How do you construct a Knowledge Graph?

Ana Iglesias-Molina and David Chaves-Fraga

Half-day tutorial at ECAI2024

Ontology Engineering Group

POLITÉCNICA

USC UNIVERSIDADE DE SANTIAGO DE COMPOSTELA

CiTiUS Centro Singular de Investigación en Tecnoloxías Intelixentes

About us...

David Chaves-Fraga
Assistant Professor
CiTIUS / GSI
Universidade de Santiago de Compostela

Ana Iglesias-Molina
PhD Student
Ontology Engineering Group (OEG)
Universidad Politécnica de Madrid

KGC-CG in W3C

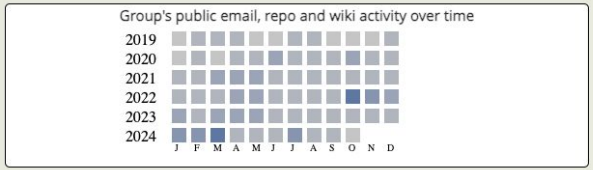
W3C Community Group Website

W3C Community Group GitHub

KNOWLEDGE GRAPH CONSTRUCTION COMMUNITY GROUP

The overall goal of this community group is to support its participants into developing better methods for Knowledge Graphs construction. The Community Group will (i) study current Knowledge Graph construction methods and implementations, (ii) identify the corresponding requirements and issues that hinter broader Knowledge Graph construction, (iii) discuss use cases, (iv) formulate guidelines, best practices and test cases for Knowledge Graph construction, (v) develop methods, resources and tools for evaluating Knowledge Graphs construction, and in general (vi) continue the development of the W3C-recommended R2RML language beyond relational databases. The proposed Community Group could be instrumental to advance research, increase the level of education and awareness and enable learning and participation with respect to Knowledge Graph construction.

kg-construct



Note: Community Groups are proposed and run by the community. Although W3C hosts these conversations, the groups do not necessarily represent the views of the W3C Membership or staff.

Chairs, when logged in, may publish draft and final reports. Please see report requirements.

biweekly meetings

Anastasia Dimou | Posted on: April 6, 2021

The Community Group meets every 2nd Monday at 15h CET.

The details to join the meetings:

Join Zoom Meeting
https://upm.zoom.us/j/87908976407

Tools for this group

- Mailing List
- IRC
- Github repositories
- RSS
- Contact This Group

Get involved

Anyone may join this Community Group. All participants in this group have signed the W3C Community Contributor License Agreement.

JOIN OR LEAVE THIS GROUP

Chairs


Anastasia Dimou


David Chaves-Fraga


Alessandro Negro

Participants (184)

View all participants

Technical requirements

Text Editor with YAML support (e.g., Sublime Text)

Google Colab / Jupyter Notebook

https://w3id.org/kg-construct/tutorials/ecai2024

Agenda

- RDF Knowledge Graphs and the Semantic Web
- RML: The RDF Mapping Language
- Coffee Break
- Hands-on session on KG Construction:
  - Mapping Data with YARRRML
  - Transforming Heterogeneous Data to RDF
- Resources for KG Construction
  - Ontology development, mapping supporters, KGC engines, benchmarks

Knowledge Graphs

14:00 - 14:30

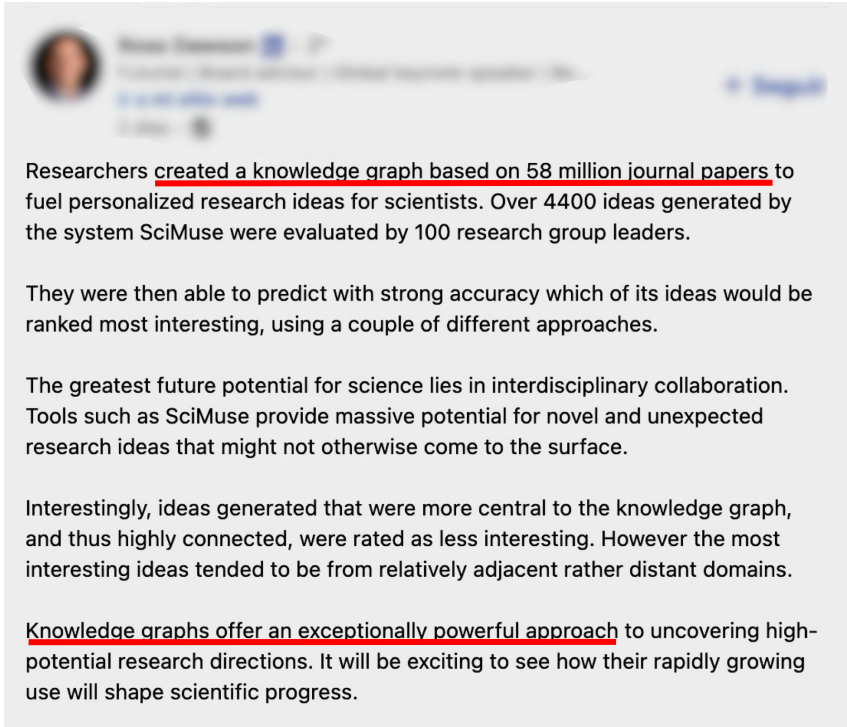
Knowledge Graph definition

Semantic Web technologies

KG Construction

History of mapping languages

My LinkedIn these days...



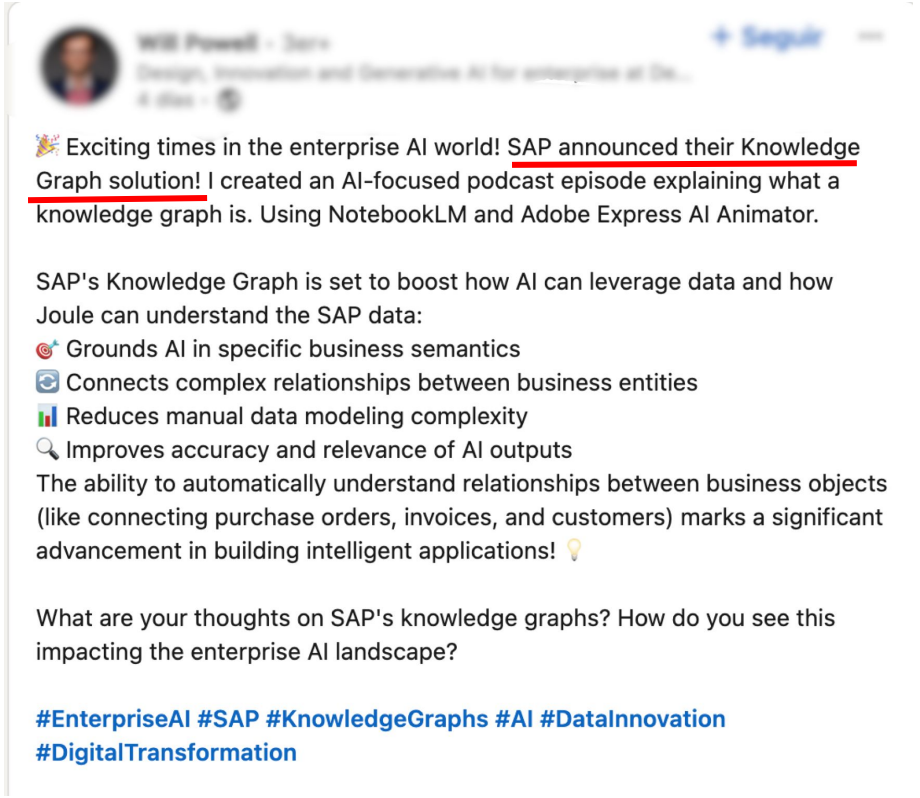
Researchers created a knowledge graph based on 58 million journal papers to fuel personalized research ideas for scientists. Over 4400 ideas generated by the system SciMuse were evaluated by 100 research group leaders.

They were then able to predict with strong accuracy which of its ideas would be ranked most interesting, using a couple of different approaches.

The greatest future potential for science lies in interdisciplinary collaboration. Tools such as SciMuse provide massive potential for novel and unexpected research ideas that might not otherwise come to the surface.

Interestingly, ideas generated that were more central to the knowledge graph, and thus highly connected, were rated as less interesting. However the most interesting ideas tended to be from relatively adjacent rather distant domains.

Knowledge graphs offer an exceptionally powerful approach to uncovering high-potential research directions. It will be exciting to see how their rapidly growing use will shape scientific progress.



🎉 Exciting times in the enterprise AI world! SAP announced their Knowledge Graph solution! I created an AI-focused podcast episode explaining what a knowledge graph is. Using NotebookLM and Adobe Express AI Animator.

SAP's Knowledge Graph is set to boost how AI can leverage data and how Joule can understand the SAP data:
🎯 Grounds AI in specific business semantics
🔄 Connects complex relationships between business entities
📊 Reduces manual data modeling complexity
🔍 Improves accuracy and relevance of AI outputs
The ability to automatically understand relationships between business objects (like connecting purchase orders, invoices, and customers) marks a significant advancement in building intelligent applications! 💡

What are your thoughts on SAP's knowledge graphs? How do you see this impacting the enterprise AI landscape?

#EnterpriseAI #SAP #KnowledgeGraphs #AI #DataInnovation #DigitalTransformation

Who has a KG?


European
Commission


Roche


SAP


LinkedIn


Google


ontotext


SEMANTIC
WEB COMPANY


aws


Stardog


SIEMENS


SANOFI


THE
BRITISH
MUSEUM


MPIWG
Max Planck Institute
for the History of Science


poolparty


BASF
We create chemistry

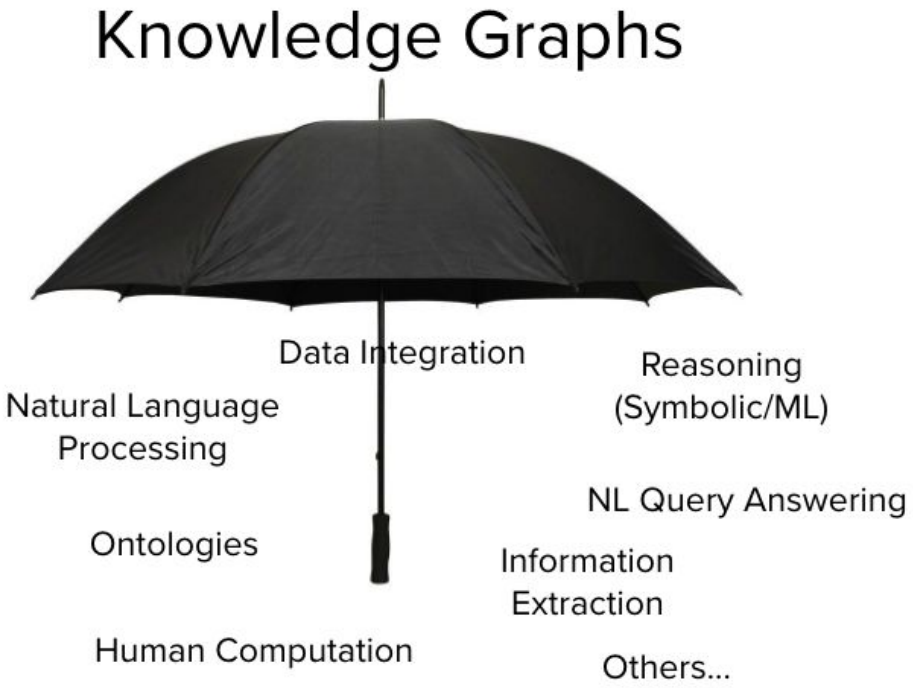

orange™


ebay


Telefónica


Uber

What is a Knowledge Graph?



Extracted from Mariano Rodriguez Muro from KGB2019 Keynote ESWC2019

What is a Knowledge Graph?

"we define a knowledge graph as a **graph of data intended to accumulate and convey knowledge of the real world**, whose **nodes represent entities** of interest and whose **edges represent potentially different relations between these entities**"

Hogan, A., Blomqvist, E., Cochez, M., d'Amato, C., Melo, G. D., Gutierrez, C., ... & Zimmermann, A. (2021). Knowledge graphs. *ACM Computing Surveys (Csur)*, *54*(4), 1-37.

What is a Knowledge Graph?

Knowledge: something that is known and can be written down.

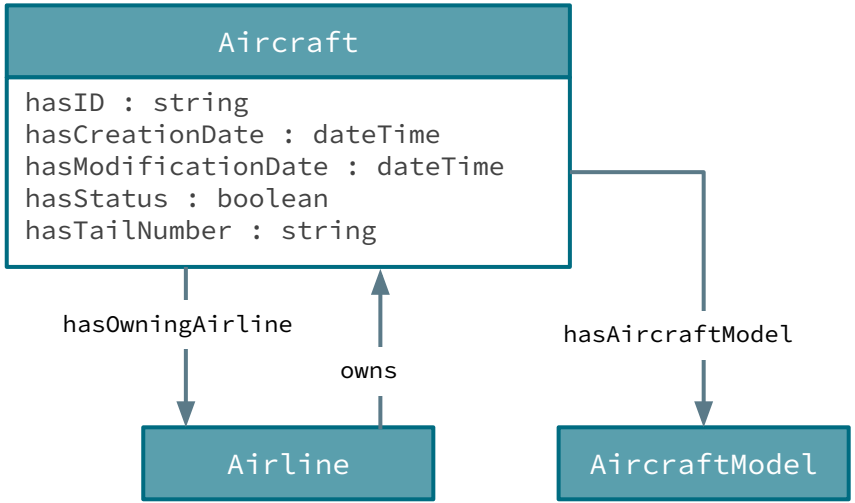
Knowledge may be **composed of simple or quantified statements**

For quantified statements, a more expressive way to represent knowledge **—such as ontologies** or rules— is required

Hogan, A., Blomqvist, E., Cochez, M., d'Amato, C., Melo, G. D., Gutierrez, C., ... & Zimmermann, A. (2021). Knowledge graphs. *ACM Computing Surveys (Csur)*, *54*(4), 1-37.

TBox & ABox

Tbox - Ontology

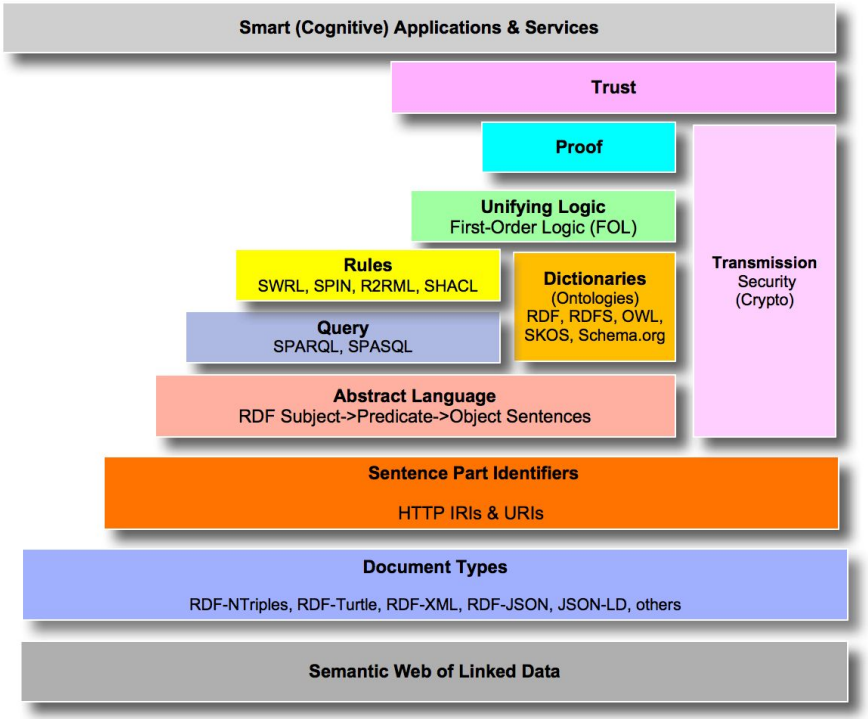


ABox - Data Graph

IB_X123 is a Aircraft
IB_X123 hasID “IBX123”
IB_X123 hasCreationDate 28/12/2021
…
…
IB_X123 hasAircraftModel X123
X123 is a AircraftModel
Iberia owns IB_X123
Iberia is a Airline

ABox statements must be TBox-compliant:
they are assertions that use the vocabulary defined by the TBox.

The Semantic Web stack



Ontologies → OWL

Data → RDF

Queries → SPARQL

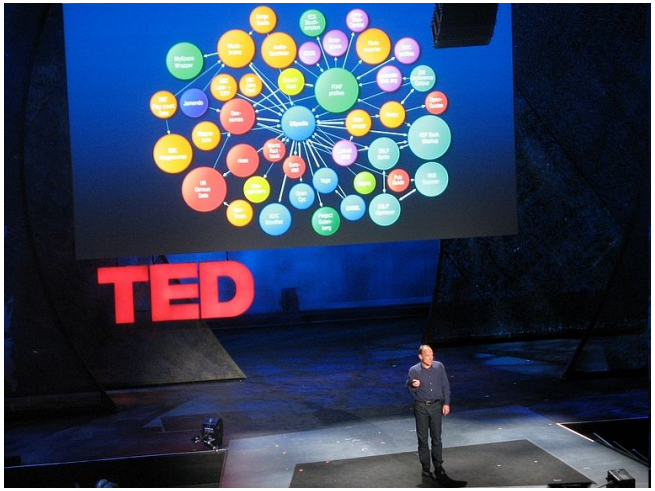
Validation -> SHACL

Construction → [R2]RML

Semantic Web: RDF, OWL, etc.

http://www.w3.org/DesignIssues/LinkedData.html

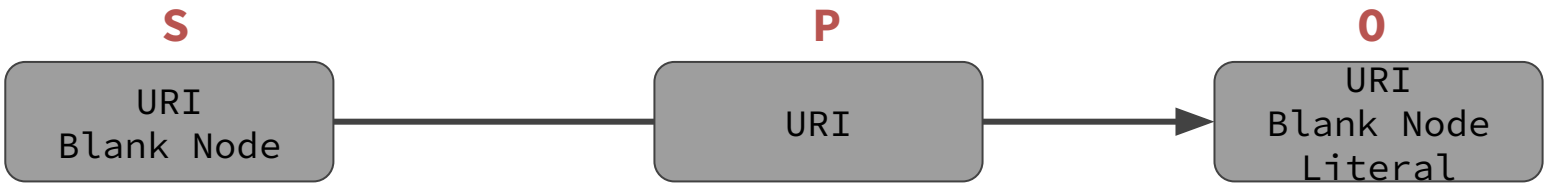
- Use URIs as names for things
- Use HTTP URIs so that people can look up those names.
- When someone looks up a URI, provide useful information using standards (RDF*, SPARQL)
- Include links to other URIs, so that they can discover more things.



https://youtu.be/OM6XIICm_qo

Semantic Web: RDF

RDF triple is composed by (Subject Predicate Object), where:



http://example.org/IB_X123 http://example.org/hasID “IBX123”

RDF graph is a **set of RDF triples.**

An RDF graph can be identified by an URI (RDF Quads)

A **set of RDF graphs is an RDF Dataset**

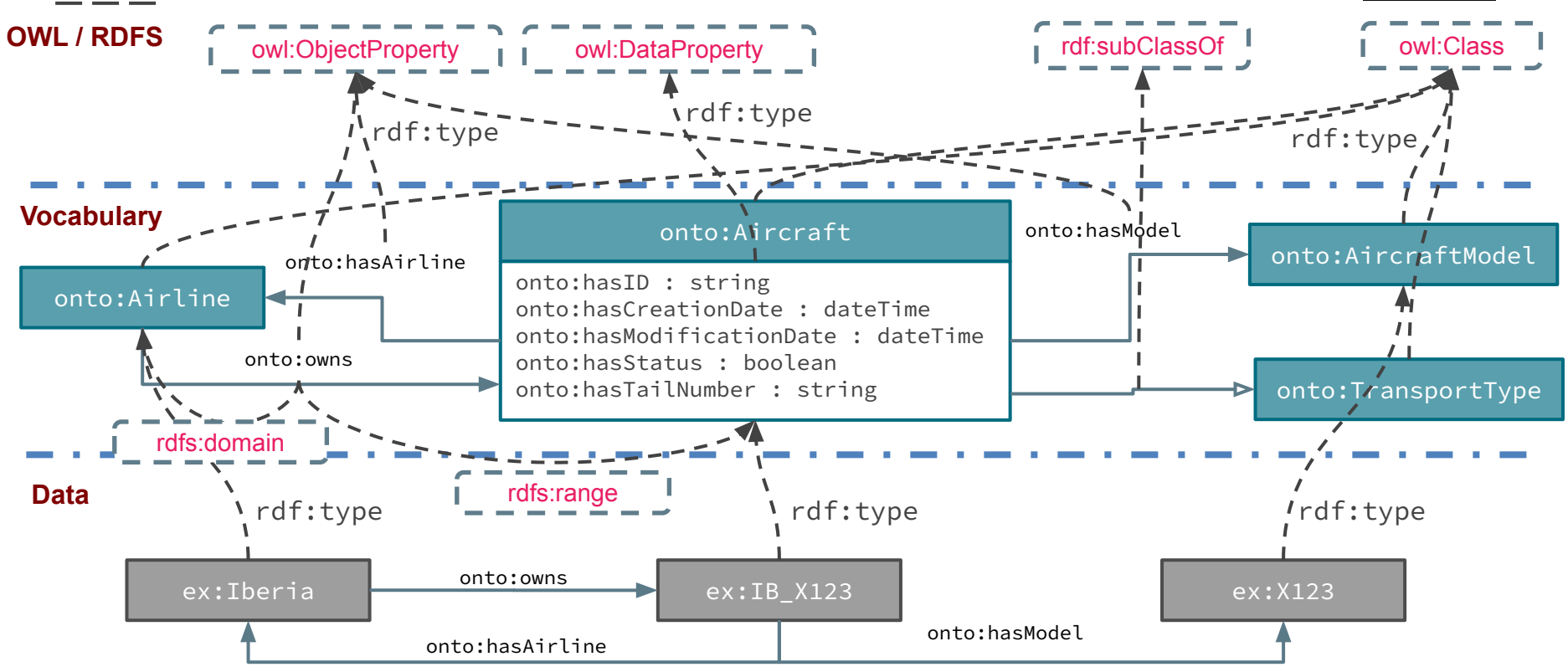
Literals may have an XSD datatype associated

Semantic Web: RDF

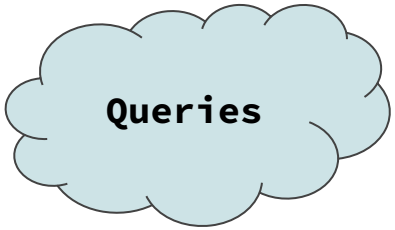
IB_X123 is a Aircraft
IB_X123 hasID “IBX123”
IB_X123 hasCreationDate 28/12/2021
IB_X123 hasAircraftModel X123
X123 is a AircraftModel
Iberia owns IB_X123
Iberia is a Airline

```
@prefix ex: <http://example-ecai.org/resource/>
@prefix onto: <http://example-ecai.org/ontology/>
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
@prefix xsd: <http://www.w3.org/2001/XMLSchema#>

ex:IB_X123     rdf:type              onto:Aircraft
ex:IB_X123     onto:hasID            “IBX123”^^xsd:string
ex:IB_X123     onto:hasCreationDate  “2021-12-28”^^xsd:date
ex:IB_X123     onto:hasAircraftModel ex:X123
ex:X123        rdf:type              onto:AircraftModel
ex:Iberia      onto:owns             ex:IB_X123
ex:Iberia      rdf:type              onto:Airline
```

Semantic Web: Ontologies - OWL



Semantic Web: SPARQL & SHACL



SPARQL 1.1 Overview

W3C Recommendation 21 March 2013

This version:
http://www.w3.org/TR/2013/REC-sparql11-overview-20130321/
Latest version:
http://www.w3.org/TR/sparql11-overview/
Previous version:
http://www.w3.org/TR/2012/PR-sparql11-overview-20121108/
Editor:
The W3C SPARQL Working Group, see Acknowledgements <public-rdf-dawg-comments@w3.

Please refer to the **errata** for this document, which may include some normative corrections.

See also **translations**.

Shapes Constraint Language (SHACL)

W3C Recommendation 20 July 2017

This version:
https://www.w3.org/TR/2017/REC-shacl-20170720/
Latest published version:
https://www.w3.org/TR/shacl/
Latest editor's draft:
https://w3c.github.io/data-shapes/shacl/
Implementation report:
https://w3c.github.io/data-shapes/data-shapes-test-suite/
Previous version:
https://www.w3.org/TR/2017/PR-shacl-20170608/
Editors:
Holger Knublauch, TopQuadrant, Inc.
Dimitris Kontokostas, University of Leipzig
Repository:
GitHub
Issues
Test Suite:
SHACL Test Suite

Please check the **errata** for any errors or issues reported since publication.

See also **translations**.

Semantic Web: Construction with R2RML

W3C®

R2RML: RDB to RDF Mapping Language

W3C Recommendation 27 September 2012

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

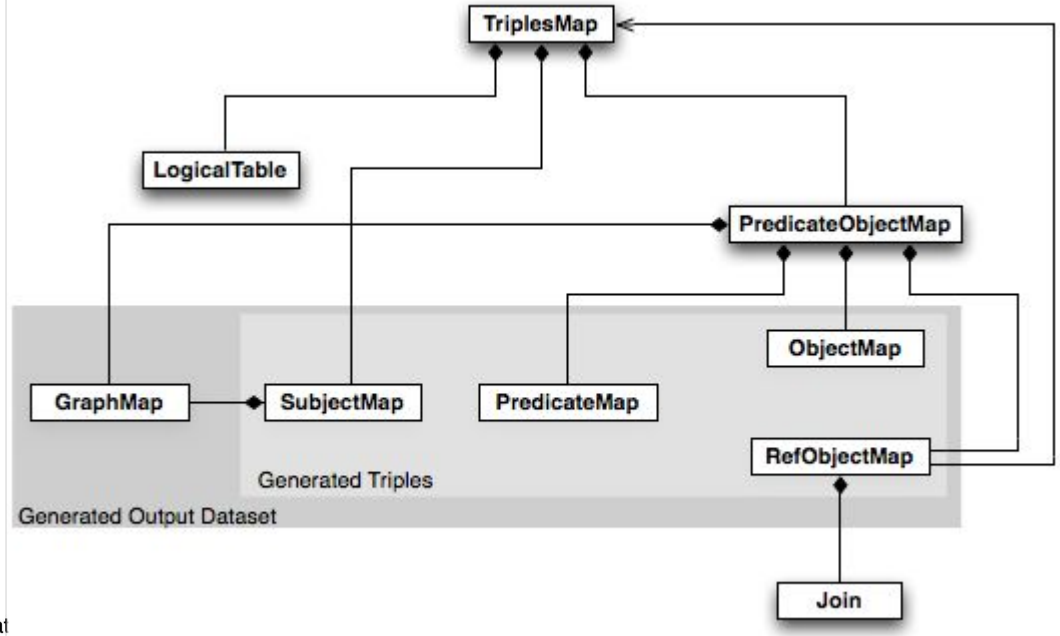
Latest version:
http://www.w3.org/TR/r2rml/

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

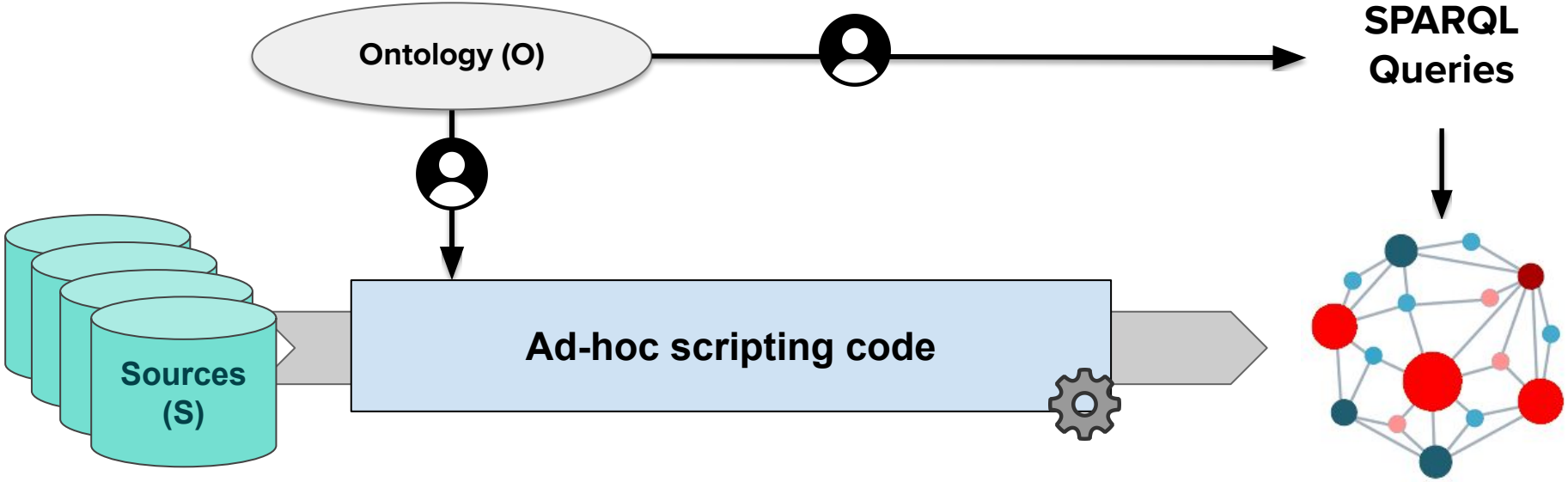
Editors:
Souripriya Das, Oracle
Seema Sundara, Oracle
Richard Cyganiak, DERI, National University of Ireland, Galway

Please refer to the **errata** for this document, which may include some normat

See also **translations**.

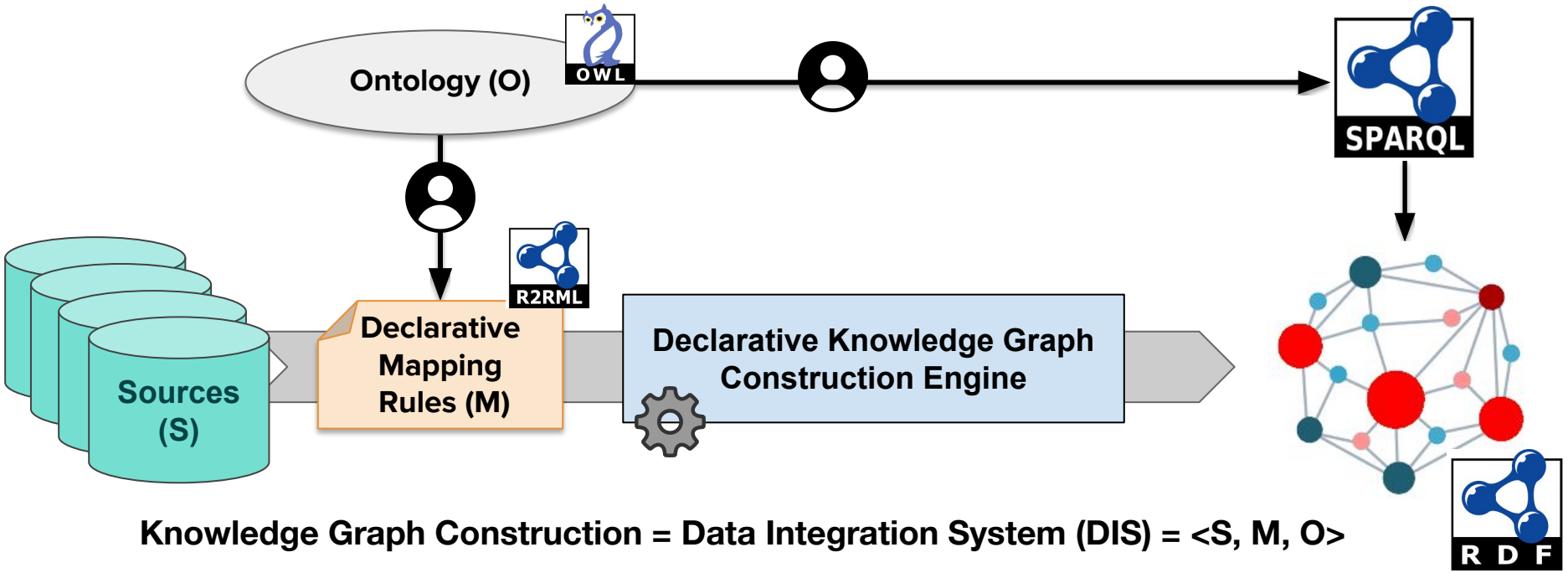


Knowledge Graph Construction: Script based



Efficient	Scalable	Maintainable	Robust	Reproducible
🤷🏽	🤷🏽	🙅	🙅	🙅

Knowledge Graph Construction: Materialization

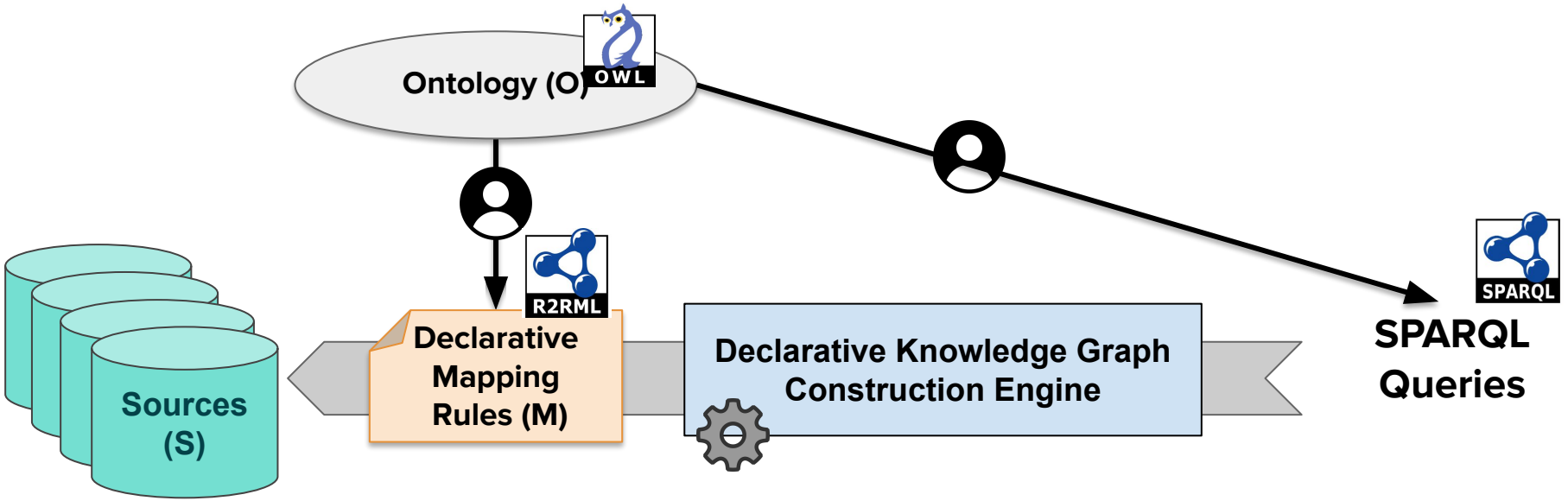


Knowledge Graph Construction = Data Integration System (DIS) = <S, M, O>

Poggi, A., Lembo, D., Calvanese, D., De Giacomo, G., Lenzerini, M., & Rosati, R. (2008). Linking data to ontologies. In *Journal on data semantics X*

Lenzerini, M. Data integration: A theoretical perspective. In *Proceedings of the 21st ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*

Knowledge Graph Construction: Virtualization



Knowledge Graph Construction = Data Integration System (DIS) = <S, M, O>

Poggi, A., Lembo, D., Calvanese, D., De Giacomo, G., Lenzerini, M., & Rosati, R. (2008). Linking data to ontologies. In *Journal on data semantics X*
Lenzerini, M. Data integration: A theoretical perspective. In *Proceedings of the 21st ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*

Mapping Languages for KG Construction

14:30 - 15:30

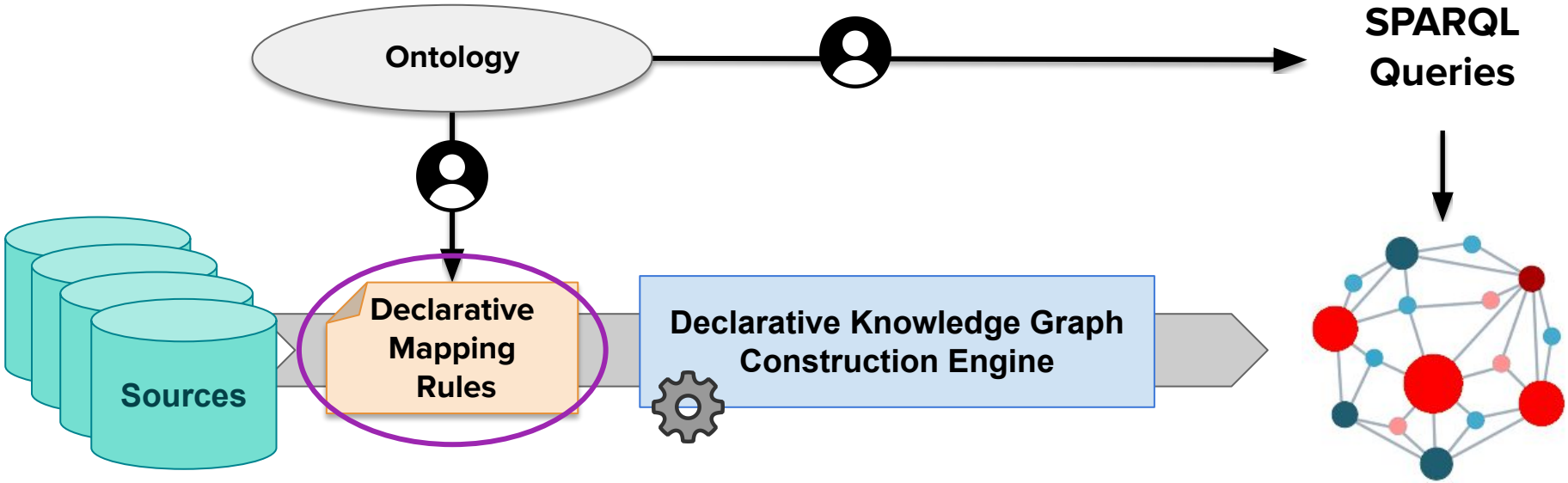
Mapping languages and evolution

RML: The RDF Mapping Language

Who is using RML

- Orange
- EU Comission
- Bosch
- Google
- BASF
-

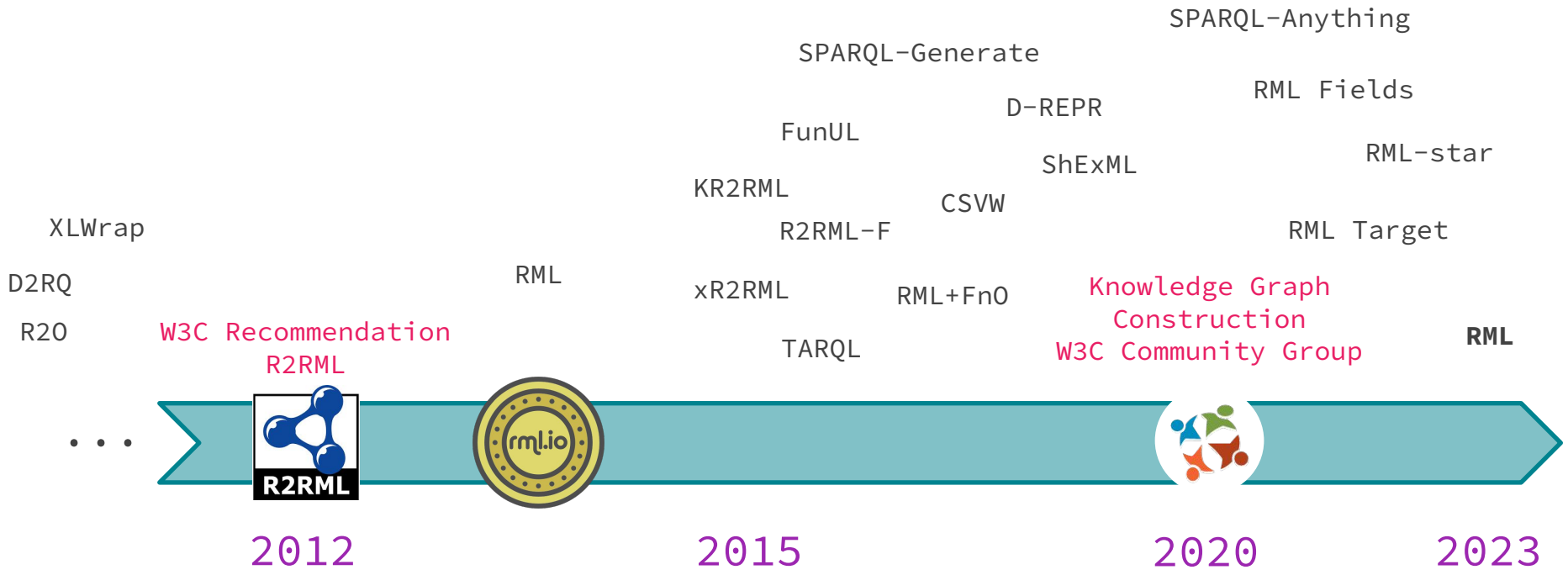
Mapping rules



Mapping languages

- Allow specifying **transformation rules** from heterogeneous data sources into RDF graphs
  - **Data source** and **access** description
  - **Triple** generation:
    - Subject, Predicate, Object
    - Language tags and datatypes
  - Additional:
    - Join conditions
    - Data transformation functions
- Many different languages proposed throughout the years
  - Different **features**
  - Variety of **implementations**

The history so far



Mapping example
–
RML

Data sources

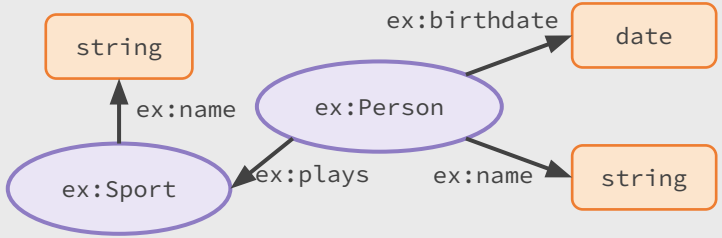
people.csv

ID	Name	Age	SportID
1	Emily	23	2
2	Jonah	22	2

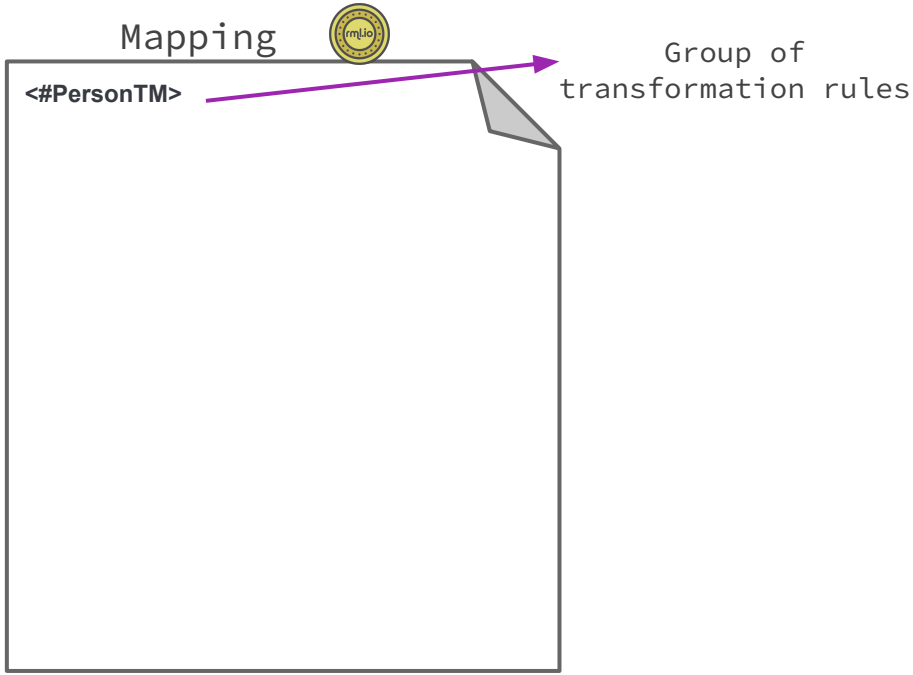
sports.csv

ID	Sport
1	Ice Skating
2	Rugby

Ontology



Mapping



Group of transformation rules

Output Knowledge Graph

Data sources

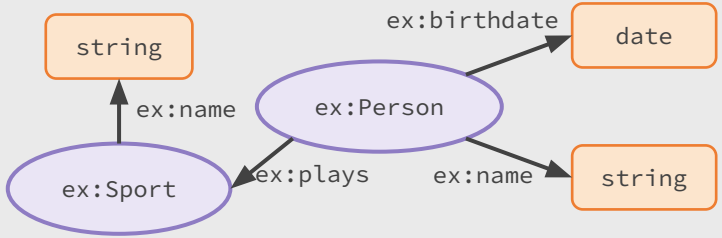
people.csv

ID	Name	Age	SportID
1	Emily	23	2
2	Jonah	22	2

sports.csv

ID	Sport
1	Ice Skating
2	Rugby

Ontology



Mapping

<#PersonTM>
rml:logicalSource [
rml:source “people.csv” ;
rml:referenceFormulation ql:CSV];

Output Knowledge Graph

Data sources

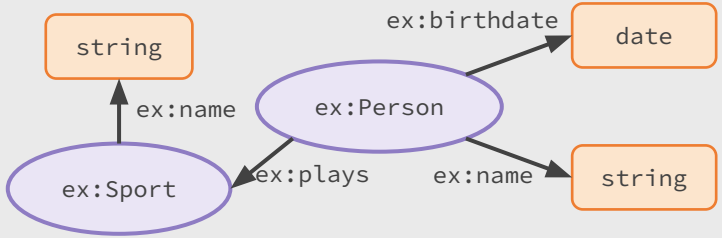
people.csv

ID	Name	Age	SportID
1	Emily	23	2
2	Jonah	22	2

sports.csv

ID	Sport
1	Ice Skating
2	Rugby

Ontology



Mapping

```
<#PersonTM>
rml:logicalSource [
  rml:source “people.csv” ;
  rml:referenceFormulation ql:CSV ];
rml:subjectMap [
  rml:template "Person/{ID}";
  rml:class ex:Person ];
```

Output Knowledge Graph

Data sources

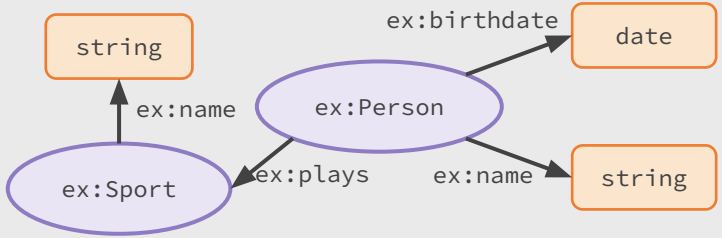
people.csv

ID	Name	Age	SportID
1	Emily	23	2
2	Jonah	22	2

sports.csv

ID	Sport
1	Ice Skating
2	Rugby

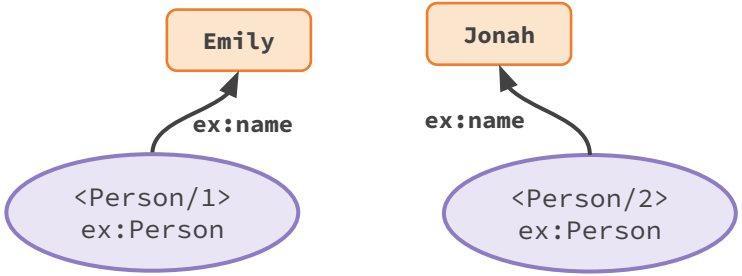
Ontology



Mapping

```
<#PersonTM>
rml:logicalSource [
  rml:source “people.csv” ;
  rml:referenceFormulation ql:CSV ];
rml:subjectMap [
  rml:template "Person/{ID}";
  rml:class ex:Person ];
rml:predicateObjectMap [
  rml:predicate ex:name;
  rml:objectMap [
    rml:reference “Name”] ];
```

Output Knowledge Graph



Data sources

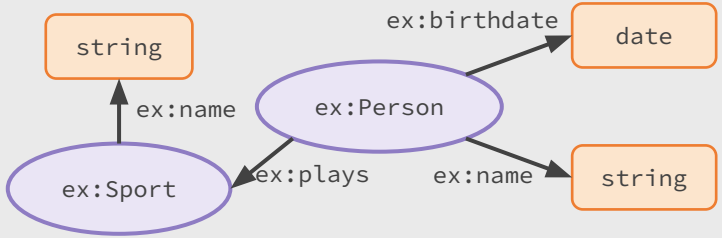
people.csv

ID	Name	Age	SportID
1	Emily	23	2
2	Jonah	22	2

sports.csv

ID	Sport
1	Ice Skating
2	Rugby

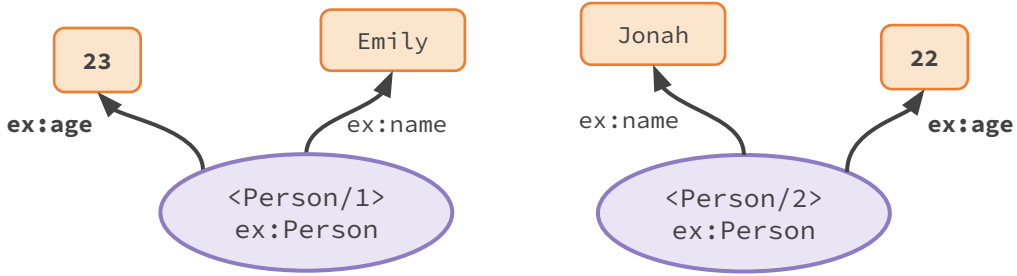
Ontology



Mapping

```
<#PersonTM>
rml:logicalSource [
  rml:source “people.csv” ;
  rml:referenceFormulation ql:CSV ];
rml:subjectMap [
  rml:template "Person/{ID}";
  rml:class ex:Person ];
rml:predicateObjectMap [
  rml:predicate ex:name;
  rml:objectMap [
    rml:reference “Name”] ];
rml:predicateObjectMap [
  rml:predicate ex:age;
  rml:objectMap [
    rml:reference “Age”] ;
...
```

Output Knowledge Graph



Data sources

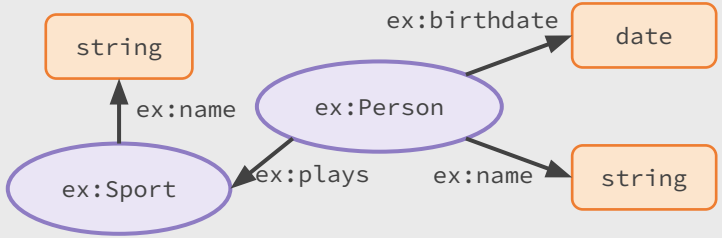
people.csv

ID	Name	Age	SportID
1	Emily	23	2
2	Jonah	22	2

sports.csv

ID	Sport
1	Ice Skating
2	Rugby

Ontology

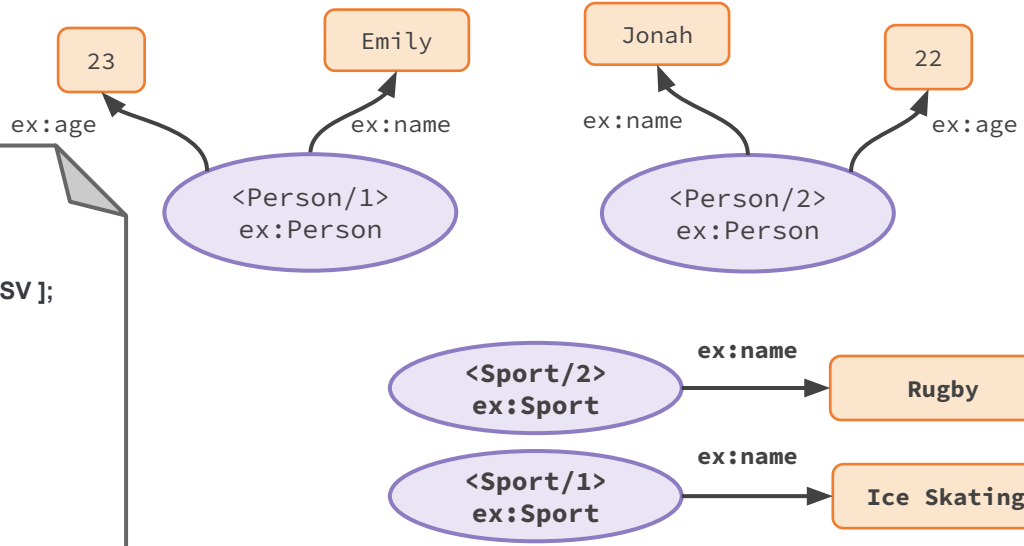


Mapping

```
<#PersonTM>
rml:logicalSource [
  rml:source “people.csv” ;
  rml:referenceFormulation ql:CSV ];
rml:subjectMap [
  rml:template "Person/{ID}";
  rml:class ex:Person ];
rml:predicateObjectMap [
  rml:predicate ex:name;
  rml:objectMap [
    rml:reference ”Name”] ];
rml:predicateObjectMap [
  rml:predicate ex:birthdate;
  rml:objectMap [
    rml:reference “Birthdate”]
...
```

```
<#SportTM>
rml:logicalSource [
  rml:source “sports.csv” ;
  rml:referenceFormulation ql:CSV ];
rml:subjectMap [
  rml:template “Sport/{ID}”;
  rml:class ex:Sport ];
rml:predicateObjectMap [
  rml:predicate ex:name;
  rml:objectMap [
    rml:reference “Sport”] ] .
```

Output Knowledge Graph



Data sources

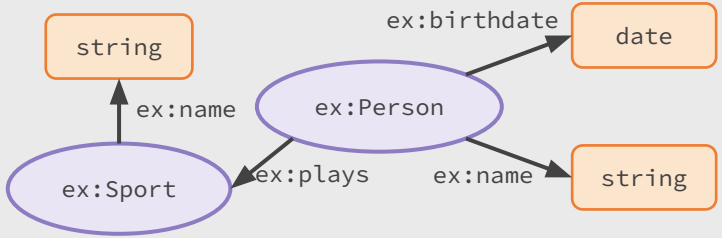
people.csv

ID	Name	Age	SportID
1	Emily	23	2
2	Jonah	22	2

sports.csv

ID	Sport
1	Ice Skating
2	Rugby

Ontology

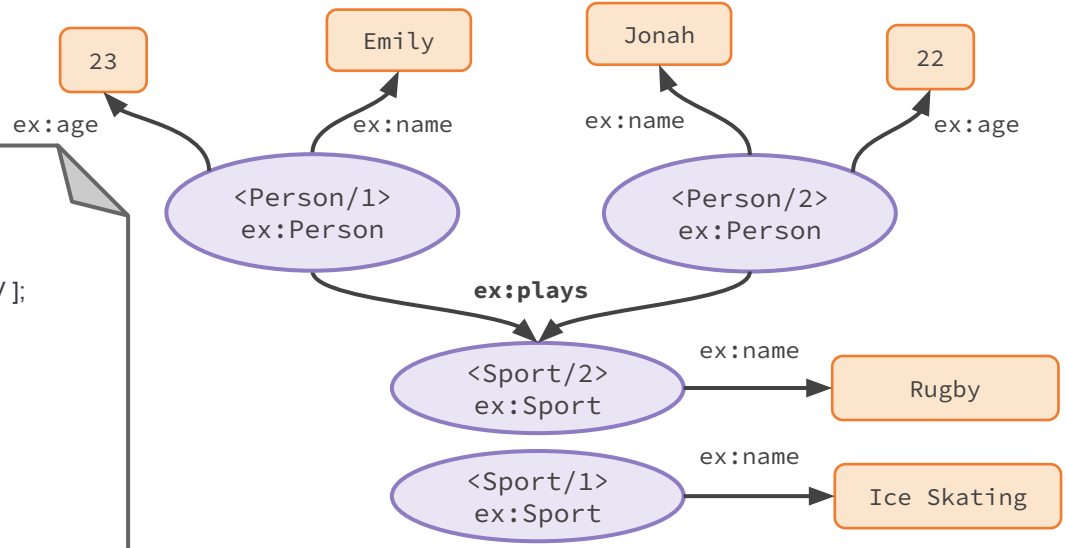


Mapping

```
<#PersonTM>
...
rml:predicateObjectMap [
 rml:predicate ex:plays;
 rml:objectMap [
  rml:parentTriplesMap
        <#SportTM>;
  rml:joinCondition [
   rml:child "SportID";
   rml:parent "ID";
 ]; ]; ].
```

```
<#SportTM>
rml:logicalSource [
  rml:source "sports.csv" ;
  rml:referenceFormulation ql:CSV ];
rml:subjectMap [
  rml:template "Sport/{ID}";
  rml:class ex:Sport ];
rml:predicateObjectMap [
  rml:predicate ex:name;
  rml:objectMap [
    rml:reference "Sport"] ] .
```

Output Knowledge Graph



Data sources

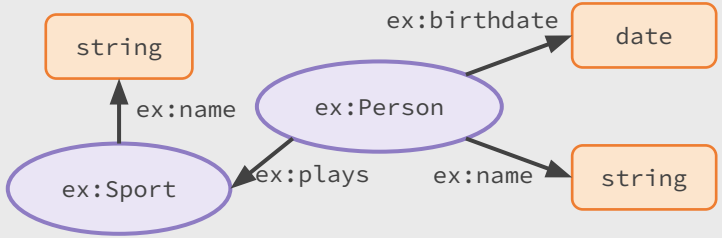
people.csv

ID	Name	Age	SportID
1	Emily	23	2
2	Jonah	22	2

sports.csv

ID	Sport
1	Ice Skating
2	Rugby

Ontology

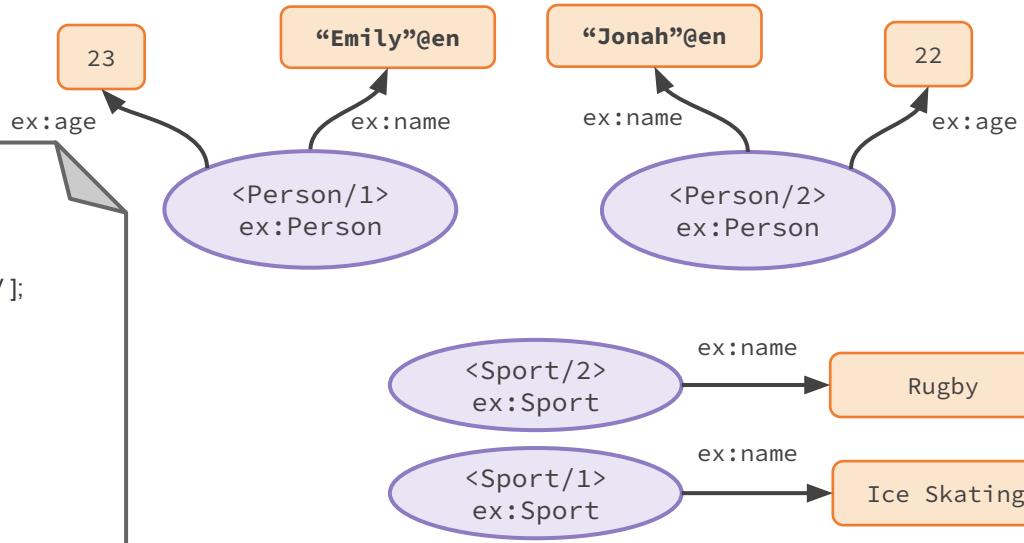


Mapping

```
<#PersonTM>
rml:logicalSource [
  rml:source “people.csv” ;
  rml:referenceFormulation ql:CSV ];
rml:subjectMap [
  rml:template "Person/{ID}";
  rml:class ex:Person ];
rml:predicateObjectMap [
  rml:predicate ex:name;
  rml:objectMap [
    rml:reference ”Name”; rml:language “en”] ];
rml:predicateObjectMap [
  rml:predicate ex:birthdate;
  rml:objectMap [
    rml:reference “Birthdate”]
...
```

```
urce [
sports.csv” ;
eFormulation ql:CSV ];
ap [
“Sport/{ID}”;
:Sport ];
ObjectMap [
e ex:name;
ap [
ce “Sport”] ] .
```

Output Knowledge Graph



Data sources

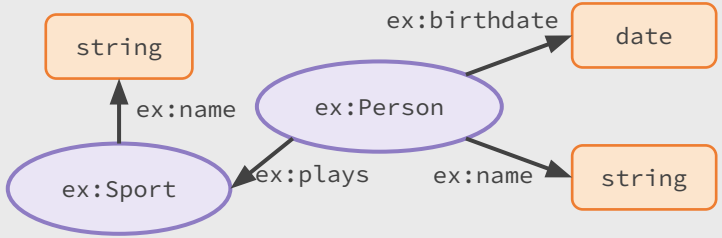
people.csv

ID	Name	Age	SportID
1	Emily	23	2
2	Jonah	22	2

sports.csv

ID	Sport
1	Ice Skating
2	Rugby

Ontology

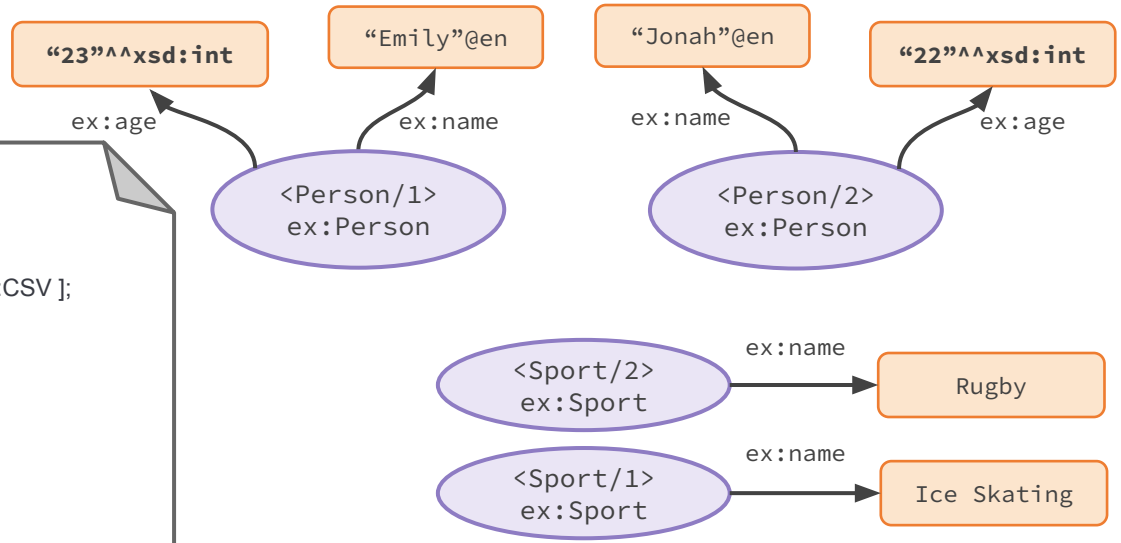


Mapping

```
<#PersonTM>
rml:logicalSource [
  rml:source “people.csv” ;
  rml:referenceFormulation ql:CSV ];
rml:subjectMap [
  rml:template "Person/{ID}";
  rml:class ex:Person ];
rml:predicateObjectMap [
  rml:predicate ex:name;
  rml:objectMap [
    rml:reference ”Name”; rml:language “en”] ];
rml:predicateObjectMap [
  rml:predicate ex:birthdate;
  rml:objectMap [
    rml:reference “Birthdate”; rml:datatype xsd:int]
...
```

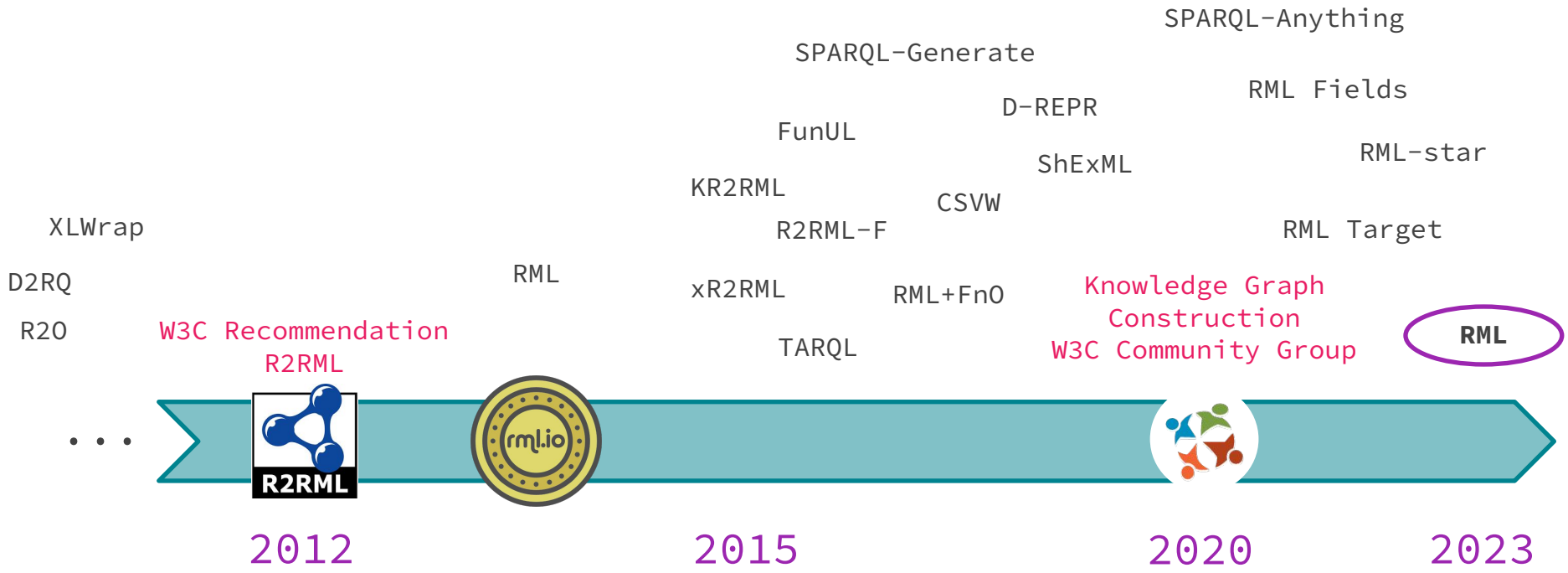
```
urce [
sports.csv” ;
eFormulation ql:CSV ];
ap [
e “Sport/{ID}”;
:Sport ];
ObjectMap [
e ex:name;
ap [
ce “Sport”] ] .
```

Output Knowledge Graph



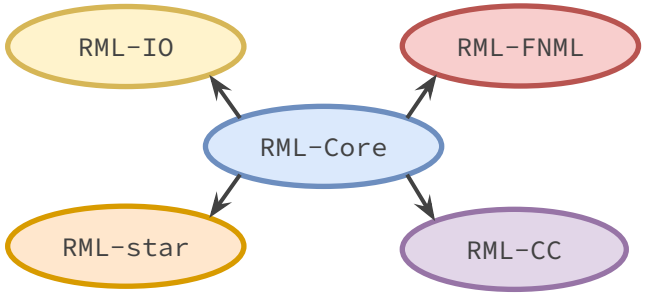
RML - 2023 revision

The history so far

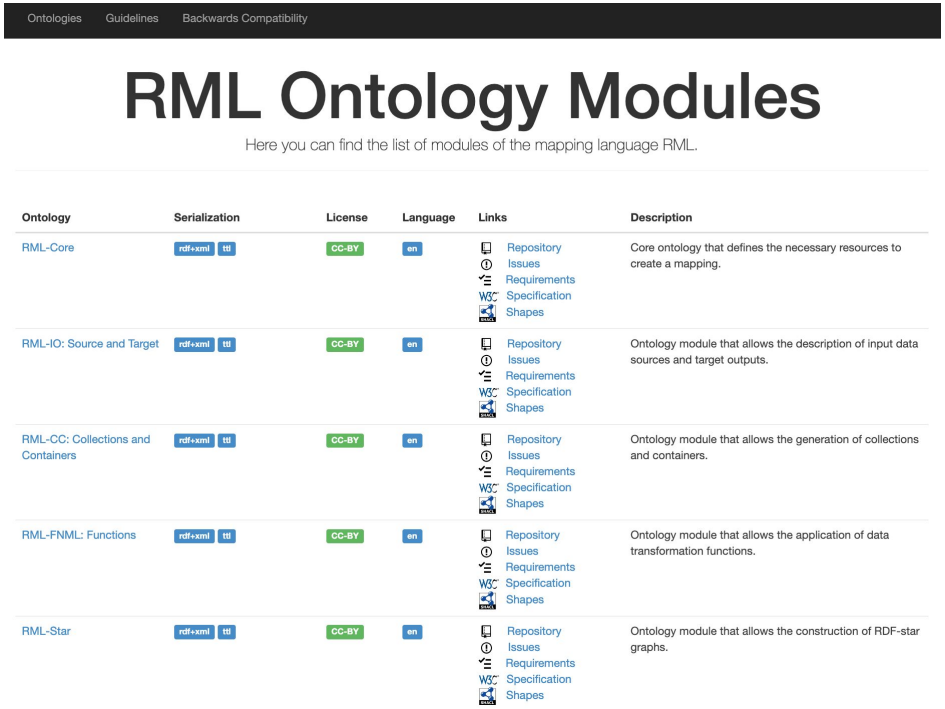


The RML Ontology: Overview

- **Reference** ontology specification addressing **identified challenges**
- **Modular** design
- Developed following the **Linked Open Terms (LOT) methodology**
- Maintaining **backwards compatibility** with R2RML
  - **Semantic correspondences** with previous resources
- All resources in all modules use **same namespace and prefix**
  - **rml:** https://w3id.org/rml/



The RML Ontology: Modules



Ontologies Guidelines Backwards Compatibility

RML Ontology Modules

Here you can find the list of modules of the mapping language RML.

Ontology	Serialization	License	Language	Links	Description
RML-Core	rdf+xml ttl	CC-BY	en	Repository Issues Requirements Specification Shapes	Core ontology that defines the necessary resources to create a mapping.
RML-IO: Source and Target	rdf+xml ttl	CC-BY	en	Repository Issues Requirements Specification Shapes	Ontology module that allows the description of input data sources and target outputs.
RML-CC: Collections and Containers	rdf+xml ttl	CC-BY	en	Repository Issues Requirements Specification Shapes	Ontology module that allows the generation of collections and containers.
RML-FNML: Functions	rdf+xml ttl	CC-BY	en	Repository Issues Requirements Specification Shapes	Ontology module that allows the application of data transformation functions.
RML-Star	rdf+xml ttl	CC-BY	en	Repository Issues Requirements Specification Shapes	Ontology module that allows the construction of RDF-star graphs.

Composed of **5 modules:**

- **RML-Core:** schema transformations
- **RML-IO:** source and target data
- **RML-FNML:** data transformation functions
- **RML-CC:** collections and containers
- **RML-star:** RDF-star

w3id.org/rml/portal

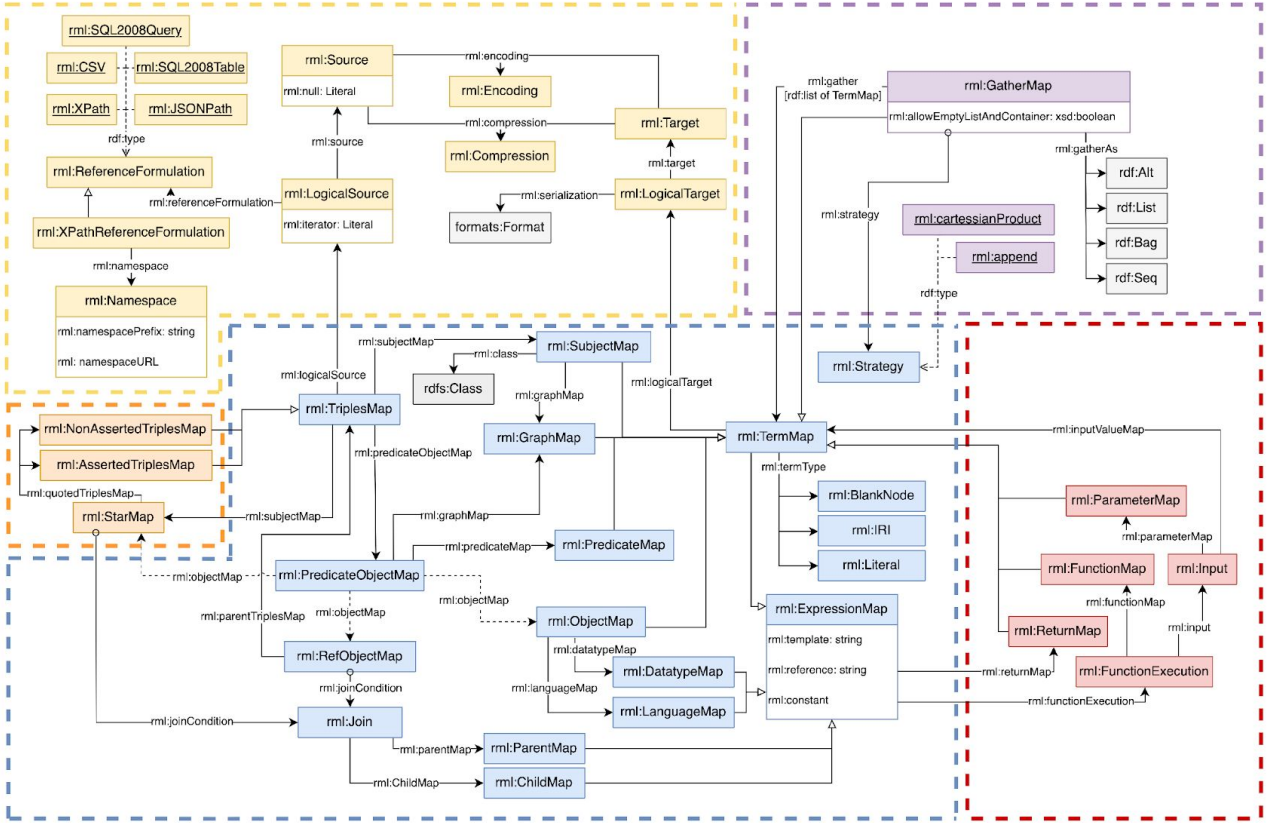
Network of Resources

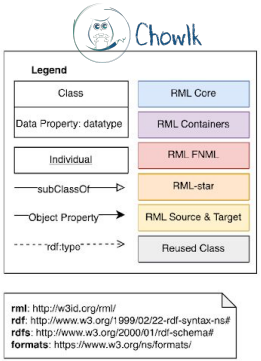
Each module is composed of:

- **Ontology** w3id.org/rml/core w3id.org/rml/io w3id.org/rml/fnml w3id.org/rml/cc w3id.org/rml/star
  - Implemented in OWL2: Definition of concepts, relationships (w/ domain and range)
  - Available in multiple serializations and a human readable documentation
- **Specification** w3id.org/rml/core/spec w3id.org/rml/io/spec w3id.org/rml/fnml/spec w3id.org/rml/cc/spec w3id.org/rml/star/spec
  - Feature details, examples of use and implementation instructions
- **SHACL shapes** w3id.org/rml/core/shapes w3id.org/rml/io/shapes w3id.org/rml/fnml/shapes w3id.org/rml/cc/shapes w3id.org/rml/star/shapes
  - Restrictions of module constructs
  - Usable to test correctness of mappings
- **Test cases (WIP)**
  - Language coverage by processing engines

w3id.org/rml/portal

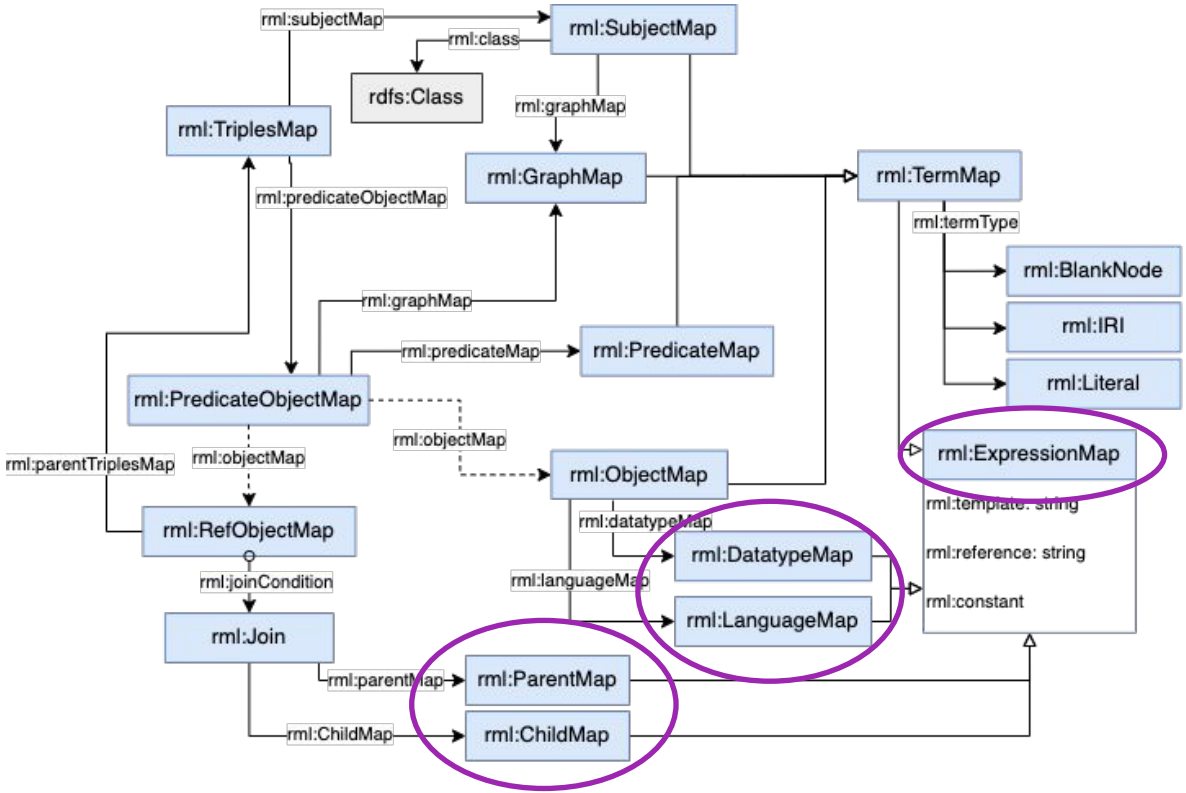
The RML Ontology: Overview





w3id.org/rml/portal

RML-Core: Schema transformations



- Maintains **R2RML** basic structure
- **Dynamic** generation of:
  - Language tags
  - Data types
- Increased **flexibility** for join conditions

w3id.org/rml/core

w3id.org/rml/core/spec

w3id.org/rml/core/shapes

RML-Core: Example

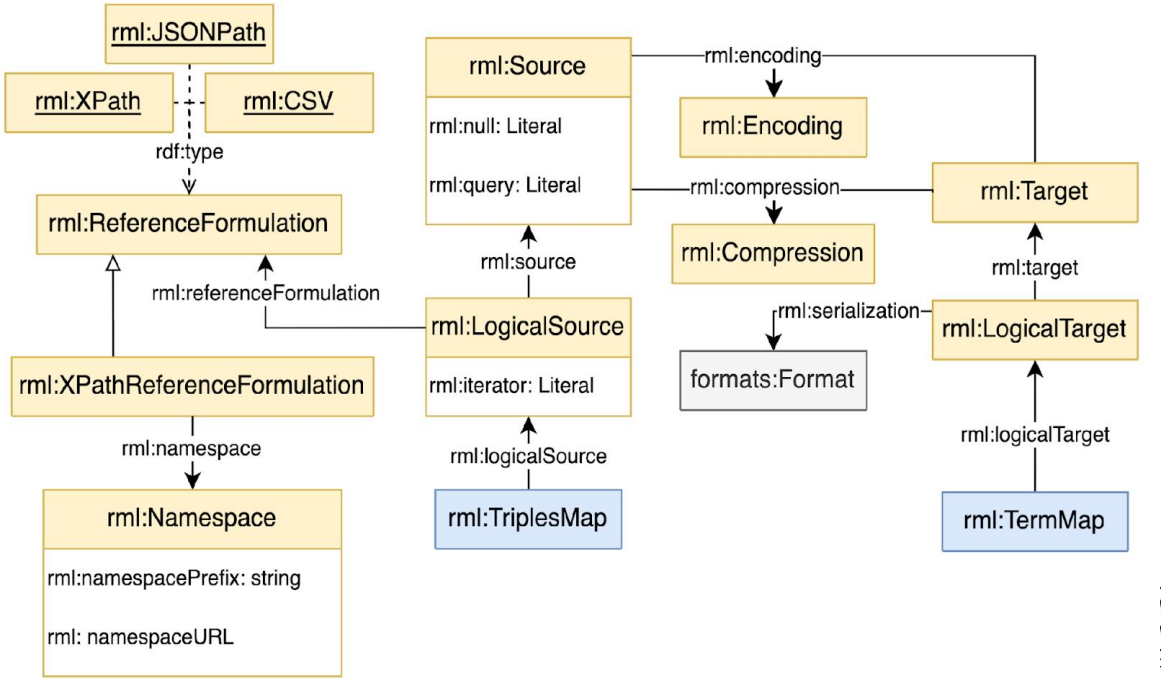
```
<#NameTriplesMap> a rml:TriplesMap ;
  rml:logicalSource <#JSONSource> ;
  rml:subjectMap [
    a rml:SubjectMap, rml:ExpressionMap ;
    rml:template "{$.NAME}" ;
    rml:class ex:Athlete ;
    ] ;
  rml:predicateObjectMap [
    a rml:PredicateObjectMap ;
    rml:predicate ex:name ;
    rml:objectMap [
      a rml:ObjectMap, rml:ExpressionMap ;
      rml:reference "$.NAME" ;
      rml:languageMap [
        a rml:LanguageMap;
        rml:reference "$.COUNTRY"]
     ] ] .
```

{JSON}

```
[ { "NAME": "Duplantis",
    "RANK": "1",
    "MARK": "6.22",
    "COUNTRY": "sv"
  },{
    "NAME": "Guttormsen",
    "RANK": "2",
    "MARK": "6.00",
    "COUNTRY": "no" } ]
```

```
:Duplantis a ex:Athlete ;
     ex:name "Duplantis"@sv .
:Guttormsen a ex:Athlete ;
     ex:name "Guttormsen"@no .
```

RML-IO: Data source and target



- **Extended input** data source description
- **Output data** description
- Leverage of existing **vocabularies** (SCAT, SPARQL-SD, VoID)

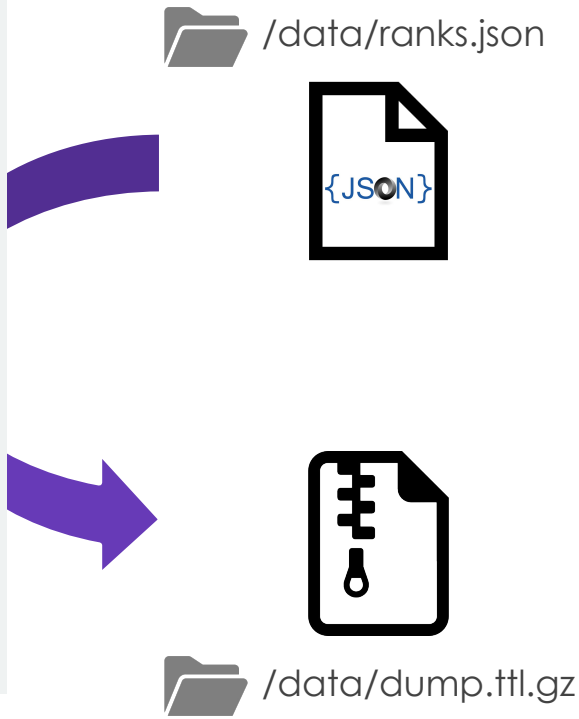
w3id.org/rml/io

w3id.org/rml/io/spec

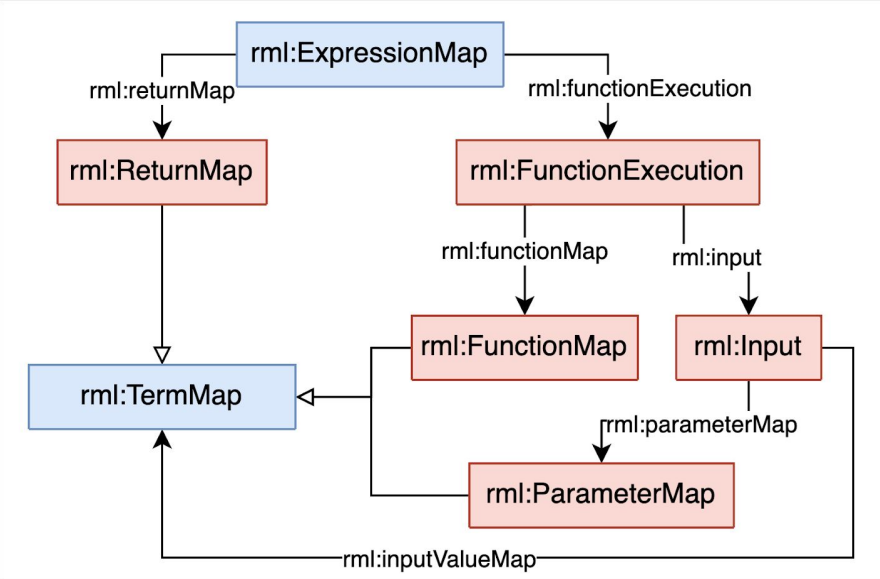
w3id.org/rml/io/shapes

RML-IO: Example

```
<#JSONSource> a rml:LogicalSource ;
  rml:source [ a rml:Source, dcat:Distribution ;
    dcat:accessURL <file:///data/ranks.json> ];
  rml:referenceFormulation rml:JSONPath ;
  rml:iterator "$.[*]"; .

<#FileTarget> a rml:LogicalTarget ;
  rml:target [ a rml:Target, void:Dataset ;
    void:dataDump <file:///data/dump.ttl.gz> ;
    rml:compression rml:gzip ;
    rml:encoding rml:UTF-8 ] ;
  rml:serialization formats:Turtle ; .

<#RankTriplesMap> a rml:TriplesMap ;
  rml:logicalSource <#JSONSource> ;
  rml:subjectMap <#RankSubjectMap> .

<#RankSubjectMap> rml:logicalTarget <#FileTarget>.
```



RML-FNML: Data Transformations



- Refines **RML+FnO** approach
- Reference connector between RML and the **Function Ontology** (FnO)

w3id.org/rml/fnml

w3id.org/rml/fnml/spec

w3id.org/rml/fnml/shapes

RML-FNML: Example

```
<#DateTriplesMap> a rml:TriplesMap ;
  rml:logicalSource <#JSONSource> ;
  rml:subjectMap  <#RankSubjectMap> ;
  rml:predicateObjectMap [
    rml:predicate ex:jumpOnDate ;
    rml:objectMap [
      rml:functionExecution <#Execution> ;
      rml:return ex:dateOut ] ] .

<#Execution> a rml:FunctionExecution ;
  rml:function ex:parseDate ;
  rml:input
     [ a rml:Input ;
    rml:parameter ex:valueParam;
    rml:inputValueMap [
       rml:reference "$.DATE" ] ] ,
    [ a rml:Input ;
    rml:parameter ex:dateFormatParam ;
    rml:inputValueMap [
       rml:constant ”DD-MM-YYYY" ] ] .
```

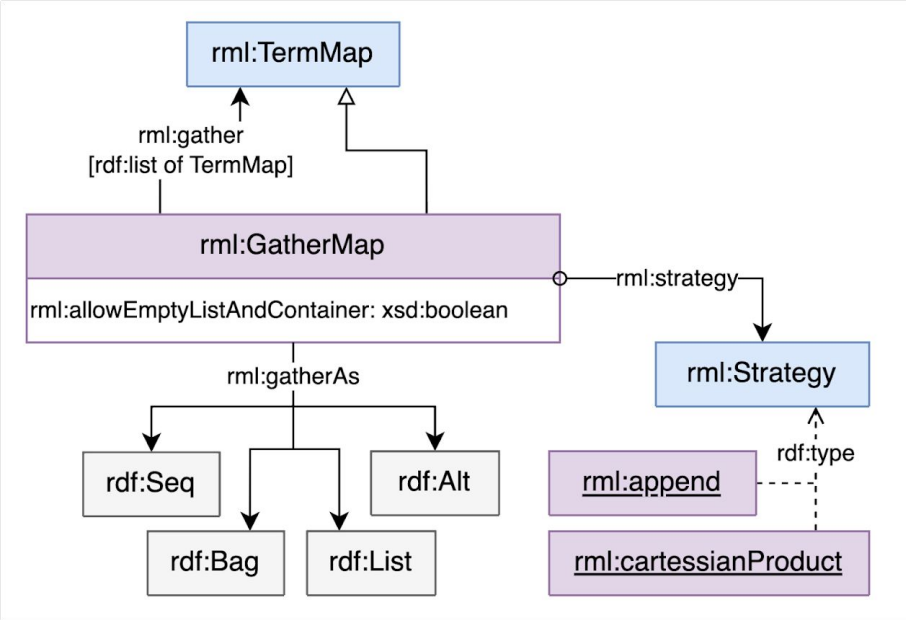
{JSON}

```
[ { "NAME": "Duplantis",
    "RANK": "1",
    "MARK": "6.22",
    "DATE": 02-25-2023
  },{
    "NAME": "Guttormsen",
    "RANK": "2",
    "MARK": "6.00",
    "DATE": 03-10-2023} ]
```

RDF

```
:Duplantis a ex:Athlete ;
  ex:jumpsOnDate “2023-25-02”.
:Guttormsen a ex:Athlete ;
  ex:jumpsOnDate “2023-10-03”.
```

RML-CC: Collections and Containers



- Introduces generation of **collections and containers**
- Specifies how **gather terms** into a CC and **manage** them: to assign them a IRI or BN, manage empty CC, how the gathering is performed...

w3id.org/rml/cc

w3id.org/rml/cc/spec

w3id.org/rml/cc/shapes

RML-CC: Example

```
<#RankingTriplesMap> a rml:TriplesMap ;
  rml:logicalSource <#JSONSource> ;
 rml:subjectMap [
     rml:constant :Ranking23 ];
  rml:predicateObjectMap [
    rml:predicate ex:contains;
    rml:objectMap [
      rml:gather ( [
        rml:template "{$.*.NAME}";
        rml:termType rml:IRI ] );
      rml:gatherAs rdf:List; ] ] .
```

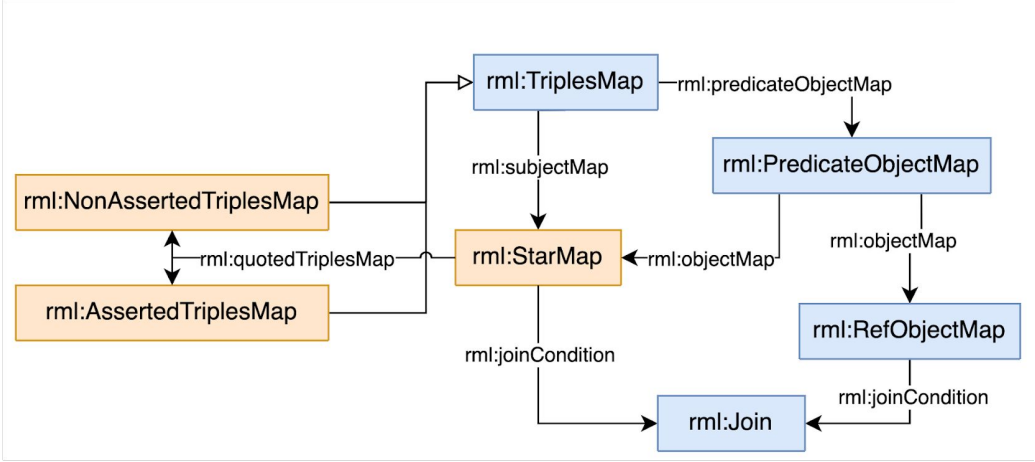
{JSON}

```
[ { "NAME": "Duplantis",
    "RANK": "1",
    "MARK": "6.22",
    "DATE": 02-25-2023
  },{
    "NAME": "Guttormsen",
    "RANK": "2",
    "MARK": "6.00",
    "DATE": 03-10-2023} ]
```

RDF

```
:Ranking23 ex:contains (:Duplantis
:Guttormsen ).
```

RML-star: RDF-star



- **Recursiveness** in mapping rules to generate quoted triples
- Applicable in **subject** and **object** position
- Asserted and non-asserted quoted triples

w3id.org/rml/star

w3id.org/rml/star/spec

w3id.org/rml/star/shapes

RML-star: Example

```
<#QuotedRankTM> a rml:AssertedTriplesMap ;
  rml:logicalSource <#JSONSource> ;
  rml:subjectMap [rml:template "{$.NAME}"];
  rml:predicateObjectMap [
     rml:predicate ex:mark ;
    rml:objectMap [
       rml:reference "$.MARK" ;
] ] .

<#DateTM> a rml:TriplesMap ;
  rml:logicalSource <#JSONSource> ;
  rml:subjectMap [
     rml:quotedTriplesMap <#QuotedRankTM>
];
  rml:predicateObjectMap [
    rml:predicate ex:date ;
    rml:objectMap [
       rml:reference "$.DATE";
] ] .
```

{JSON}

```
[ { "NAME": "Duplantis",
    "RANK": "1",
    "MARK": "6.22",
    "DATE": 02-25-2023
  },{
    "NAME": "Guttormsen",
    "RANK": "2",
    "MARK": "6.00",
    "DATE": 03-10-2023} ]
```

```
:Duplantis ex:mark “6.22” .
<< :Duplantis ex:mark “6.22” >>
     ex:date “02-25-2023” .
:Guttormsen ex:mark “6.00” .
<< :Guttormsen ex:mark “6.00” >>
     ex:date “03-10-2023” .
```

How do you construct a Knowledge Graph?

David Chaves-Fraga and
Ana Iglesias-Molina

Coffee Break!

We will be back at 16:00

Easy KG construction

16:00 - 17:00

YARRRML: User-Friendly Syntax for RML

Hands-on example

User-friendly approaches to write mappings

- [R2]RML mappings are written as Turtle files:
  - Risk of syntax errors
  - Too many language constructs
  - Long process (~6 months)
- Approaches developed to aid the mapping writing:
  - User-friendly syntax:
    - YARRRML ➡️ RML
    - XRM ➡️ R2RML, CSVW, RML (Zazuko)
    - SMS2 ➡️ R2RML (Stardog)
    - obda ➡️ R2RML (Ontop)
  - Editors
    - RMLEditor ➡️ RML
    - Mapeathor ➡️ R2RML, RML, YARRRML
    - …

YARRRML

User-friendly serialization for RML based on YAML syntax:

- Compact syntax
- Widely adopted in real-life projects
  - Entity Reconciliation API on Google Specification
  - https://w3id.org/kg-construct/yarrrml
- Online service
  - https://rml.io/yarrrml/matey/

Mapping example
–
YARRRML

Data sources

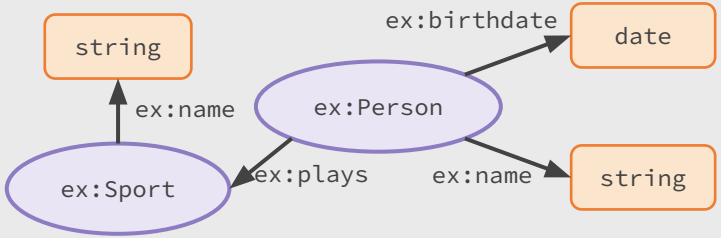
people.csv

ID	Name	Birthdate	SportID
1	Emily	08/02/90	2
2	Jonah	18/11/75	2

sports.csv

ID	Sport
1	Ice Skating
2	Rugby

Ontology



Mapping

```
mappings:
  PERSON:
```

Group of transformation rules

Output Knowledge Graph

Data sources

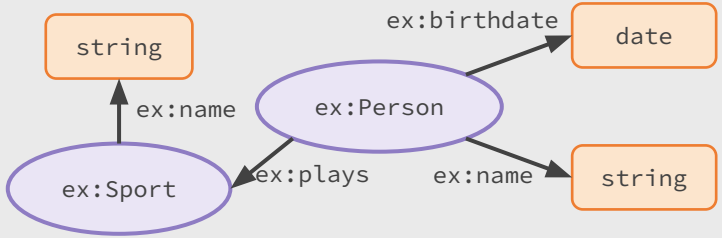
people.csv

ID	Name	Birthdate	SportID
1	Emily	08/02/90	2
2	Jonah	18/11/75	2

sports.csv

ID	Sport
1	Ice Skating
2	Rugby

Ontology



Mapping

```
mappings:
 PERSON:
  sources:
   - ['data/people.csv~csv']
```

Output Knowledge Graph

Data sources

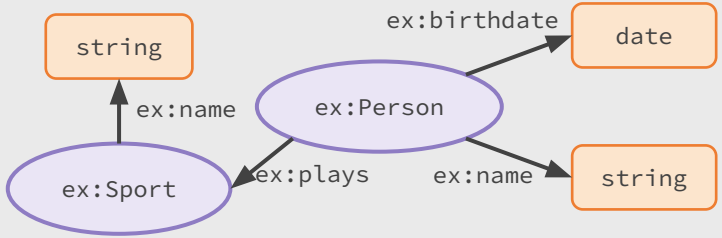
people.csv

ID	Name	Birthdate	SportID
1	Emily	08/02/90	2
2	Jonah	18/11/75	2

sports.csv

ID	Sport
1	Ice Skating
2	Rugby

Ontology



Mapping

```
mappings:
 PERSON:
  sources:
   - ['data/people.csv~csv']
  s: http://ex.com/Person/$(ID)
```

Output Knowledge Graph

Data sources

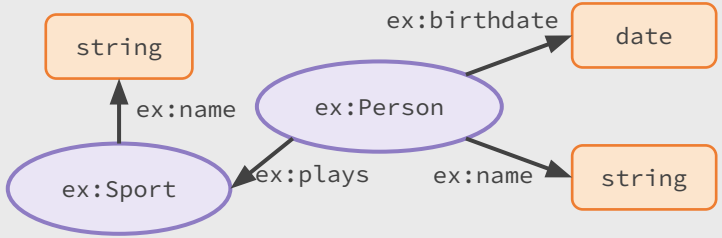
people.csv

ID	Name	Birthdate	SportID
1	Emily	08/02/90	2
2	Jonah	18/11/75	2

sports.csv

ID	Sport
1	Ice Skating
2	Rugby

Ontology



Mapping

```
mappings:
 PERSON:
  sources:
   - ['data/people.csv~csv']
  s: http://ex.com/Person/$(ID)
  po:
   - [a, ex:Person]
```

Output Knowledge Graph

Data sources

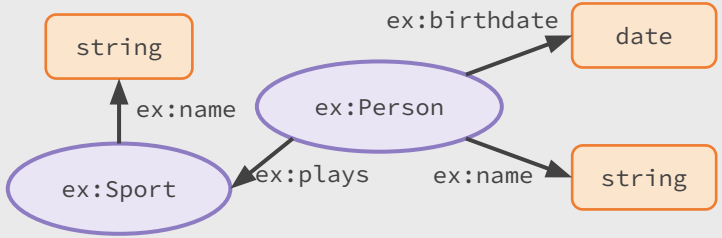
people.csv

ID	Name	Birthdate	SportID
1	Emily	08/02/90	2
2	Jonah	18/11/75	2

sports.csv

ID	Sport
1	Ice Skating
2	Rugby

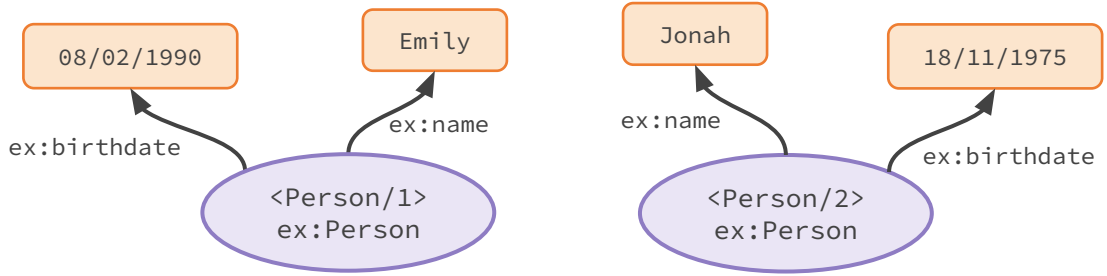
Ontology



Mapping

```
mappings:
 PERSON:
  sources:
   - ['data/people.csv~csv']
  s: http://ex.com/Person/$(ID)
  po:
   - [a, ex:Person]
   - [ex:name, $(Name)]
   - [ex:birthdate, $(Birthdate)]
```

Output Knowledge Graph



Data sources

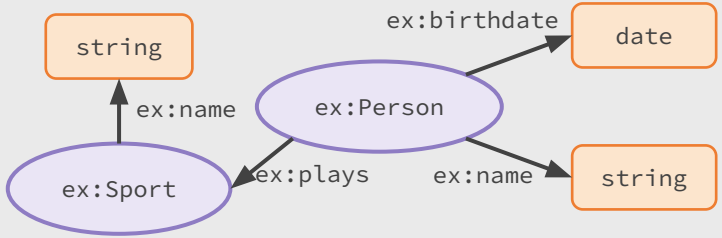
people.csv

ID	Name	Birthdate	SportID
1	Emily	08/02/90	2
2	Jonah	18/11/75	2

sports.csv

ID	Sport
1	Ice Skating
2	Rugby

Ontology

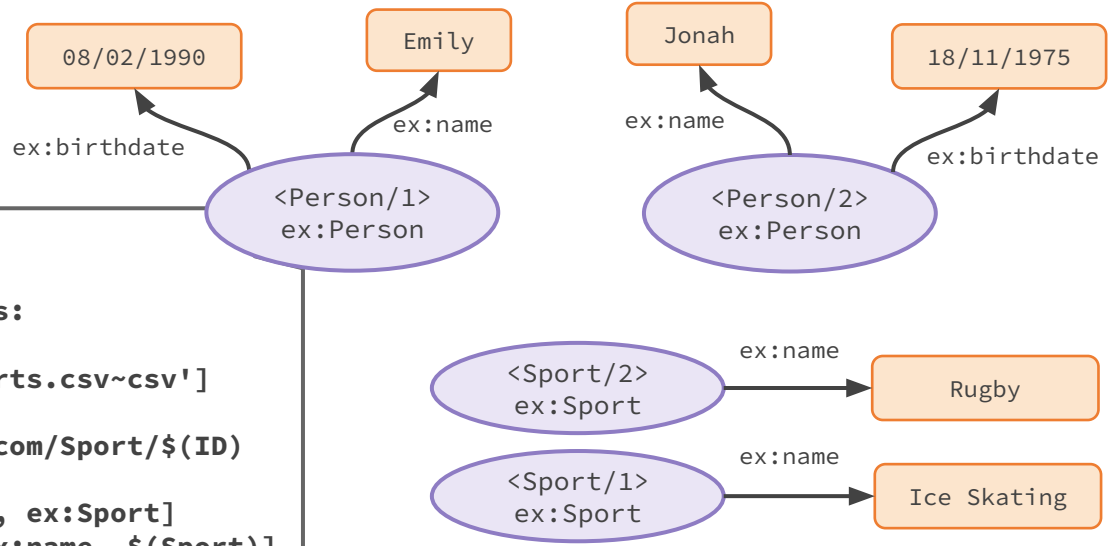


Mapping

```
mappings:
 PERSON:
  sources:
   - ['data/people.csv~csv']
  s: http://ex.com/Person/$(ID)
  po:
   - [a, ex:Person]
   - [ex:name, $(Name)]
   - [ex:birthdate, $(Birthdate)]
```

```
mappings:
 SPORT:
    sources:
      -
['data/sports.csv~csv']
    s:
http://ex.com/Sport/$(ID)
    po:
      - [a, ex:Sport]
      - [ex:name, $(Sport)]
```

Output Knowledge Graph



Data sources

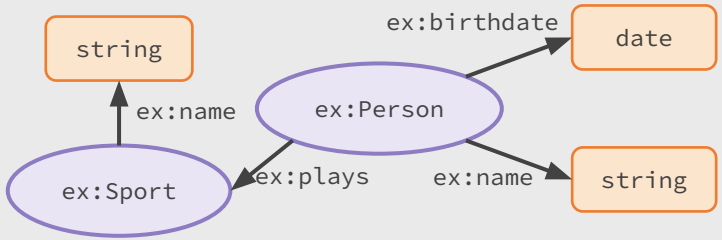
people.csv

ID	Name	Birthdate	SportID
1	Emily	08/02/90	2
2	Jonah	18/11/75	2

sports.csv

ID	Sport
1	Ice Skating
2	Rugby

Ontology

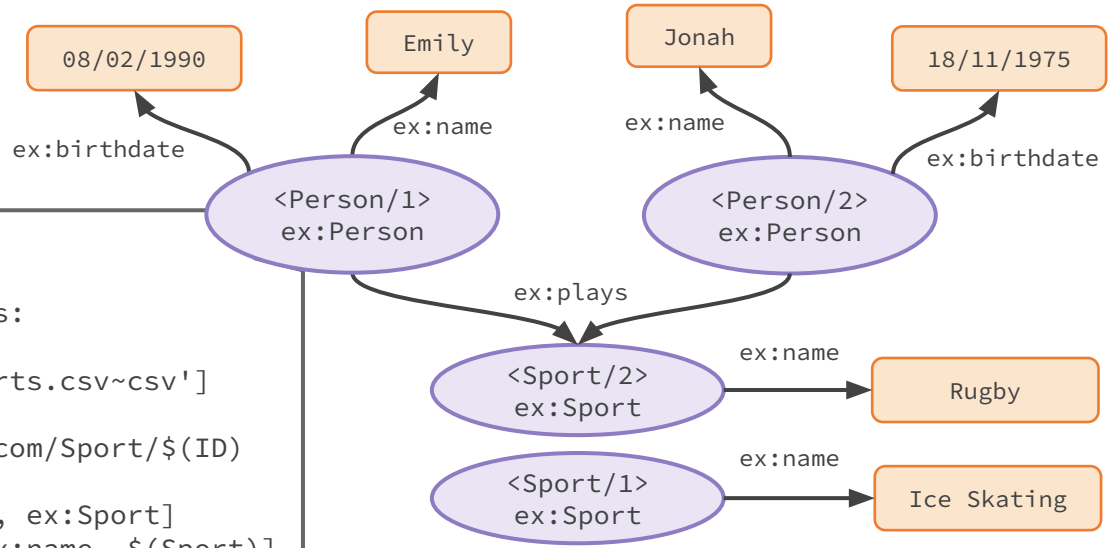


Mapping

```
mappings:
 PERSON:
  sources:
   - ['data/people.csv~csv']
  s: http://ex.com/Person/$(ID)
  po:
   - [a, ex:Person]
   - [ex:name, $(Name)]
   - [ex:birthdate, $(Birthdate)]
   - p: ex:plays
     o:
      - mapping: SPORT
        condition:
          function: equal
          parameters:
           - [str1, $(SportID)]
           - [str2, $(ID)]
```

```
mappings:
 SPORT:
    sources:
      -
['data/sports.csv~csv']
    s:
http://ex.com/Sport/$(ID)
    po:
      - [a, ex:Sport]
      - [ex:name, $(Sport)]
```

Output Knowledge Graph



Hands-on exercise

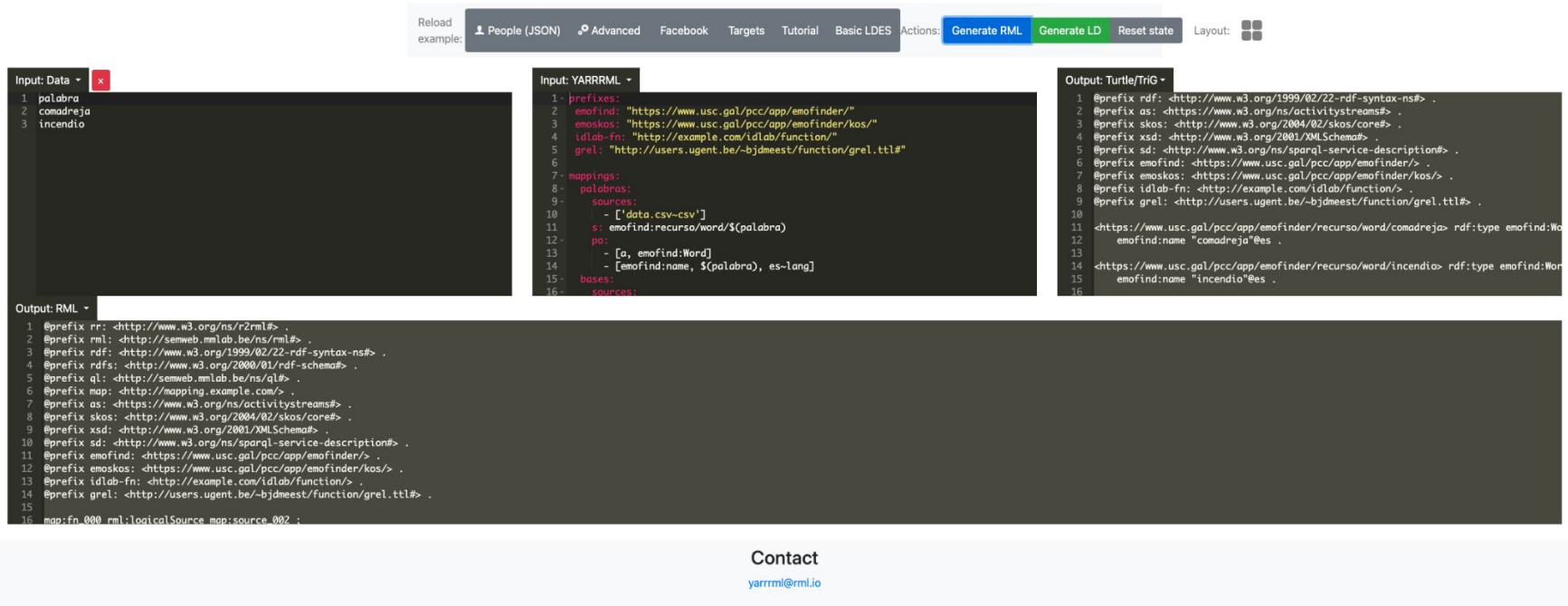
Hands-on exercise

- Creating a KG from a CSV file using YARRRML

Hands-on exercise

- Creating a KG from a CSV file using YARRRML:
  1. Access the resources in: https://bit.ly/4h2rgcW
  2. Create the mapping completing the file ‘yarrrml_template.yml’
  3. Can be tested using Matey or Yatter
  4. Run the mapping and the data with Morph-KGC to create the KG

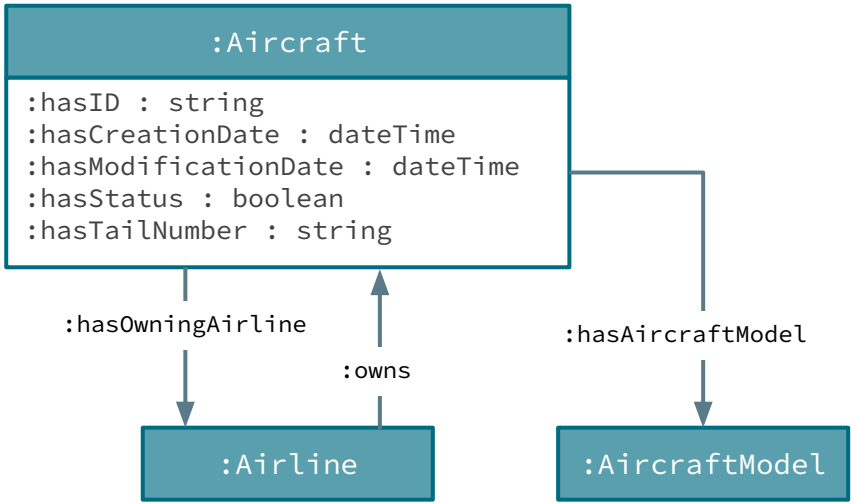
Mapping Supporters: Matey (YARRRML-parser)



https://rml.io/yarrrml/matey/ and https://github.com/RMLio/yarrrml-parser (NodeJS)

Hands-on exercise

- Creating a KG from a CSV file using YARRRML



Resources

https://bit.ly/4h2rgcW

sfo-aircraft-tail-numbers-and-models.csv

- aircraft_id
- aircraft_model
- airline
- creation_date
- modification_date
- status
- tail_number

Resources for KG Construction

17:00 - 17:25

Ontology development

Mapping supporters

KG Construction engines

Benchmarks

The RML ecosystem

Compliant systems

- RMLMapper
- RocketRML
- Chimera
- Morph-KGC
- SDM-RDFizer
- CARML
- RDF Processing Toolkit
- Mapeathor
- FlexRML
- BURP
- MEL
- Excel in RML
- Matey
- Yatter
- Spread2RML
- Dragoman
- RMLEditor
- YARRRML-parser
- mapping-template
- RMLStreamer
- RMLWeaver-js

Interoperability

- SPARQL-Generate
- ShExML
- Helio
- YARRRML
- XRM

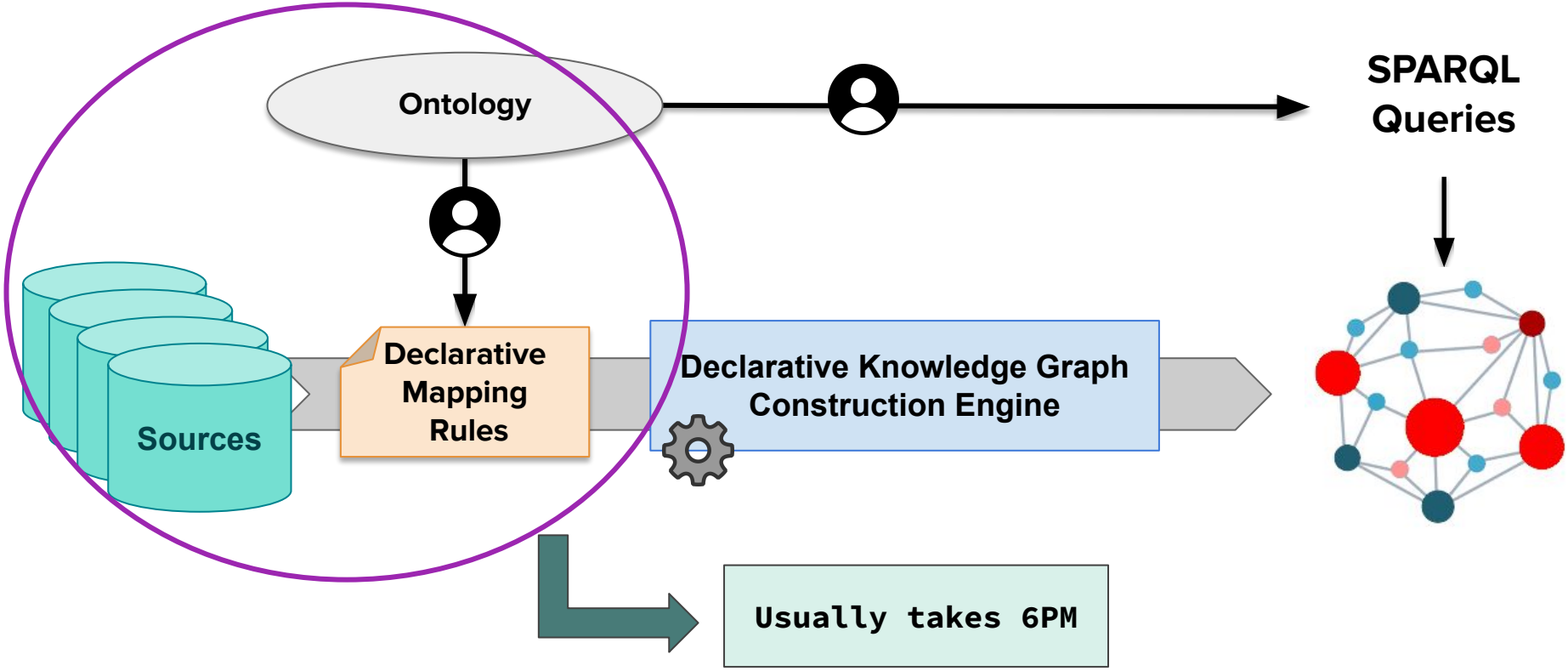
Benchmarks

- GTFS-Madrid Bench
- SDM-Genomic benchmark
- LUBM4OBDA
- KROWN (KGC Challenge)

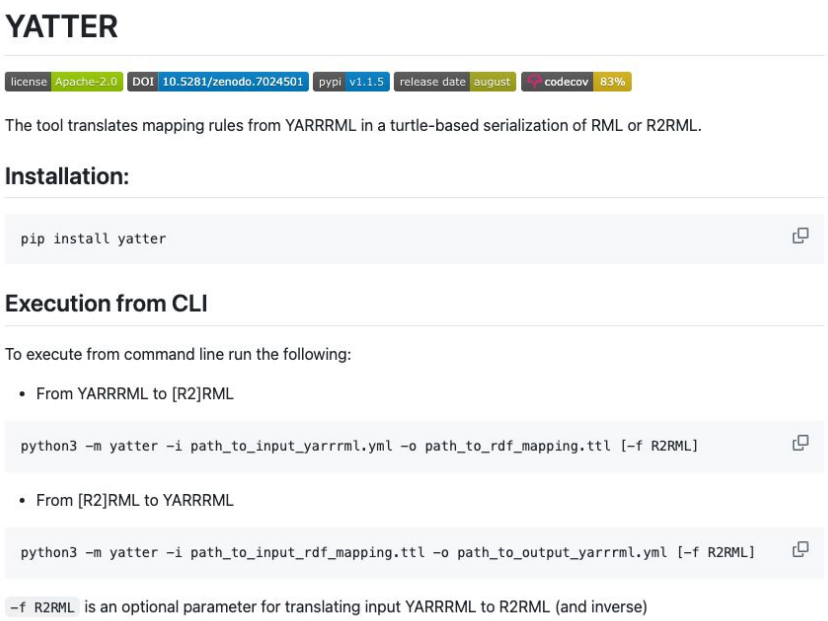
Use cases

- National and European Projects
- European Commission (ERA, PPDS)
- Companies (Orange, BASF, REALE, Bosch, Google)

Mapping rules



Mapping Supporters: Yatter



YATTER

license Apache-2.0 | DOI 10.5281/zenodo.7024501 | pypi v1.1.5 | release date august | codecov 83%

The tool translates mapping rules from YARRRML in a turtle-based serialization of RML or R2RML.

Installation:

```
pip install yatter
```

Execution from CLI

To execute from command line run the following:

- From YARRRML to [R2]RML

```
python3 -m yatter -i path_to_input_yarrrml.yml -o path_to_rdf_mapping.ttl [-f R2RML]
```

- From [R2]RML to YARRRML

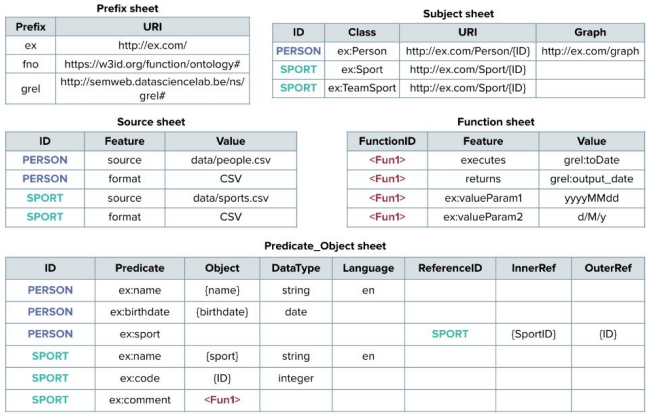
```
python3 -m yatter -i path_to_input_rdf_mapping.ttl -o path_to_output_yarrrml.yml [-f R2RML]
```

`-f R2RML` is an optional parameter for translating input YARRRML to R2RML (and inverse)

- Easy to read translation
- Python-based
- Code: https://github.com/oeg-upm/yatter

Iglesias-Molina, A., Chaves-Fraga, D., Dasoulas, I., & Dimou, A. (2023). Human-friendly RDF graph construction: Which one do you chose?. In International Conference on Web Engineering

Mapping Supporters: Mapeathor



Prefix sheet

Prefix	URI
ex	http://ex.com/
fno	https://w3id.org/function/ontology#
grel	http://semweb.datasciencelab.be/ns/grel#

Subject sheet

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

Source sheet

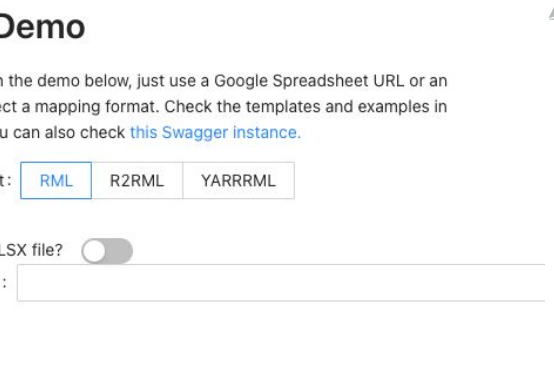
ID	Feature	Value
PERSON	source	data/people.csv
PERSON	format	CSV
SPORT	source	data/sports.csv
SPORT	format	CSV

Function sheet

FunctionID	Feature	Value
<Fun1>	executes	grel:toDate
<Fun1>	returns	grel:output_date
<Fun1>	ex:valueParam1	yyyyMMdd
<Fun1>	ex:valueParam2	d/M/y

Predicate_Object sheet

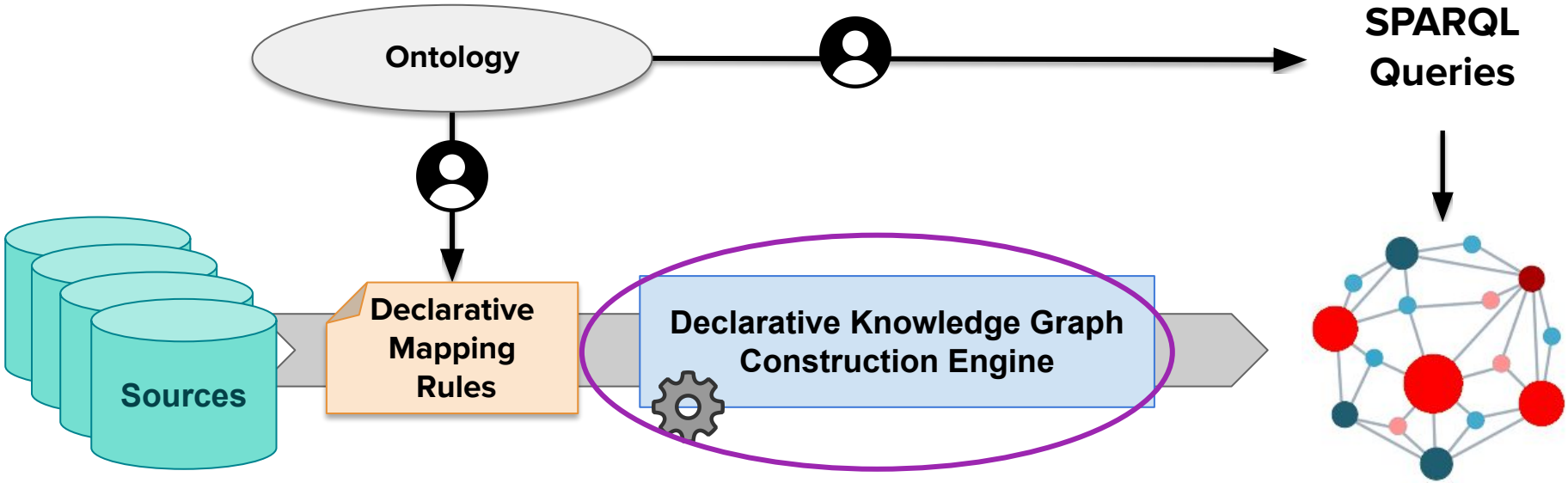
ID	Predicate	Object	DataType	Language	ReferenceID	InnerRef	OuterRef
PERSON	ex:name	{name}	string	en			
PERSON	ex:birthdate	{birthdate}	date				
PERSON	ex:sport				SPORT	{SportID}	{ID}
SPORT	ex:name	{sport}	string	en			
SPORT	ex:code	{ID}	integer				
SPORT	ex:comment	<Fun1>					



- Excel-based mapping
- Translation into YARRRML, R2RML and RML
- REST API, Python Library
- Code: https://github.com/oeg-upm/mapeathor

Iglesias-Molina, A., Pozo-Gilo, L., Dona, D., Ruckhaus, E., Chaves-Fraga, D., & Corcho, O. (2020). Mapeathor: Simplifying the Specification of Declarative Rules for Knowledge Graph Construction. In International Semantic Web Conference

Mapping rules



Engines for KG Construction: SDM-RDFizer

Install at: https://pypi.org/project/rdfizer/ (pip install rdfizer)

Github website: https://github.com/SDM-TIB/SDM-RDFizer

Features:

- Very efficient for large datasets
- Efficient execution of joins and duplicates removal
- Python based
- A bit difficult to run (config file needs to be filled)
- Some quality problems

SDM-RDFizer

Engines for KG Construction: RMLMapper

Download/Install at: https://github.com/RMLio/rmlmapper-java

Features:

- Good parser in terms on data quality
- Java based
- Really simple to run (-d is for removing duplicates):

  java -jar rmlmapper.jar -m mapping.rml -o output.nt -d
- Not very efficient when you have many joins or duplicates

Engines for KG Construction: RMLStreamer

Download/Install at: https://github.com/RMLio/RMLStreamer

Features:

- To mapping stream data into RDF streams
- Scala based
- Not very efficient when you have joins

Engines for KG Construction: FlexRML

Download/Install at: https://github.com/wintechis/flex-rml

Install: https://www.npmjs.com/package/flexrml-node

Features:

- Only supports CSV data
- Super efficient (the best at this moment)
- It can runs materialization on the edge
- Written in C++ (JavaScript version also available)

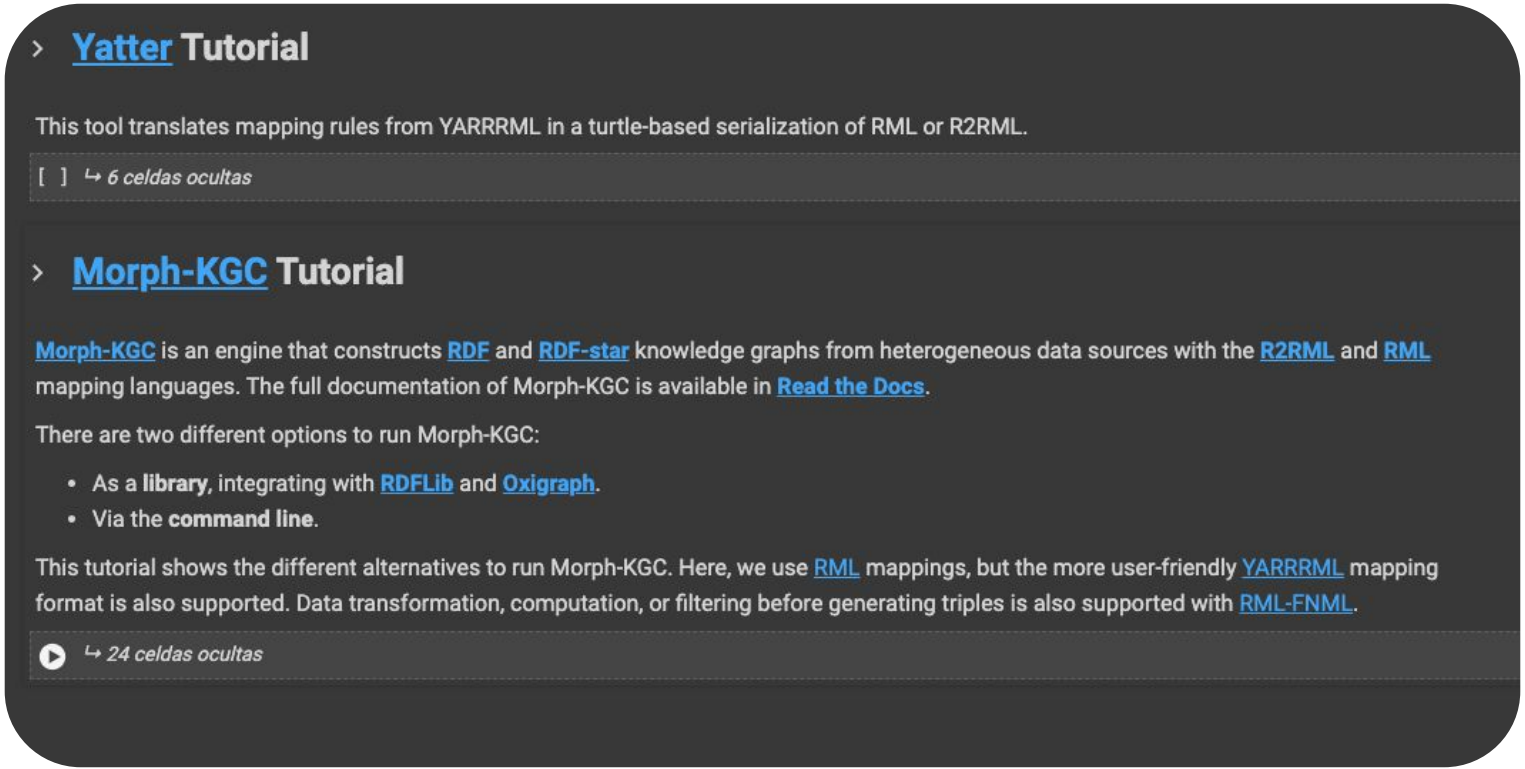
Engines for KG Construction: Morph-KGC

Install at: https://pypi.org/project/morph-kgc/ (pip install morph-kgc)
Documentation: https://morph-kgc.readthedocs.io/en/latest/documentation/

Features:

- Very efficient for big data
- Python based
- Optimizations in planification of the mappings
- Continuous integration for testing
- Generates RDF-star datasets

Let's play with Yatter+Morph-KGC in our example



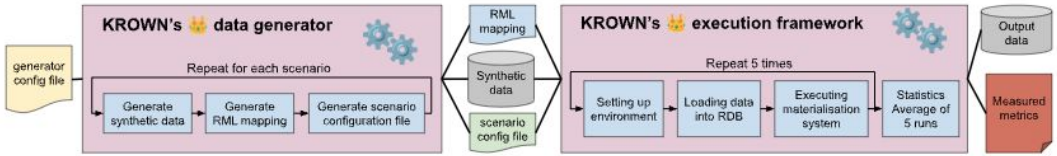
https://github.com/kg-construct/tutorials/tree/main/ecai2024/resources/notebooks

Benchmarks for KG Construction

KROWN 👑: A Benchmark for RDF Graph Materialization

KROWN 👑 is a benchmark for materialization systems to construct Knowledge Graphs from (semi-)heterogeneous data sources using declarative mappings such as RML.

Many benchmarks already exist for virtualization systems e.g. GTFS-Madrid-Bench, NPD, BSBM which focus on complex queries with a single declarative mapping. However, materialization systems are unaffected by complex queries since their input is the dataset and the mappings to generate a Knowledge Graph. Some specialized datasets exist to benchmark specific limitations of materialization systems such as duplicated or empty values in datasets e.g. GENOMICS, but they do not cover all aspects of materialization systems. Therefore, it is hard to compare materialization systems among each other in general which is where KROWN 👑 comes in!



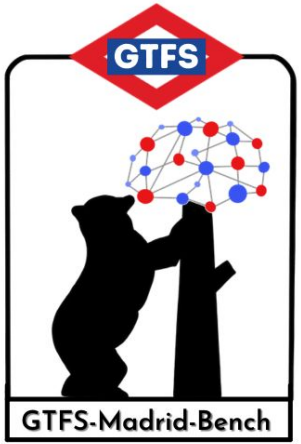
https://github.com/kg-construct/krown

Van Assche, D., Chaves-Fraga, D., & Dimou, A. KROWN: A Benchmark for RDF Graph Materialisation. International Semantic Web Conference (2024)

Benchmarks for KG Construction

The GTFS-Madrid-Bench

We present GTFS-Madrid-Bench, **a benchmark to evaluate declarative KG construction engines** that can be used for the provision of access mechanisms to (virtual) knowledge graphs. Our proposal introduces several scenarios that aim at measuring performance and scalability as well as the query capabilities of all this kind of engines, considering their heterogeneity. The data sources used in our benchmark are derived from the GTFS data files of the subway network of Madrid. They can be transformed into several formats (CSV, JSON, SQL and XML) and scaled up. The query set aims at addressing a representative number of SPARQL 1.1 features while covering usual queries that data consumers may be interested in.



https://github.com/oeg-upm/gtfs-bench

Chaves-Fraga, D., Priyatna, F., Cimmino, A., Toledo, J., Ruckhaus, E., & Corcho, O. (2020). GTFS-Madrid-Bench: A benchmark for virtual knowledge graph access in the transport domain. *Journal of Web Semantics*.

Benchmarks for KG Construction

SDM-GENOMICS Dataset:https://doi.org/10.57702/4c9ivpgs

- Testbeds for duplicate removal and join conditions
- RML support

BSBM: http://wbsg.informatik.uni-mannheim.de/bizer/berlinsparqlbenchmark/

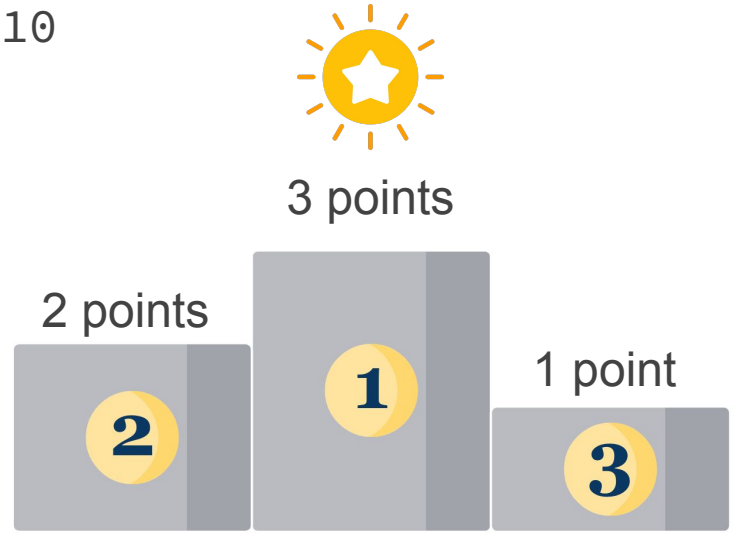
- Well-known benchmark to test triplestores capabilities
- Adapted to virtual KG Construction with R2RML

NPD: https://github.com/ontop/npd-benchmark

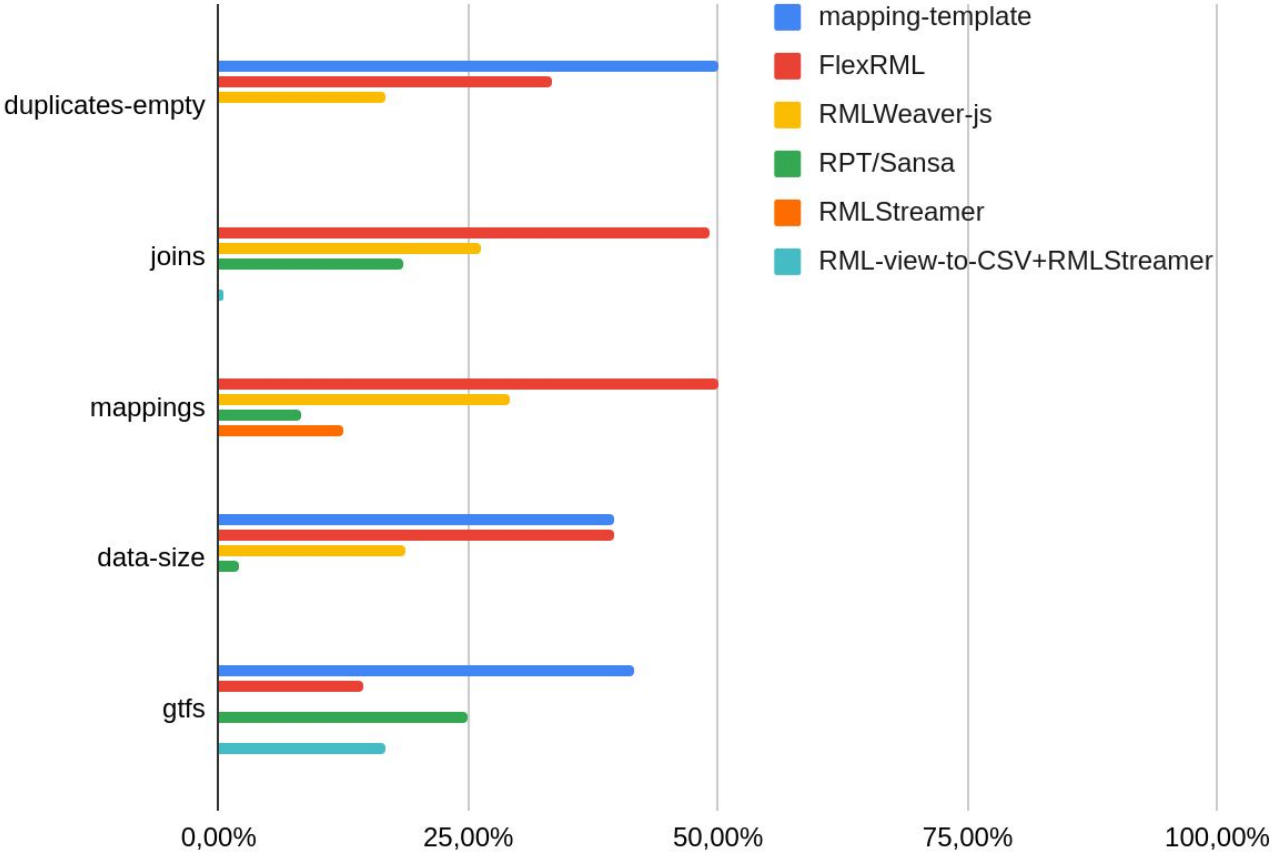
- Focuses on reasoning during virtual KG Construction
- R2RML suport

Last performance results: GTFS-Madrid-Bench + KG parameters

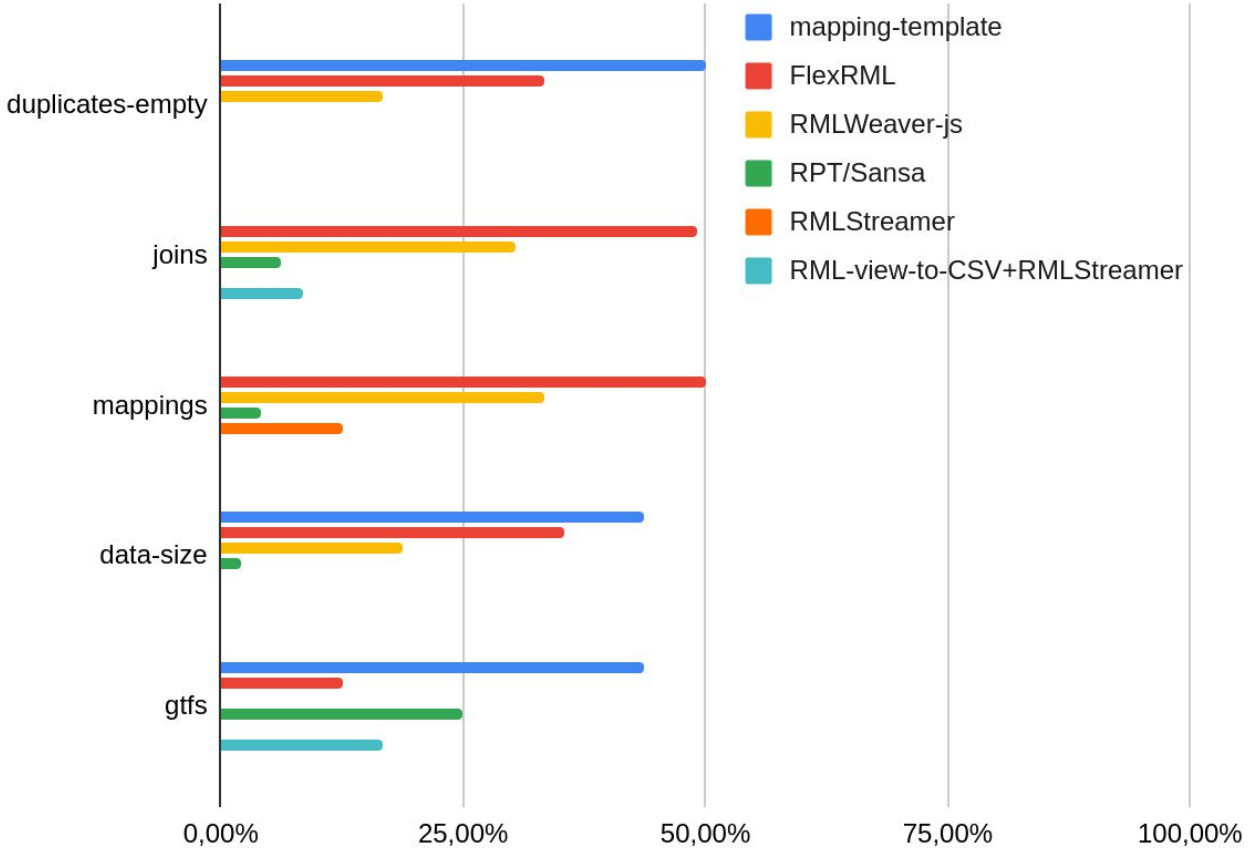
- **3 Variables**: execution time, CPU and RAM usage
- **65 Scenarios**:
  - KG parameters: 57
    - Duplicates & empty values: 10
    - Data size: 8
    - Mappings: 4
    - Joins: 35
  - GTFS: 8
- **Who gets the award?**
  - Top 3 tools per scenario and variable get points
  - Final count of all points



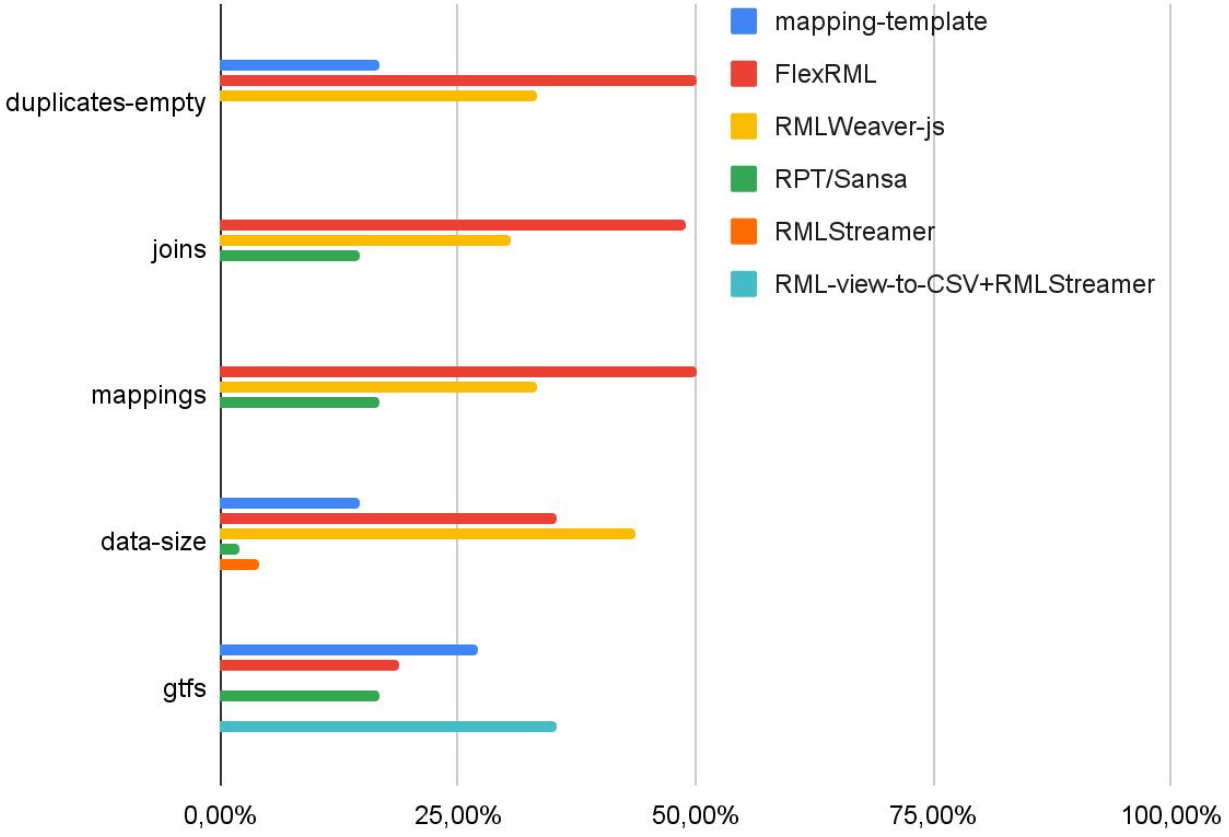
Track 2: Points for execution time



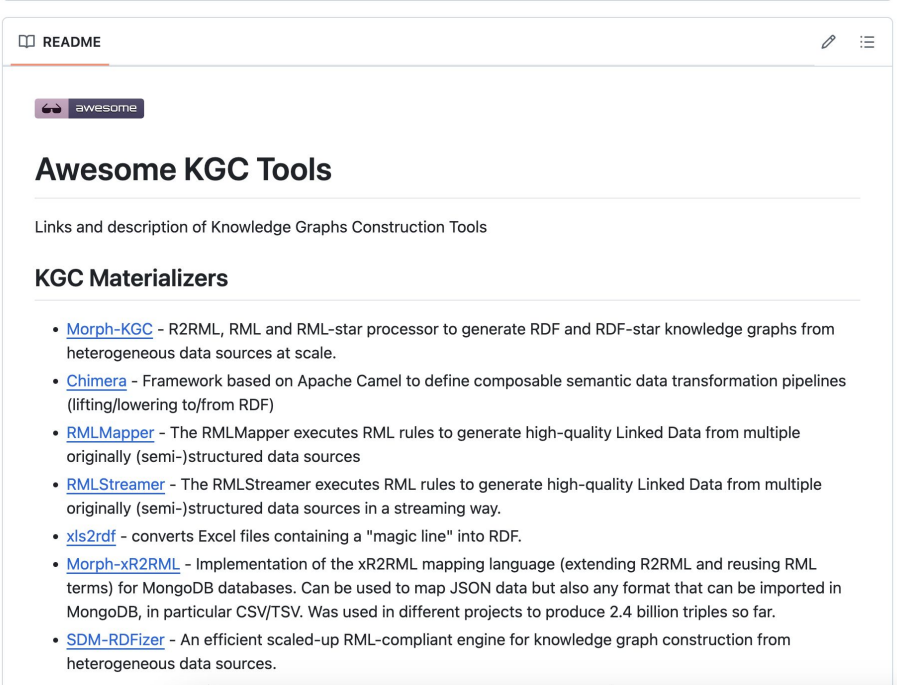
Track 2: Points for CPU usage



Track 2: Points for RAM usage



Other resources: AWESOME-KGC-TOOLS list



README

awesome

Awesome KGC Tools

Links and description of Knowledge Graphs Construction Tools

KGC Materializers

- Morph-KGC - R2RML, RML and RML-star processor to generate RDF and RDF-star knowledge graphs from heterogeneous data sources at scale.
- Chimera - Framework based on Apache Camel to define composable semantic data transformation pipelines (lifting/lowering to/from RDF)
- RMLMapper - The RMLMapper executes RML rules to generate high-quality Linked Data from multiple originally (semi-)structured data sources
- RMLStreamer - The RMLStreamer executes RML rules to generate high-quality Linked Data from multiple originally (semi-)structured data sources in a streaming way.
- xls2rdf - converts Excel files containing a "magic line" into RDF.
- Morph-xR2RML - Implementation of the xR2RML mapping language (extending R2RML and reusing RML terms) for MongoDB databases. Can be used to map JSON data but also any format that can be imported in MongoDB, in particular CSV/TSV. Was used in different projects to produce 2.4 billion triples so far.
- SDM-RDFizer - An efficient scaled-up RML-compliant engine for knowledge graph construction from heterogeneous data sources.

https://kg-construct.github.io/awesome-kgc-tools/

Ontology Development

Sunday 20th (tomorrow)

09:00 - 12:30

Room: TBC

https://chowlk.linkeddata.es/

https://lot.linkeddata.es/

Boosting ontology conceptualization with Chowlk

Tutorial at ECAI 2024

1. Description

This tutorial aims to support ontology developers to reduce the time consumed in ontology implementation by generating graphical ontology conceptualizations that can be converted into code automatically. Participants will gain knowledge about the Chowlk visual notation for ontologies, how to use the Chowlk converter to accelerate ontology implementation based on the generated conceptualizations, how to use the related resources as templates and, finally, how to create new templates.

This tutorial will focus on explaining 1) the Chowlk notation, origin, motivation and current status; 2) how to use the converter to accelerate the ontology implementation activity based on the generated diagram; 3) how to use the related resources as templates; and finally 4) how to create new templates. The tutorial will be mainly practical and interactive, driving the attendees through the Chowlk framework features by using them. In this sense we will "Introduce expert non-specialists to an AI subarea" (ontology development) and "Provide instruction in established but specialized AI methodologies."

2. Speakers

María Poveda-Villalón

mpoveda (at) fi.upm.es

Dr. María Poveda-Villalón is an associate professor at the Artificial Intelligence Department of the Universidad Politécnica de Madrid and is also part of the Ontology Engineering Group research lab. Her research activities focus on Ontological Engineering, Ontology Evaluation, Knowledge Representation and the Semantic Web. Previously she finished her studies in Computer Science (2009) by Universidad Politécnica de Madrid, and then she moved to study the Artificial Intelligence Research Master finished in 2010 in the same university. She has contributed to the ontology engineering field developing methodologies and tools like OOPS! (Ontology Pitfall Scanner!) which has been broadly adopted by the community, the LOT methodology for building ontologies and the Chowk framework for ontology conceptualization and implementation. She has worked in the context of Spanish research projects as well as in European. She has contributed to the organization of the "Linked Data in Architecture and Construction Workshop" since 2015 edition, the "13th OWL: Experiences and Directions Workshop and 5th OWL reasoner evaluation workshop" in 2016, the Workshop on Ontology Patterns (WOP) in 2019 and 2022, the "Linked Energy Data Vocamp" in 2015 and she has organized the 2nd Summer School of Linked Data in Architecture and Construction. Finally, she is part of the W3C Web of Things Working Group. Her teaching activities involve teaching semantic web and ontological engineering in different courses at official degrees, masters courses, MOOCs and online masters as well to deliver courses to the administration and private companies. She has carried out similar tutorials as "Catching up with ontological engineering. To git-commit and beyond" at EKAW2018 and "Integrating ontological development with software engineering trends" FDL2019.

Conclusions

- Systematic construction of RDF Knowledge Graphs
- RML as declarative language for KGC
- YARRRML as human-syntax to facilite the use
- Several engines with their own features
- Benchmarks to test performance (time, memory, CPU)
- Big community (>180 members in W3C)
- Towards the standardization process
- Impact over companies (Google, Orange, Bosch, etc)
- In-production on European Data Spaces (PPDS, ERA)

How do you construct a Knowledge Graph?

David Chaves-Fraga and
Ana Iglesias-Molina

Thanks for attending!