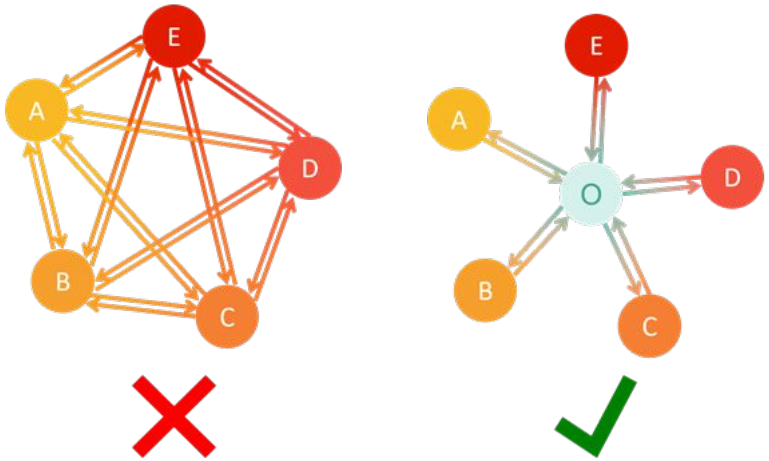
POLITECNICO DI MILANO

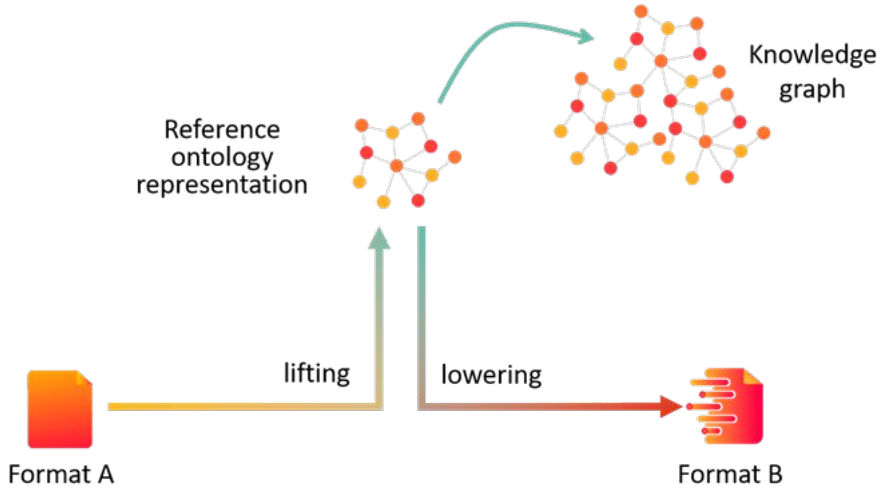
Chimera Use Case: **Semantic Data Conversion**

Many stakeholders with an interoperability need: **any-to-one centralized mapping approach**

- Lets stakeholders keep using their current legacy systems
- A stakeholder only needs to define lifting/lowering mappings to/from the reference ontology
- Knowledge graph as an additional valuable product of the conversion

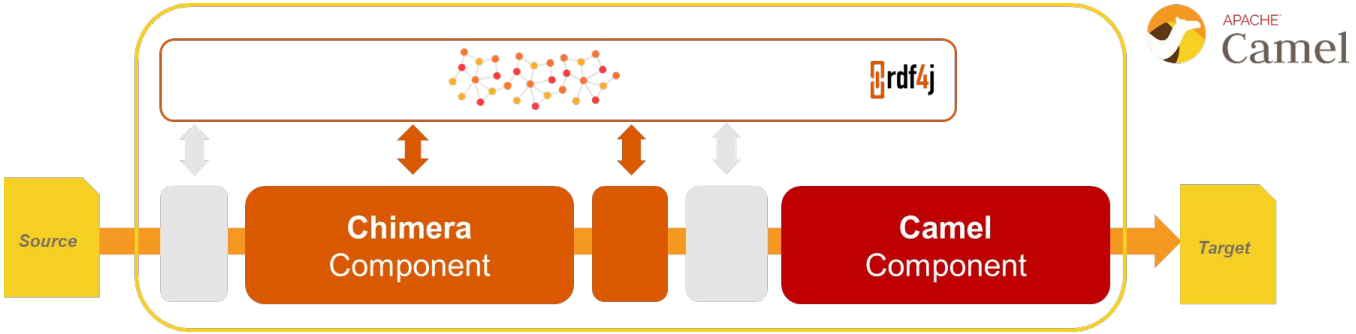
Goal: easily compose **conversion pipelines** with different requirements and involving **semantic web** technologies





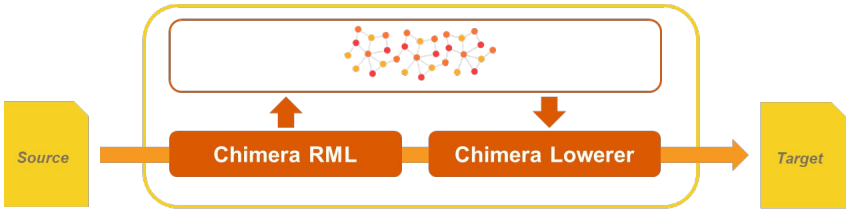
Video: https://youtu.be/R2ndUb4XtAI

Overview of Chimera



Chimera is a **modular** and **configurable solution** for the Apache Camel integration framework

- **Chimera components** minimise the effort required to specify and configure a pipeline
- Production-ready **Camel components** can be used to implement **Enterprise Integration Patterns** and data sources/sinks integration
- **Additional components** can be easily (implemented and) integrated, e.g., for pre-processing



Lifting: RML (fork of the *rmlmapper-java*)

Lowering: Custom implementation (repo) based on *Apache Velocity+SPARQL*

Discover more about Chimera

Distinctive features

- ✓ High configurability of pipelines allows satisfying different data integration requirements using Semantic Web Technologies
- ✓ Easy to integrate with existing data sources and sinks thanks to Apache Camel components

Limitations

- Steep learning curve due to different languages (Camel XML/DSL, RML, VTL, SPARQL,…)
- No graphical interface (yet) to compose pipelines

Chimera v3 will be released soon as a Camel Component

Watch the **video**: bit.ly/chimera-video

Chimera **tutorial**: bit.ly/chimera-tutorial

Chimera **repository**: bit.ly/kgc-chimera

Morph-KGC

Julián Arenas-Guerrero

julian.arenas.guerrero@upm.es

ArenasGuerreroJ

Acknowledgements:

ENTENTE

Main Characteristics

morph

Mapping languages: **R2RML, RML and RML-star**

RDF serializations: **N-TRIPLES, N-QUADS**

Operating Systems:

License: **Apache 2.0**

Supported Data Formats

morph

RELATIONAL DBs

TABULAR FILES

HIERARCHICAL FILES

Architecture

morph

Implemented in:

Built on top of:

Relational databases access with:

SQLA

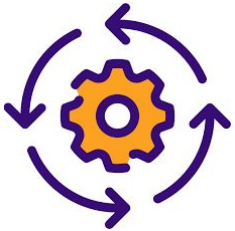
Why Morph-KGC

morph

- Simple and easy: PyPi

- Well-documented (we do not want users to get lost)

- Reliable and robust: continuous integration!

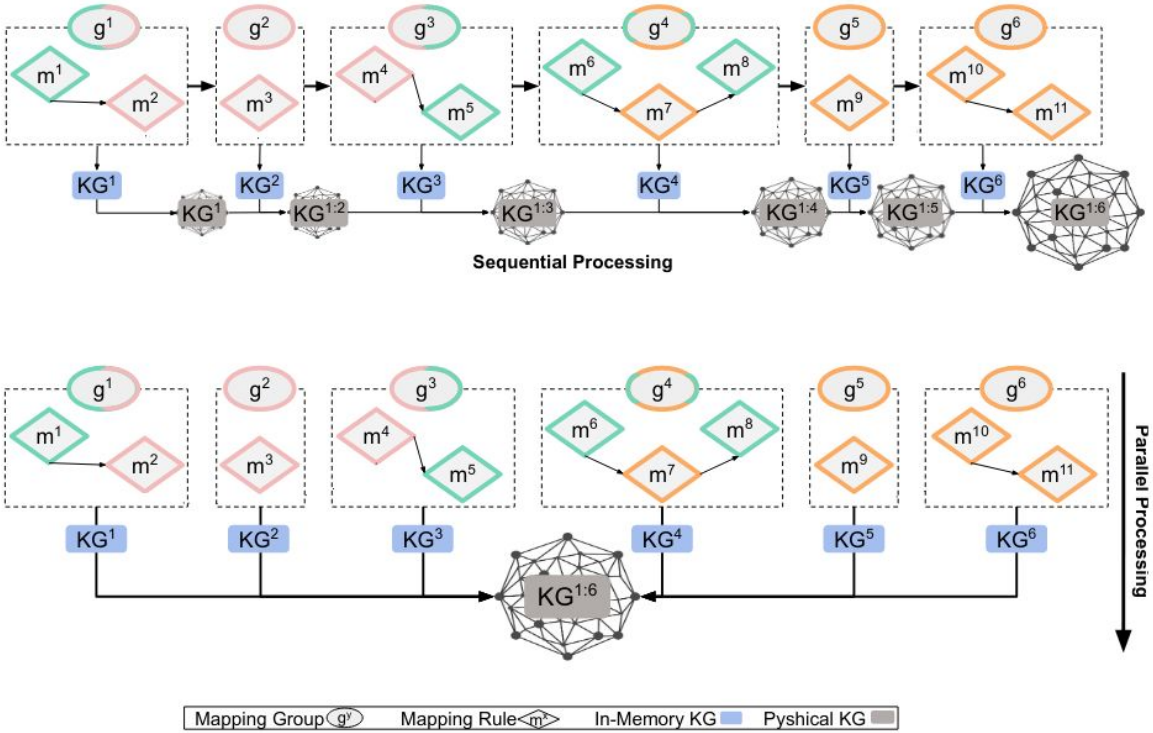


- Fast materialization (very fast!)

- Actively maintained

Efficient? How?

morph



Mapping partitioning:

- Sequential processing
- Parallel processing

Hands-on time!

morph

https://github.com/oeg-upm/morph-kgc/

https://pypi.org/project/morph-kgc/

Tutorial on Colab→ https://colab.research.google.com/drive/1ByFx_NOEfTZeaJ1Wtw3UwTH3H3-Sye2O?usp=sharing

RMLStreamer

Dylan Van Assche

dylan.vanassche@ugent.be

@DylanVanAssche

https://dylanvanassche.be

Data streams cannot be transformed into RDF

Data streams poses the following problems for most RDF generation approaches:

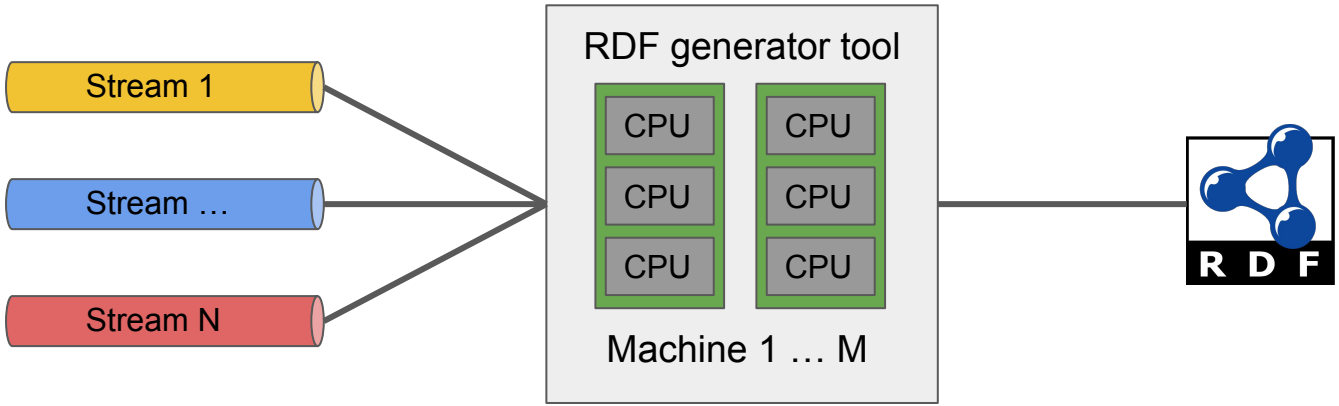
1. **Volume**
   The amount of data is enormous
2. **Velocity**
   Data changes rapidly
3. **Unbounded**
   Streams provide continuously new data

Problem 1: data volume

Tools have to scale horizontally and vertically to deal huge data volume:

Horizontal scaling
Distribute the processing among machines

Vertical scaling
Parallelize among CPU cores on the same machine



Problem 2: data velocity

Tools must be fast enough to transform data into RDF

Latency
Processing time must be lower than the fastest input stream

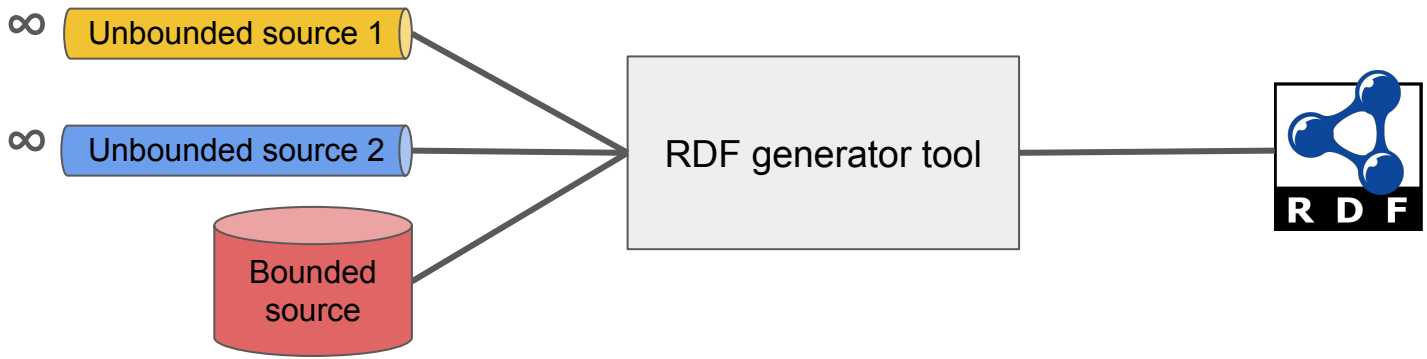
Velocity differences
Every stream has its own velocity, tools should block on the slowest stream

Problem 3: unbounded data

Streams are unbounded data, the provide an infinite amount of data

Computing resources
Streams can never be stored completely in memory because they may be unbounded

Joins
Joining multiple streams and bounded data requires additional effort of tools such as windowing



RMLStreamer executes RML rules in a streaming fashion

RMLStreamer aims to overcome the shortcomings of the RMLMapper:

- Memory limitations
- Support for unbounded data e.g. streams
- Joins among multiple bounded & unbounded data
- Scaling horizontally & vertically

https://github.com/RMLio/RMLStreamer

RMLStreamer executes RML rules in a streaming fashion

RMLStreamer is built upon Apache Flink, a widely used streaming framework for high performance & scalability

Executes in a parallel & streaming fashion RML mapping rules to generate RDF from heterogeneous data

RMLStreamer cleans your data through FnO functions

https://github.com/RMLio/RMLStreamer
https://flink.apache.org/

RMLStreamer executes RML rules in a streaming fashion

Data sources

- Bounded: files
- Unbounded: TCP, Kafka, MQTT streams

Data formats

- CSV
- JSON
- XML

https://github.com/RMLio/RMLStreamer

RMLStreamer: hands-on!

Requirements:

- Docker v19 or higher
- Docker-Compose v3 or higher
- RMLStreamer Jar
- Instructions are for Linux, but other OSes are supported as well

https://github.com/RMLio/RMLStreamer/blob/development/docker/README.md

RMLStreamer: hands-on!

1. Go to the `material/rmlstreamer` folder
2. Follow the `README.md` file:
   a. Start flink cluster: `docker-compose up`
   b. Open a browser to http://localhost:8081
   c. Click on '*Submit New Job*' in the Left menu
   d. Click '*+ Add New*' in the upper right corner
   e. Select the RMLStreamer Jar and wait until it is uploaded

We will go through these steps in the next slides.

Hands-on: Download RMLStreamer Jar

https://github.com/RMLio/RMLStreamer/releases/download/v2.3.0/RMLStreamer-2.3.0.jar

Hands-on: Start Flink cluster

```
$ docker-compose up
```

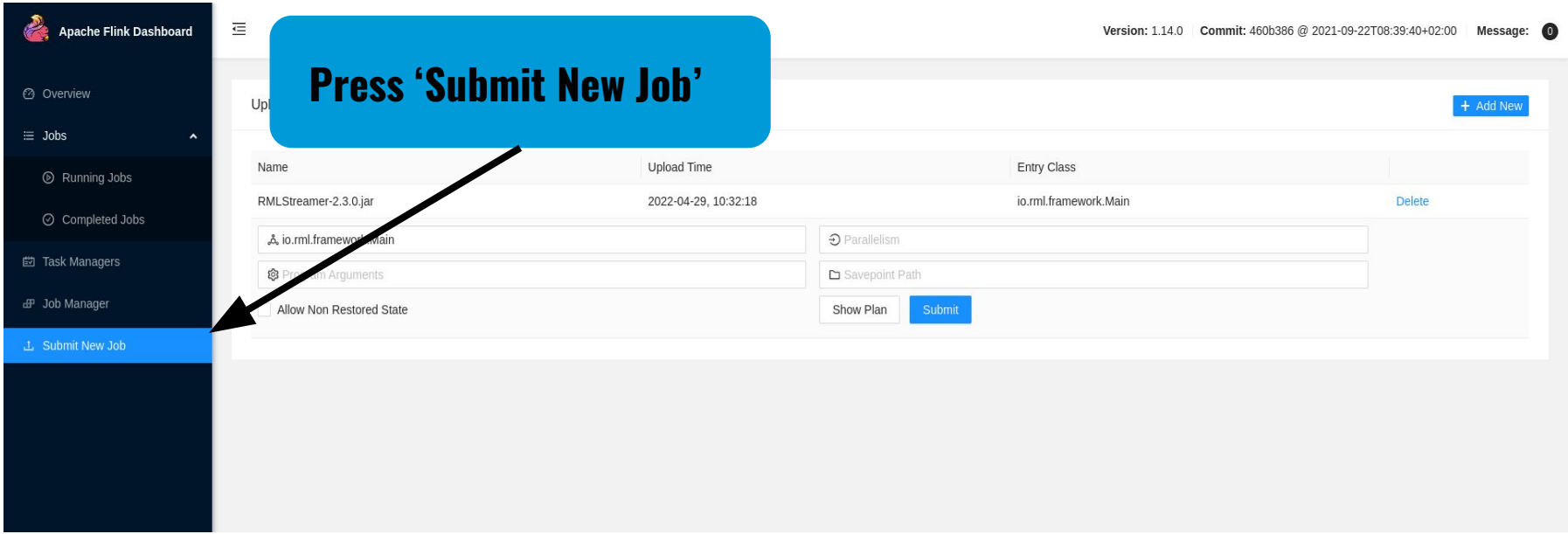
Wait a few seconds until Flink is fully started up…

Hands-on: Open a browser

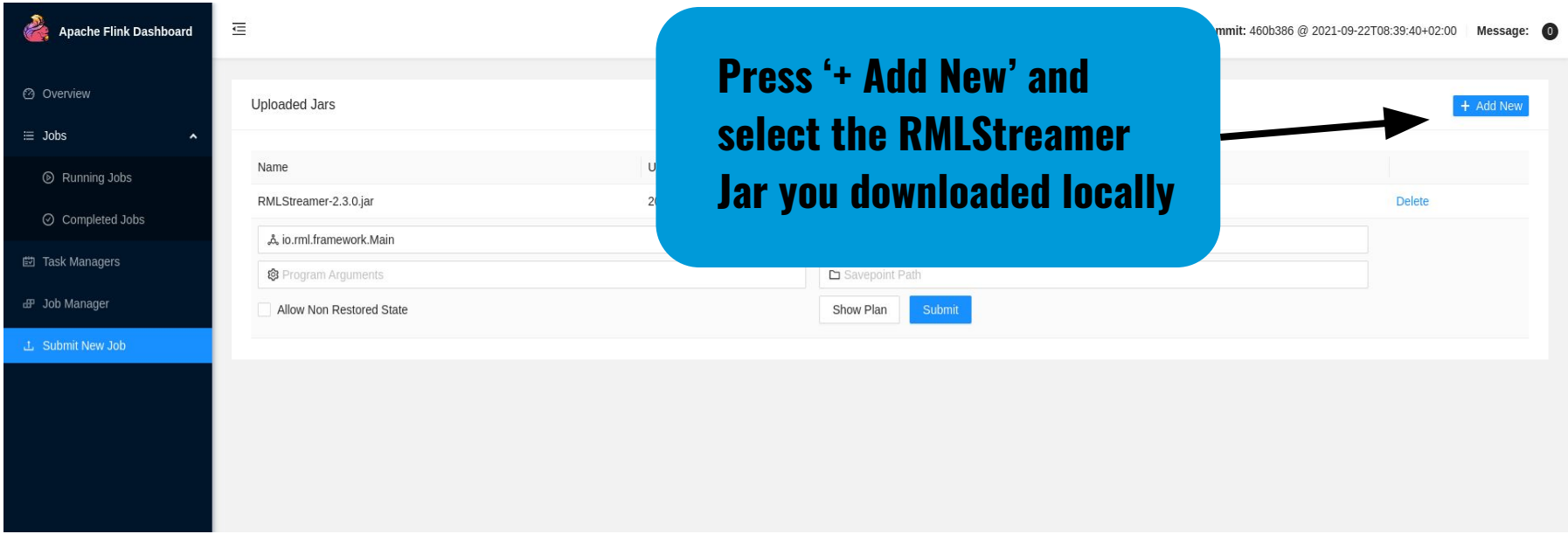
Open a browser to:

http://localhost:8081/

Hands-on: upload RMLStreamer Jar



Hands-on: upload RMLStreamer Jar



Hands-on: Copy demo data

1. Copy RML mapping rules to Docker container

```
$ docker cp data/example/mapping.rml.ttl \
rmlstreamer_taskmanager_1:/mnt/data/mapping.rml.ttl
```

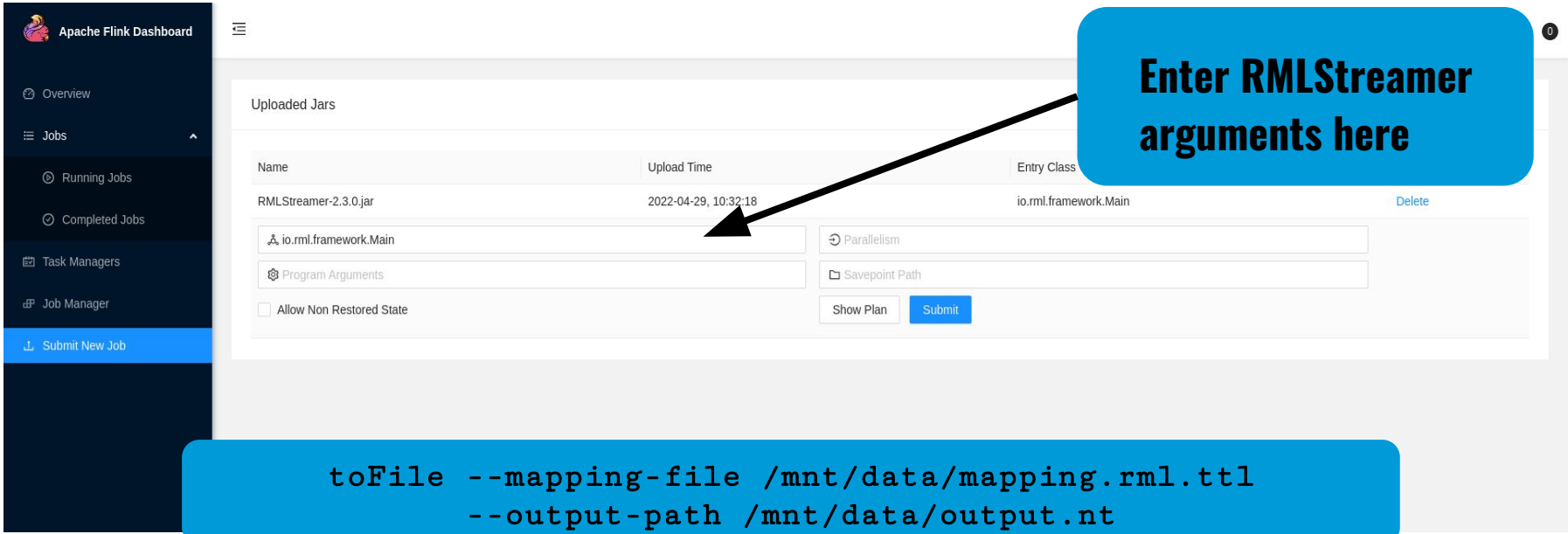
2. Copy input data to Docker container

```
$ docker cp data/example/input.json \
rmlstreamer_taskmanager_1:/mnt/data/input.json
```

Hands-on: give RMLStreamer permission to write

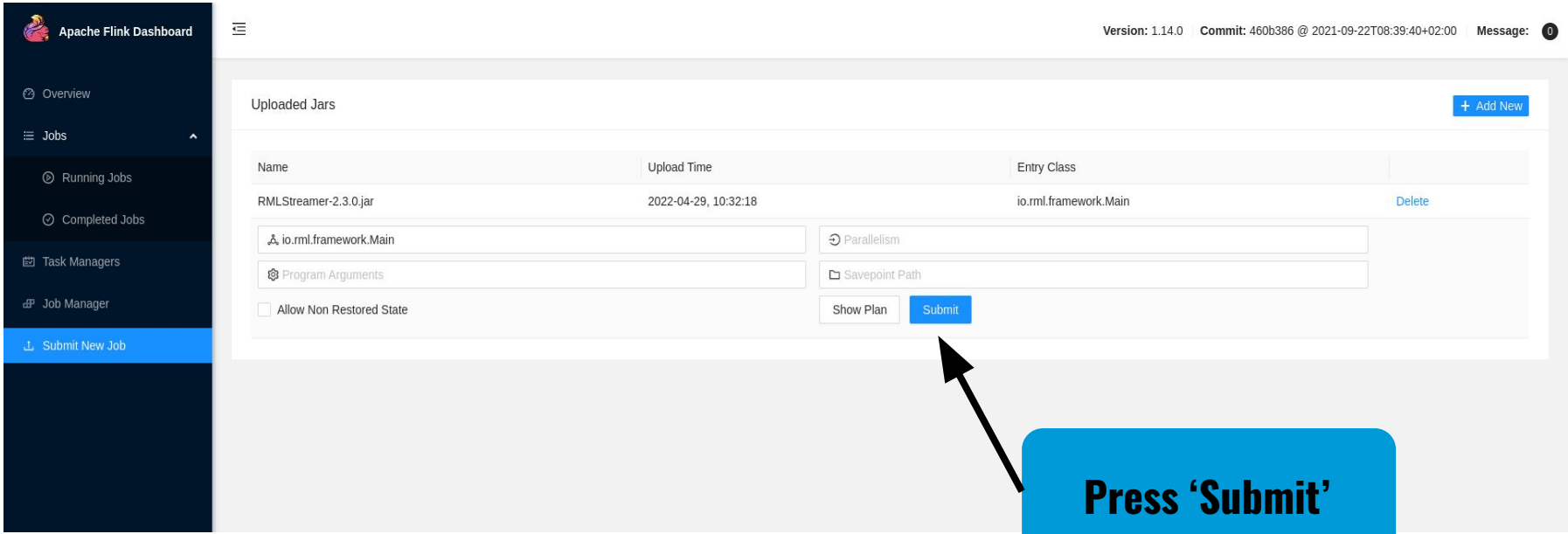
```
# chmod -R 777 \
/var/lib/docker/volumes/rmlstreamer_data/_data
```

Hands-on: Start RMLStreamer

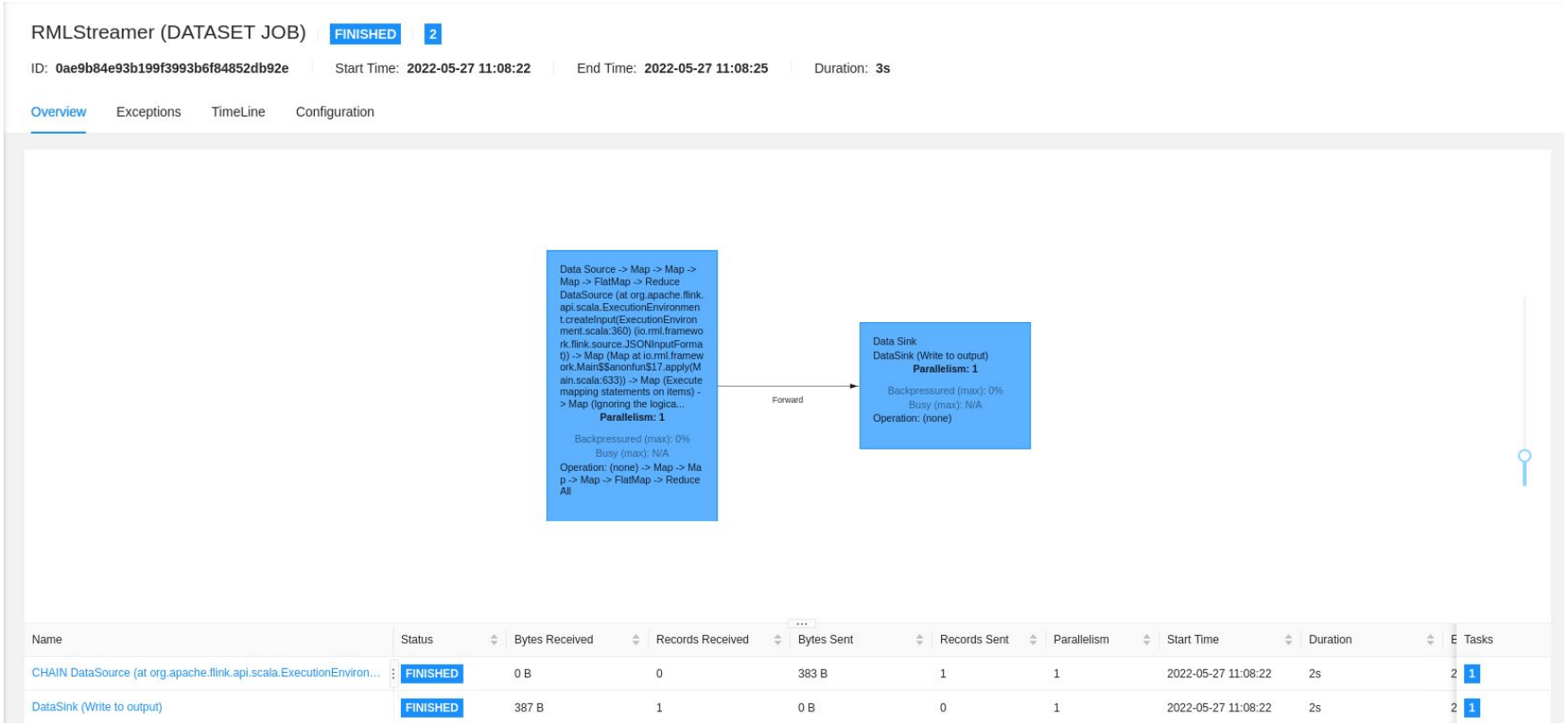



```
toFile --mapping-file /mnt/data/mapping.rml.ttl
       --output-path /mnt/data/output.nt
```

Hands-on: Start RMLStreamer



Hands-on: RMLStreamer in action



Hands-on: Get mapped RDF back

1. Copy output RDF from Docker container to your system back:

```
$ docker cp \
rmlstreamer_taskmanager_1:/mnt/data/output.nt .
```

2. Print results

```
$ cat output.nt

<http://example.com/Sue;Jones> <http://example.com/owes> "20.0E0" .
<http://example.com/Sue;Jones> <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
<http://xmlns.com/foaf/0.1/Person> .
<http://example.com/Bob;Smith> <http://example.com/owes> "30.0E0" .
<http://example.com/Bob;Smith> <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
<http://xmlns.com/foaf/0.1/Person> .
```

Hands-on: Try yourself with your own data!

- You can re-use the setup, only have to copy your own RML mapping rules & input data.
- Tip: re-use the mappings from the YARRRML & Matey tutorial if you want to have an example

https://github.com/RMLio/RMLStreamer/

Hands-on: GTFS example parallel