

TIB LEIBNIZ INFORMATION CENTRE
FOR SCIENCE AND TECHNOLOGY
UNIVERSITY LIBRARY



Feeding PIDza to VIVO: data ingest with SPARQL-Generate

12th Annual VIVO Conference 2021

Maxime Lefrançois (EMSE)

[<https://orcid.org/0000-0001-9814-8991>]

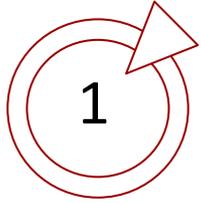
Sandra Mierz (TIB)

[<https://orcid.org/0000-0002-8913-9011>]

SPONSORED BY THE



Federal Ministry
of Education
and Research



generate2vivo

- What is it?
- data sources & queries

SPARQL-Generate

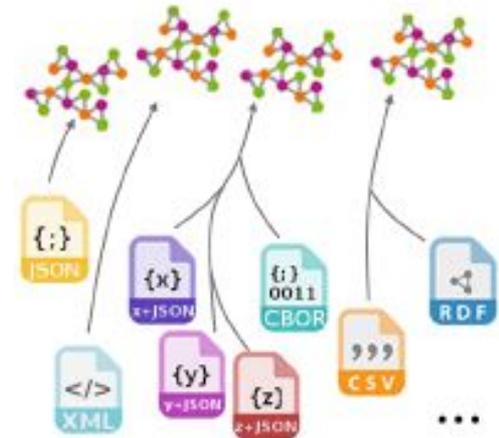
- What is it?
- How can you use it?



SPARQL-Generate - query and generate both RDF and text

An expressive template-based language to generate RDF streams or text streams from RDF datasets *and* document streams in arbitrary formats.

- Transform multiple sources ...
- ... having heterogeneous formats
- Be extensible to new data formats
- Be easy to use by Semantic Web experts
- Integrate in a typical semantic web engineering workflow
- Be flexible and easily maintainable
- Contextualize the transformation with an RDF Dataset
- Modularize / decouple / compose transformations
- Enable data transformation, aggregates, filters, ...
- Generate RDF Lists
- Performant
- Transform binary formats as well as textual formats
- Transform big documents (output HDT)
- Generate RDF streams from streams



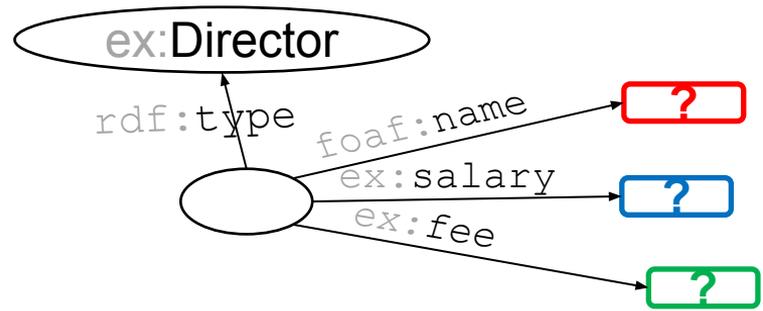
RDF generation process

```

{"DirectorCompensation":
 [
  {
    "DirectorName": "Jane Doe",
    "Salary": "1,000",
    "Bonus": "1,000",
    "DirectorFees": "1,000",
    "FairValueOfOptionsGranted": "1,000"
  },
  {
    "DirectorName": "John Doe",
    "Salary": "1,000",
    "Bonus": "1,000",
    "DirectorFees": "1,000",
    "FairValueOfOptionsGranted": "1,000"
  },
  {
    "DirectorName": "All Directors",
    "Salary": "2,000",
    "Bonus": "2,000",
    "DirectorFees": "2,000",
    "FairValueOfOptionsGranted": "2,000"
  }
 ]
}

```

Selection patterns
 Xpath, JSONpath, CSS selectors, regex, etc.



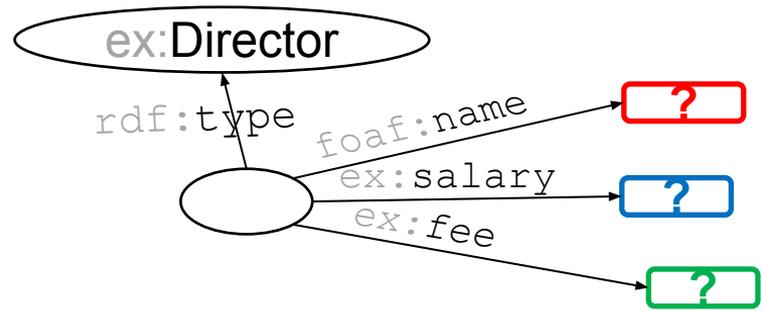
Graph pattern definition
 + Select ontologies

First SPARQL-Generate query

```

{"DirectorCompensation":
 [
  {
    "DirectorName": "Jane Doe",
    "Salary": "1,000",
    "Bonus": "1,000",
    "DirectorFees": "1,000",
    "FairValueOfOptionsGranted": "1,000"
  },
  {
    "DirectorName": "John Doe",
    "Salary": "1,000",
    "Bonus": "1,000",
    "DirectorFees": "1,000",
    "FairValueOfOptionsGranted": "1,000"
  }
 ]
}

```



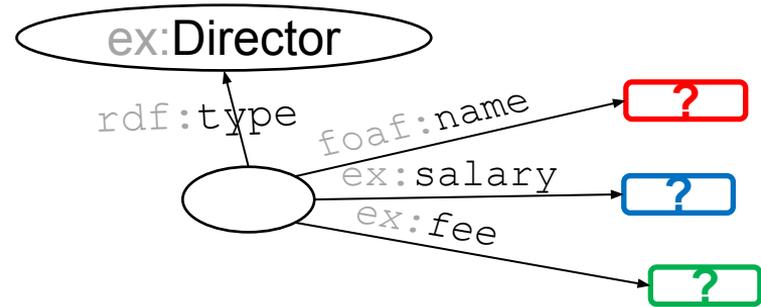
```

1 PREFIX fun: <http://w3id.org/sparql-generate/fn/>
2 PREFIX iter: <http://w3id.org/sparql-generate/iter/>
3 GENERATE {
4   [] a ex:Director ;
5     foaf:name ?name ;
6     ex:salary ?salary ;
7     ex:fee ?fee .
8 }
9 SOURCE <myDocument> AS ?myDocument
10 ITERATOR iter:JSONPath( ?myDocument , "$.DirectorCompensation[*]") AS ?director
11 WHERE {
12   BIND( fun:JSONPath( ?director , "$.DirectorName") AS ?name )
13   BIND( fun:JSONPath( ?director , "$.Salary") AS ?salary )
14   BIND( fun:JSONPath( ?director , "$.DirectorFee") AS ?fee )
15 }

```

The Graph Pattern Definition is here

```
{ "DirectorCompensation":  
  [  
    {  
      "DirectorName": "Jane Doe",  
      "Salary": "1,000",  
      "Bonus": "1,000",  
      "DirectorFees": "1,000",  
      "FairValueOfOptionsGranted": "1,000"  
    },  
    {  
      "DirectorName": "John Doe",  
      "Salary": "1,000",  
      "Bonus": "1,000",  
      "DirectorFees": "1,000",  
      "FairValueOfOptionsGranted": "1,000"  
    }  
  ]  
}
```



```
1 PREFIX fun: <http://w3id.org/sparql-generate/fn/>  
2 PREFIX iter: <http://w3id.org/sparql-generate/iter/>  
3 GENERATE {  
4   [] a ex:Director ;  
5     foaf:name ?name ;  
6     ex:salary ?salary ;  
7     ex:fee ?fee .  
8 }  
9 SOURCE <myDocument> AS ?myDocument  
10 ITERATOR iter:JSONPath( ?myDocument , "$.DirectorCompensation[*]") AS ?director  
11 WHERE {  
12   BIND( fun:JSONPath( ?director , "$.DirectorName") AS ?name )  
13   BIND( fun:JSONPath( ?director , "$.Salary") AS ?salary )  
14   BIND( fun:JSONPath( ?director , "$.DirectorFee") AS ?fee )  
15 }
```

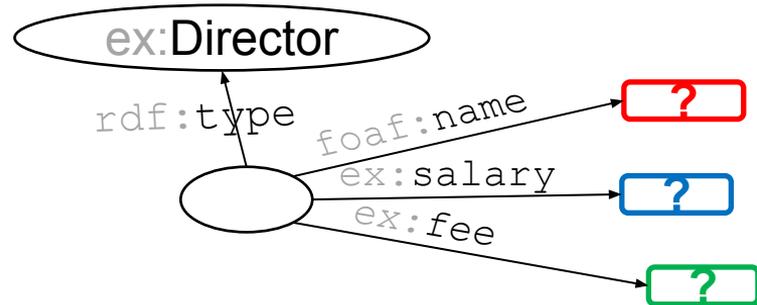
Query RDF Dataset + « Documentset »

```
{"DirectorCompensation":
```

```
[
```

```
  {  
    "DirectorName": "Jane Doe",  
    "Salary": "1,000",  
    "Bonus": "1,000",  
    "DirectorFees": "1,000",  
    "FairValueOfOptionsGranted": "1,000"  
  },
```

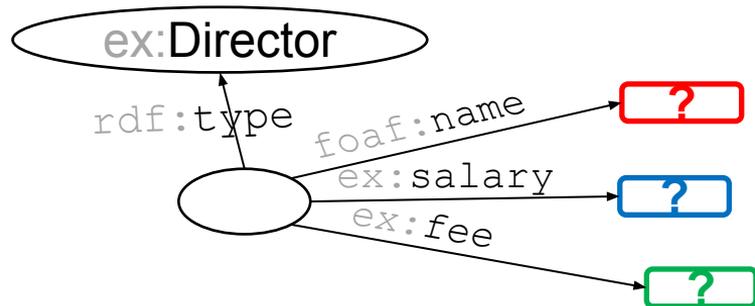
```
  {  
    "DirectorName": "John Doe",  
    "Salary": "1,000",  
    "Bonus": "1,000",  
    "DirectorFees": "1,000",  
    "FairValueOfOptionsGranted": "1,000"  
  }  
]
```



```
1 PREFIX fun: <http://w3id.org/sparql-generate/fn/>
2 PREFIX iter: <http://w3id.org/sparql-generate/iter/>
3 GENERATE {
4   [] a ex:Director ;
5     foaf:name ?name ;
6     ex:salary ?salary ;
7     ex:fee ?fee .
8 }
9 SOURCE <myDocument> AS ?myDocument
10 ITERATOR iter:JSONPath( ?myDocument , "$.DirectorCompensation[*]") AS ?director
11 WHERE {
12   BIND( fun:JSONPath( ?director , "$.DirectorName") AS ?name )
13   BIND( fun:JSONPath( ?director , "$.Salary") AS ?salary )
14   BIND( fun:JSONPath( ?director , "$.DirectorFee") AS ?fee )
15 }
```

Iterator Functions (dereferenceable)

```
{ "DirectorCompensation":  
  [  
    {  
      "DirectorName": "Jane Doe",  
      "Salary": "1,000",  
      "Bonus": "1,000",  
      "DirectorFees": "1,000",  
      "FairValueOfOptionsGranted": "1,000"  
    },  
    {  
      "DirectorName": "John Doe",  
      "Salary": "1,000",  
      "Bonus": "1,000",  
      "DirectorFees": "1,000",  
      "FairValueOfOptionsGranted": "1,000"  
    }  
  ]  
}
```



```
1 PREFIX fun: <http://w3id.org/sparql-generate/fn/>  
2 PREFIX iter: <http://w3id.org/sparql-generate/iter/>  
3 GENERATE {  
4   [] a ex:Director ;  
5     foaf:name ?name ;  
6     ex:salary ?salary ;  
7     ex:fee ?fee .  
8 }  
9 SOURCE <myDocument> AS ?myDocument  
10 ITERATOR iter:JSONPath( ?myDocument , "$.DirectorCompensation[*]") AS ?director  
11 WHERE {  
12   BIND( fun:JSONPath( ?director , "$.DirectorName") AS ?name )  
13   BIND( fun:JSONPath( ?director , "$.Salary") AS ?salary )  
14   BIND( fun:JSONPath( ?director , "$.DirectorFee") AS ?fee )  
15 }
```

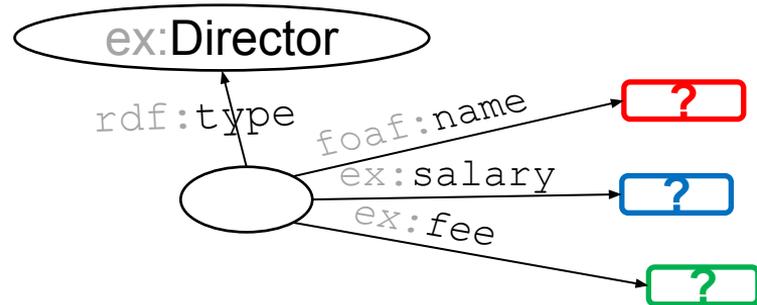
Binding Functions (that's standard SPARQL)

```
{"DirectorCompensation":
```

```
[
```

```
  {  
    "DirectorName": "Jane Doe",  
    "Salary": "1,000",  
    "Bonus": "1,000",  
    "DirectorFees": "1,000",  
    "FairValueOfOptionsGranted": "1,000"  
  },
```

```
  {  
    "DirectorName": "John Doe",  
    "Salary": "1,000",  
    "Bonus": "1,000",  
    "DirectorFees": "1,000",  
    "FairValueOfOptionsGranted": "1,000"  
  }  
]
```



```
1 PREFIX fun: <http://w3id.org/sparql-generate/fn/>
2 PREFIX iter: <http://w3id.org/sparql-generate/iter/>
3 GENERATE {
4   [] a ex:Director ;
5     foaf:name ?name ;
6     ex:salary ?salary ;
7     ex:fee ?fee .
8 }
9 SOURCE <myDocument> AS ?myDocument
10 ITERATOR iter:JSONPath( ?myDocument , "$.DirectorCompensation[*]") AS ?director
11 WHERE {
12   BIND( fun:JSONPath( ?director , "$.DirectorName") AS ?name )
13   BIND( fun:JSONPath( ?director , "$.Salary") AS ?salary )
14   BIND( fun:JSONPath( ?director , "$.DirectorFee") AS ?fee )
15 }
```

It extends SPARQL, so we got for free ...

- Implementable on top of existing SPARQL engines
- Easy to learn when you know SPARQL
- You get to know SPARQL better when you use SPARQL-Generate
- Output template (Basic Graph Pattern) **is not subject-centric**

- SPARQL 1.1 FILTER BIND OPTIONAL,...

- SPARQL 1.1 Standard binding functions and operators

STR LANG LANGMATCHES DATATYPE BOUND IRI URI BNODE RAND ABS CEIL FLOOR ROUND CONCAT SUBSTR STRLEN REPLACE UCASE LCASE
ENCODE_FOR_URI CONTAINS STRSTARTS STRENDS STRBEFORE STRAFTER YEAR MONTH DAY HOURS MINUTES SECONDS TIMEZONE TZ NOW
UUID STRUUID MD5 SHA1 SHA256 SHA384 SHA512 COALESCE IF STRLANG STRDT sameTerm isIRI isURI isBLANK isLITERAL isNUMERIC REGEX
+ * / = > < || &&

- SPARQL 1.1 Binding functions extension mechanism
- SPARQL 1.1 ORDER LIMIT OFFSET
- SPARQL 1.1 GROUP BY, HAVING, Aggregate functions

COUNT SUM MIN MAX AVG SAMPLE GROUP_CONCAT

Iterators work in batches

```
7 #date;store;amount;consumer
8 #2019-05-01T00:50:09-07:00;95;352.33;Raja
9 #2019-05-01T23:52:06-07:00;69;354.95;Tyrone
10
11 GENERATE {
12   <{?storeIRI}> a ex:Store ;
13   ex:sell [
14     a ex:Purchase ;
15     ex:customer [ a foal:Person ; foaf:name ?consumer ] ;
16     ex:amount ?{ xsd:decimal(?amount) } ;
17     ex:date "{?dateTimeStr}"^^xsd:dateTime
18   ] .
19 }
20 ITERATOR iter:CSV(<http://example.com/consumption.csv>, 1000, true ,"\",";","\n",
21   "date", "store", "amount", "consumer" )
22 AS ?dateTimeStr ?store ?amount ?consumer
23 WHERE {}
```

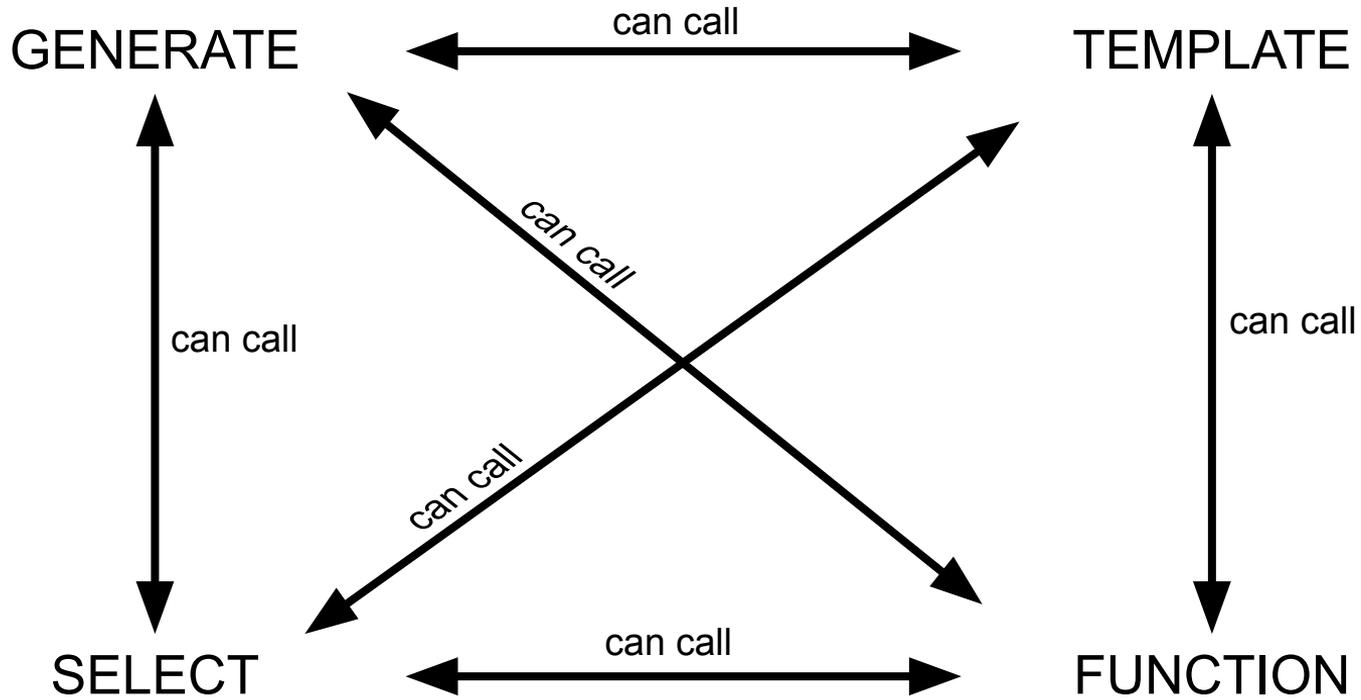
```
7 GENERATE {
8   <events/{?eventId}> a ex:Event;
9   ex:type ?type;
10  ex:happenedAt ?datetime;
11 }
12 ITERATOR iter:WebSocket("wss://api.gemini.com/v1/marketdata/BTCUSD") AS ?events
13 WHERE {
14   BIND(fun:JSONPath(?events, "$.type" ) AS ?type)
15   BIND(xsd:string(fun:JSONPath(?events, "$.eventId" )) AS ?eventId)
16   BIND(fun:dateTime(xsd:string(fun:JSONPath(?events, "$.timestampms" ))) AS ?
17     datetime)
18 }
```

Generate queries can be modularized and distributed

```
16 GENERATE {
17
18 <loc/{?idx}> a sosa:FeatureOfInterest;
19   rdfs:label ?name;
20   geo:lat ?lat;
21   geo:long ?long ;
22   ex:pollution ?aqi .
23
24   GENERATE <findEvents>( <http://example.org/loc/{?idx}>, ?lat, ?long ) .
25
26 }
```

```
17 GENERATE <findEvents>( ?near, ?lat, ?long ) {
18   <event/{?eid}> a event:Event;
19     foaf:basedNear ?near ;
20     dc:date ?date ;
21     rdfs:label ?title;
22     rdfs:comment "{?desc}"^^rdf:HTML .
23 }
24 ITERATOR iter:XPath(<http://api.eventful.com/rest/events/
  search?app_key=9p3bTRHL2NTzVgxG&location={STR(?lat)},{STR(?long)}&within=20>,"/
  search/events/event", "/event/@id", "/event/title/text()", "/event/description/
  text()", "/event/start_time/text()") AS ?event ?eid ?title ?desc ?eventDate
25 BIND("{ REPLACE( ?eventDate, " ", "T" ) }"^^xsd:dateTime AS ?date)
26 WHERE {
27   ORDER BY ?eventDate
28   LIMIT 3
```

There is more than GENERATE queries



Try it out: <https://w3id.org/sparql-generate/>

A language, and an open-source implementation based on Apache Jena

SPARQL-Generate can generate RDF streams or text streams from RDF, SQL, XML, JSON, CSV, GeoJSON, HTML, CBOR, plain text with regular expressions, large CSV documents, MQTT or WebSocket streams, HTTP operations.

Useable as:

- Web playground + tutorial
- Java library (available on Maven Central)
- Executable JAR
- Sublime Text package

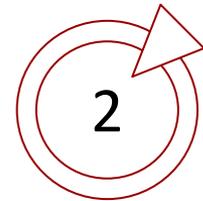


generate2vivo

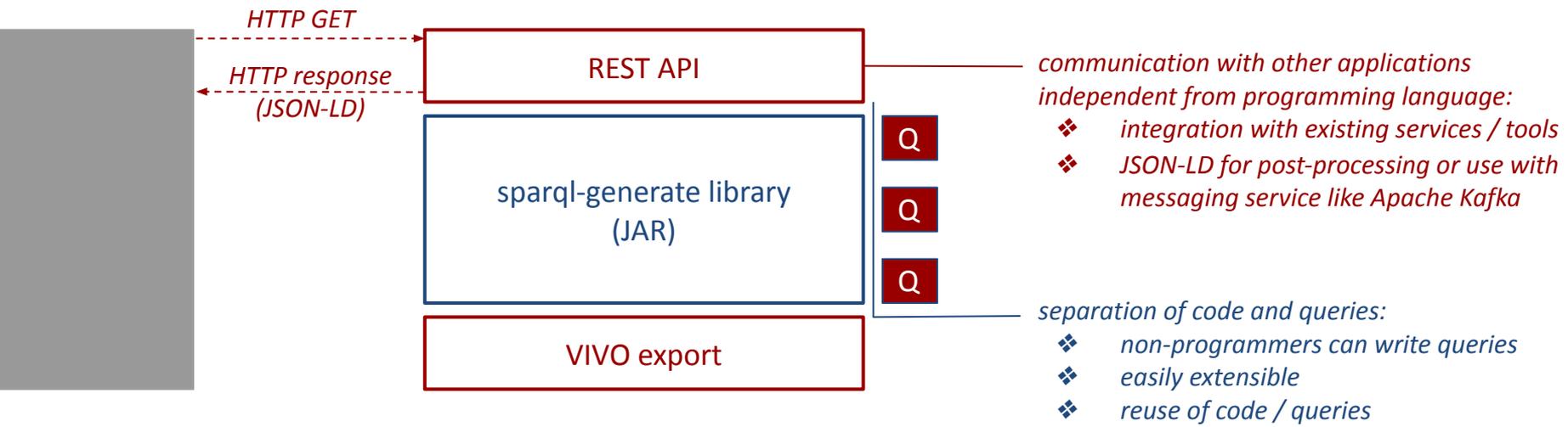
- What is it?
- data sources & queries

SPARQL-Generate

- What is it?
- How can you use it?



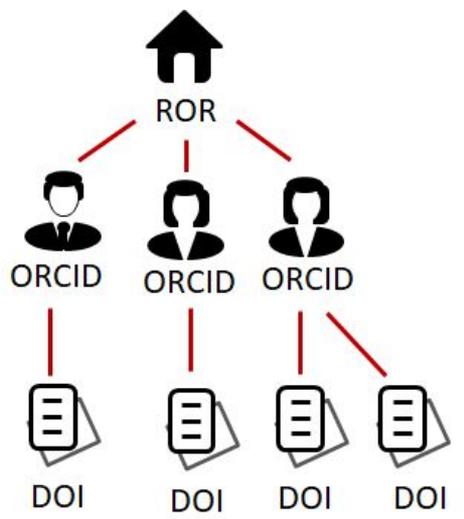
Introducing generate2vivo



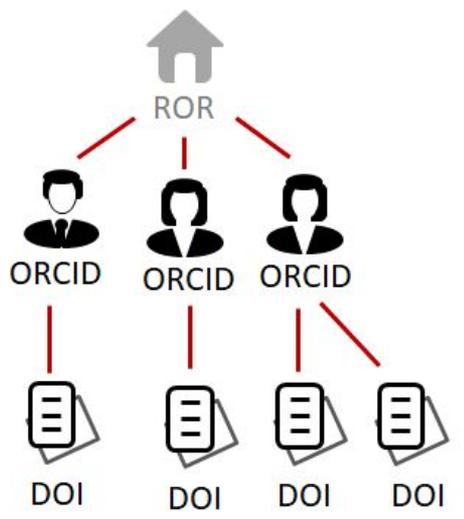
generic transformation tool that is easily extensible and integrable in Data Ingest processes

data sources & queries

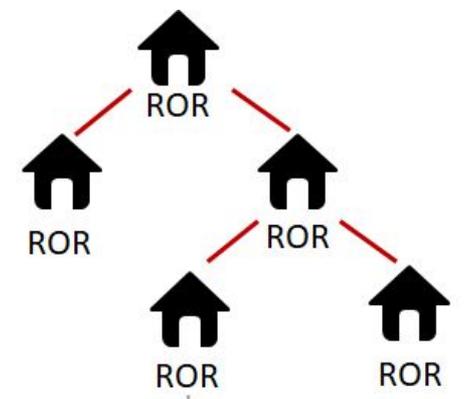
Datacite Commons



ORCID



ROR

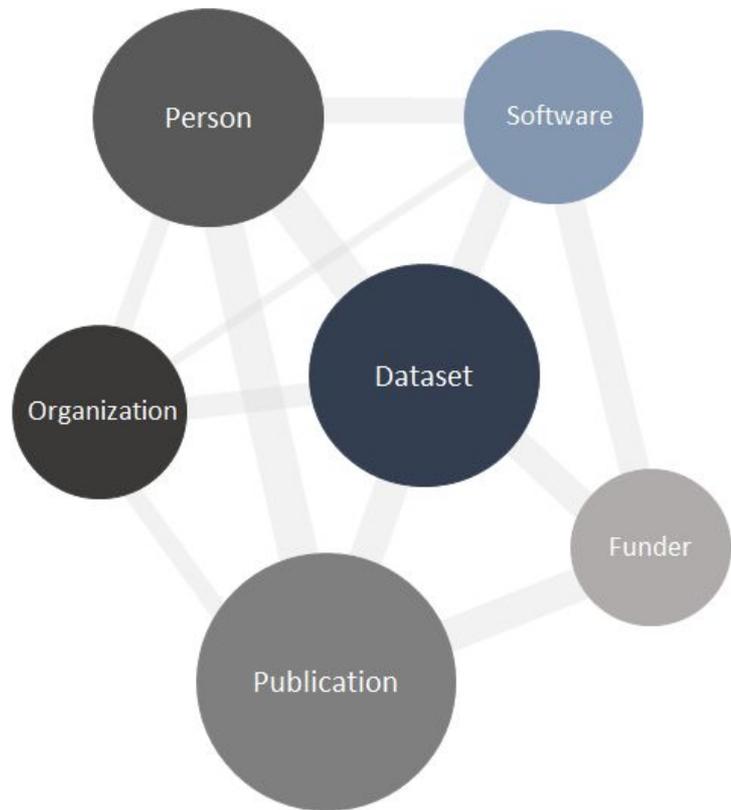


Datacite Commons

Datacite Commons includes:

- *all ROR,*
- *all ORCID IDs,*
- *all of DataCite's DOIs,*
- *nine million Crossref DOIs,*
- *Crossref Funder ID,*
- *and Registry of Research Data Repositories (re3data) records*

Source: Project Freya

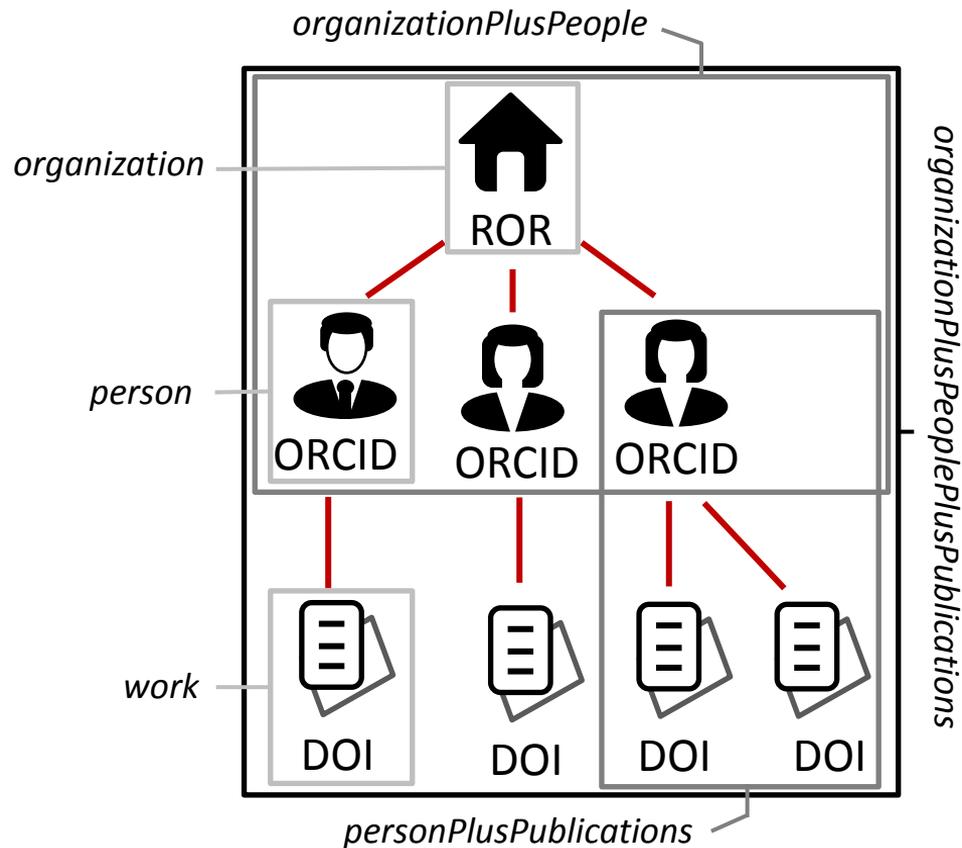


Datacite Commons : queries

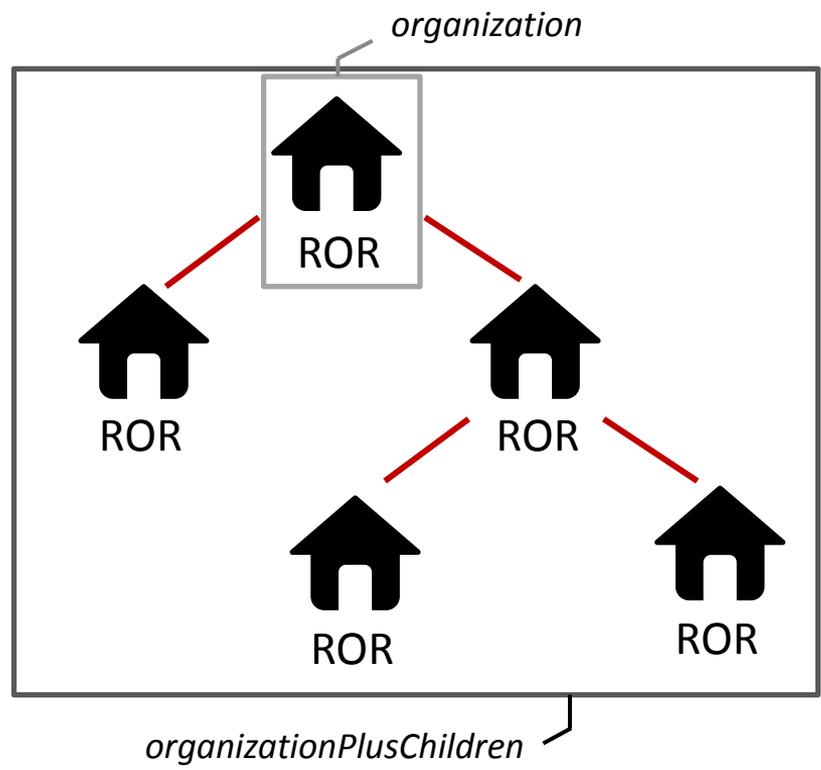
Datacite Commons includes:

- all ROR,
- all ORCID IDs,
- all of DataCite's DOIs,
- nine million Crossref DOIs,
- Crossref Funder ID,
- and Registry of Research Data Repositories (re3data) records

Source: Project Freya



ROR : queries



ORCID: queries

Datacite Commons:

organization - people:

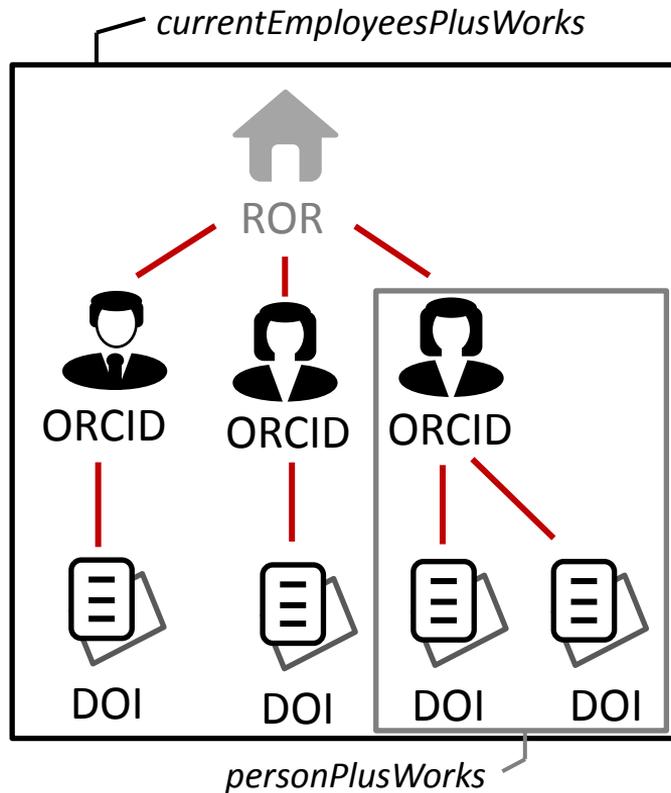
- connected via ORCID affiliation
- no time limit

Source: Lupo, ORCID

person - publication:

- 100% DOI issued by Datacite
- ~8% DOI issued by CrossRef

Source: FAIRsFAIR



ORCID:

organization - people:

- filter only current employees

person - works:

- all DOI in ORCID profile

Complete runs

	Datacite Commons <i>organizationPlusPeoplePlusPublications</i>	ORCID <i>currentEmployeesPlusWorks</i>
TIB	<p>organizations Organization (1)</p> <p>people Person (72)</p> <p>~ 413sec</p>	<p>research Academic Article (340) Article (344) Blog Posting (1) Book (10) Chapter (25) Report (17) Thesis (1)</p> <p>~ 61sec</p>
University Osnabrück	<p>organizations Organization (1)</p> <p>people Person (453)</p> <p>~ 1826sec</p>	<p>research ★ Academic Article (694) Article (694) Book (2) Chapter (10) Report (2) Thesis (2)</p> <p>~ 351sec</p>

★ no differentiated mapping for ORCID works included yet, everything mapped to article

Try it out:

<https://labs.tib.eu/tapir/>



TAPIR Labs

Try out generate2vivo on TAPIR Labs

To provide an insight into our current research and development work, we have made the tool generate2vivo and a connected VIVO instance available in TIB Labs. Here, queries can be tried out and the imported data can be viewed directly in VIVO.

Link to generate2vivo : [generate2vivo](#)

Link to VIVO instance : [VIVO](#)

Let's stay in touch!

Maxime Lefrançois

<https://www.maxime-lefrancois.info>

Sandra Mierz

<https://projects.tib.eu/tapir/en/>

References

- Project Freya:
Fenner, M. (2020b). Powering the pid graph: Announcing the datacite graphql api.
<https://doi.org/10.5438/YFCK-MV39>
- FAIRsFAIR :
Braukmann, Ricarda, Lambert, Simon, Fenner, Martin, Wimalaratne, Sarala, Ulrich, Robert, & Davidson, Joy. (2021, April). FAIRsFAIR Repository Support Series - The role of Repositories in enabling Persistent Identifier (PID) Graphs - Slides. Zenodo. <http://doi.org/10.5281/zenodo.4708938>
- Lupo :
https://github.com/datacite/lupo/blob/master/app/graphql/types/organization_type.rb
(retrieved 2021/22/06)
- ORCID :
<https://info.orcid.org/faq/how-do-i-find-orcid-record-holders-at-my-institution/>
(retrieved 2021/22/06)