

Shapes applications and tools

Jose Emilio Labra Gayo
WESO Research group
University of Oviedo, Spain



Contents

Introduction to Knowledge graphs

Types of Knowledge Graphs:

RDF, Property graphs, Wikibase, RDF-Star

Shaping RDF: ShEx & SHACL

Shaping other types of Knowledge graphs:

Wikibase and Wikidata graphs

Property Graphs

RDF-1.2

 Applications:

Inferring shapes from data, Knowledge Graphs Subsets, etc.

Some applications of Shapes

Traditional application: Validate RDF data

Tools to understand/manage the data models

- Visualizations, HTML page generation

- Editors

- Obtaining shapes from data

Creating subsets from Shapes

Other applications:

- Continuous integration

- Generate code, SPARQL queries

- Optimize SPARQL queries or triplestores based on their shapes

- ...

Tools to understand/edit shapes schemas

UML-like diagram visualizations

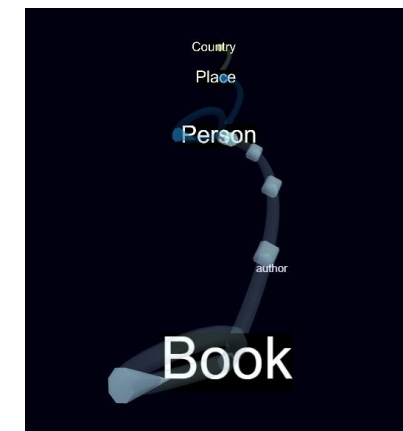
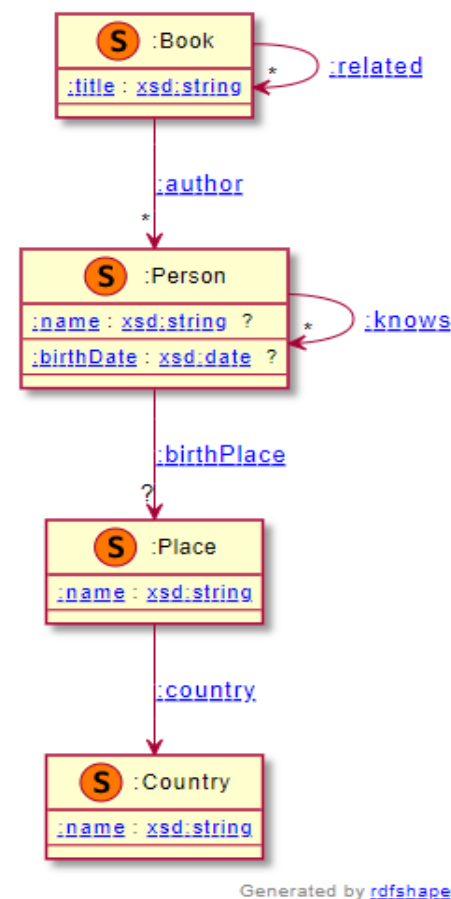
Implemented in [rdfshape](#) and [rudof](#)

Edit the data models using UML editors
(XML prototype)

HTML pages

Partial visualizations to improve usability
browsing large data models

3D prototype visualization of shapes schemas



Continuous integration with Shapes

Coexistence between ontologies/shapes

Shapes can validate the behaviour of inference systems

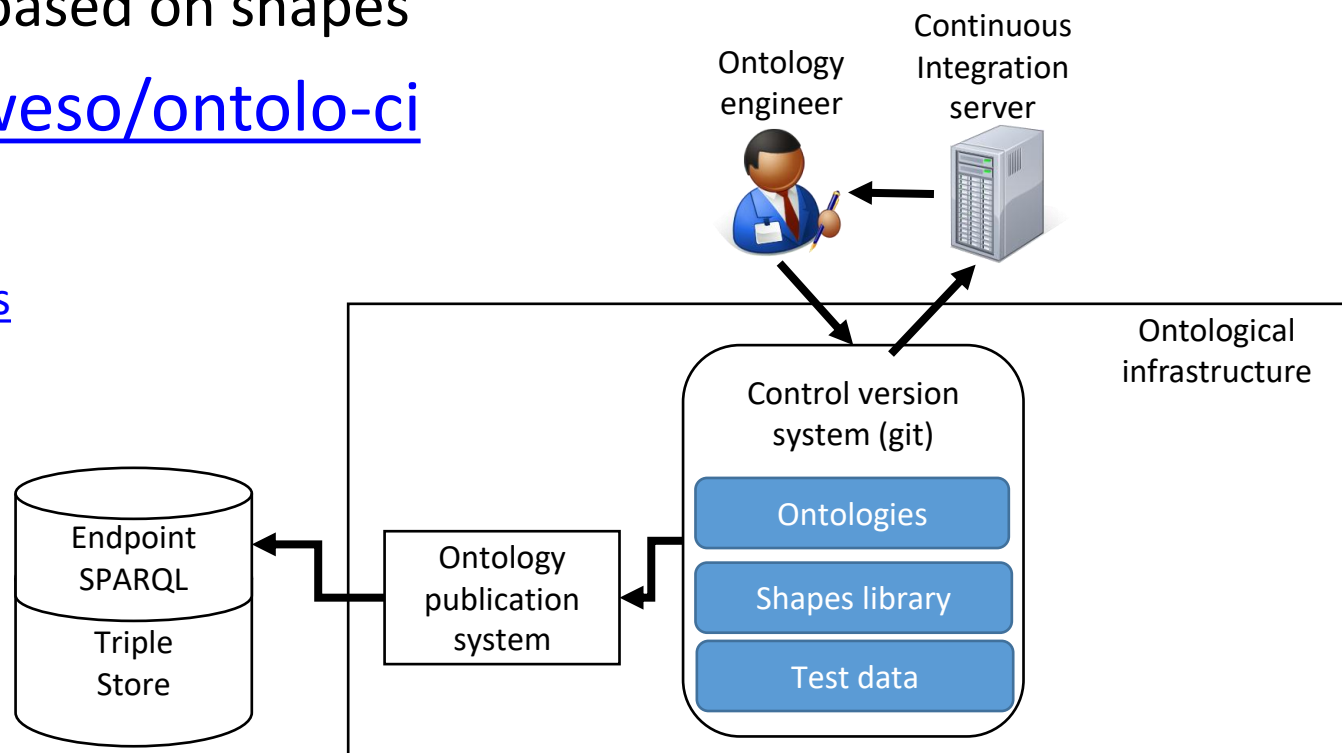
Shapes pre- and post- inference

TDD and continuous integration based on shapes

Ontolo-ci: <https://github.com/weso/ontolo-ci>

Gene Ontology Shapes:

<https://github.com/geneontology/go-shapes>



Continuous integration with Shapes

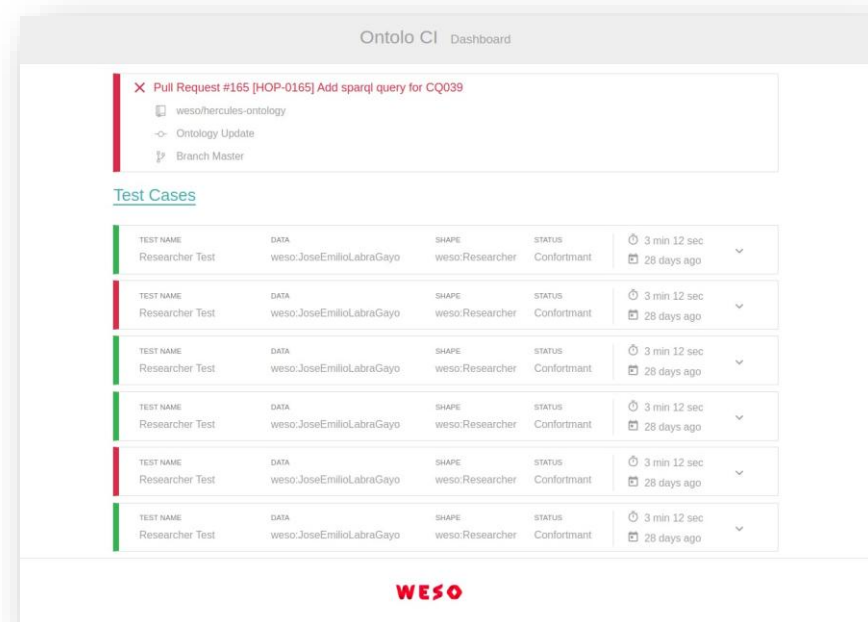
Ontolo-ci: <https://github.com/weso/ontolo-ci>

Developed as part of HERCULES-Ontology

Test-Driven-Development applied to Ontologies

Input:

- Ontologies
- Shapes
- Test data
- Input shape map (SPARQL competency question)
- Expected result shape map



Creating shapes

Shapes editors

- Text-based editors

- Visual editors and visualizers

Obtaining shapes from...

- Spreadsheets

- RDF data

- Ontologies

- Other schemas (XML Schema)



Text-based editors

YaSHE: Forked from YASGUI: <http://www.weso.es/YASHE/>

Syntax highlighting

Auto-completion



The screenshot displays the YaSHE text-based editor interface. The editor window shows a SPARQL query with syntax highlighting. The query is as follows:

```
1 PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
2 prefix wd: <http://www.wikidata.org/entity/>
3 prefix wdt: <http://www.wikidata.org/prop/direct/>
4
5 # Example SPARQL query: select ?researcher where { ?researcher wdt:P106 wd:Q1650915 } limit 5
6
7 <Researcher> EXTRA wdt:P31 wdt:P106 {
8   wdt:P31 [ wd:Q5 ] ; # Instance of = human
9   wdt:P106 [ wd:Q1650915 ] ; # Occupation = researcher
10  wdt:P101 @<Discipline> * ; # Field of work
11  wdt:P496 xsd:string ? ; # ORCID-ID
12  wdt:P1153 xsd:string ? ; # Scopus-Author ID
13 }
14
```

Auto-completion is shown for the property `wdt:P1153` on line 12. The dropdown menu lists the following options:

- Scopus Author ID (P1153)
- identifier for an author assigned in Scopus bibliographic database

The editor interface includes a toolbar with icons for undo, redo, save, and other standard text editor functions. The query is displayed with line numbers on the left side of the editor window.

Shapes author tools: Top Braid Composer

Form based editor

Integrated with Top Braid product

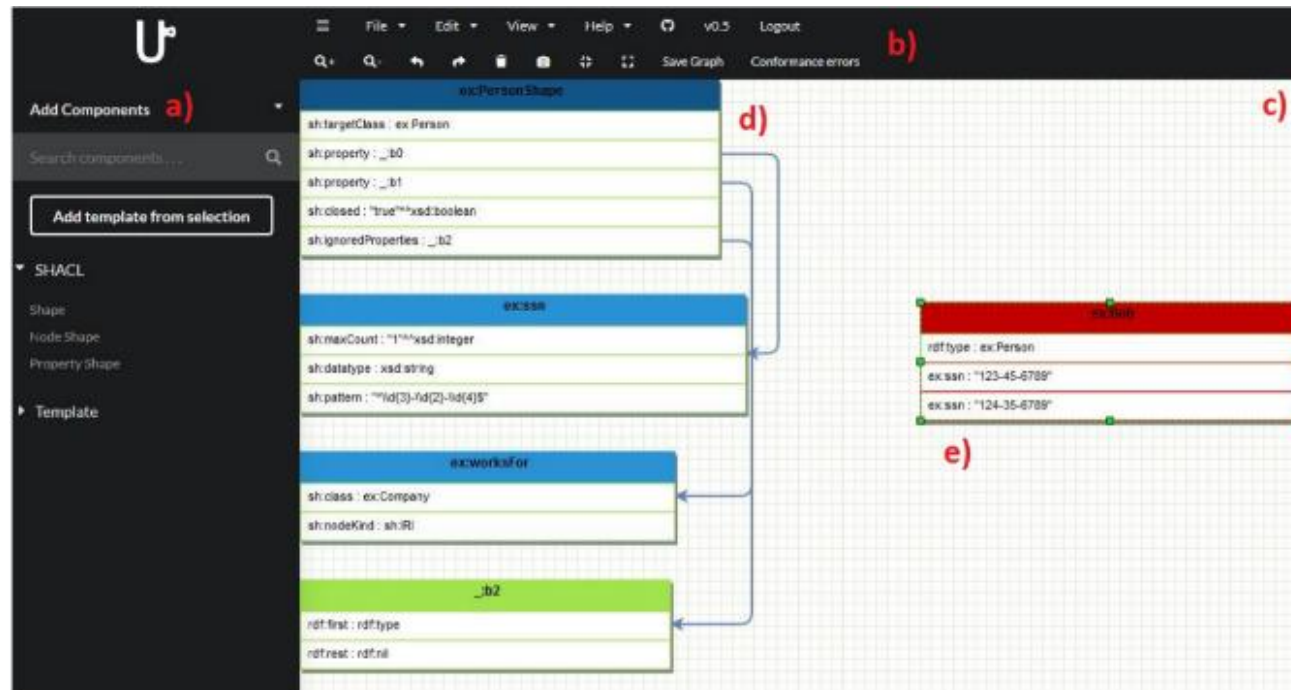
The screenshot displays the 'Node Shape Form' for a 'Person' class in the Top Braid Composer. The main window shows a tree view on the left with categories: Annotations, Constraints, and Targets. The 'Person' class is selected, and its properties are listed. A modal dialog titled 'Create sh:property for Person' is open, allowing the user to define a new property. The dialog fields are as follows:

- Predicate:** knows
- ☐ Also globally declare rdf:Property
- Display Name:** name
- Description:** Name of a person
- Count:** Unlimited [0..*]
- Node kind:** Literals
- Class:** (empty dropdown)
- Datatype:** xsd:string
- Value shape:** (empty dropdown)

At the bottom of the dialog are 'OK' and 'Cancel' buttons. The background window shows the 'Form' tab selected, and the URL bar at the bottom indicates the current context is 'http://datashapes.org/dash (owl:Imports from /topBraid/SHACL/dash.ttl)'.

Shapes author tools: UnSHACLed

Visual SHACL Editor in Javascript



B. De Meester, P. Heyvaert, A. Dimou, and R. Verborgh, "Towards a Uniform User Interface for Editing Data Shapes," in Proceedings of the 4th International Workshop on Visualization and Interaction for Ontologies and Linked Data, 2018, vol. 2187.

Shapes author tools: ShEx Author

ShEx-Author: Inspired by Wikidata Query Service

2 column: Visual one synchronized with text based

The screenshot displays the ShExAuthor web application interface. The top navigation bar includes the logo, the name "ShExAuthor", and links for "YASHE" and "About me". A "Fork me on GitHub" badge is visible in the top right corner of the interface.

The interface is divided into two main columns:

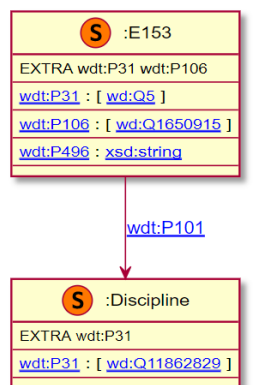
- Visual Editor (Left Column):** Labeled "Assistant" in the top left. It features a sidebar with icons for various actions. The main area shows two "Shape" definitions. The first shape, "Researcher", is defined with five triples using the "wdt" property and values "P31", "P106", "P101", "P496", and "P1153". The second shape, "Discipline", is defined with one triple using the "wdt" property and value "P31". Each triple is represented by a "Prefix" dropdown, a "Property" dropdown, and a "Value" input field, with buttons for adding and deleting triples.
- Text Editor (Right Column):** Displays the corresponding ShEx text representation of the shapes. The text is as follows:

```
1 PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
2 PREFIX wd: <http://www.wikidata.org/entity/>
3 PREFIX wdt: <http://www.wikidata.org/prop/direct/>
4
5 <Researcher> {
6   wdt:P31      IRI ;
7   wdt:P106     IRI ? ;
8   wdt:P101     @<Discipline>? ;
9   wdt:P496     xsd:string ? ;
10  wdt:P1153    xsd:string * ;
11 }
12
13 <Discipline> {
14   wdt:P31      IRI * ;
15 }
16
17
```

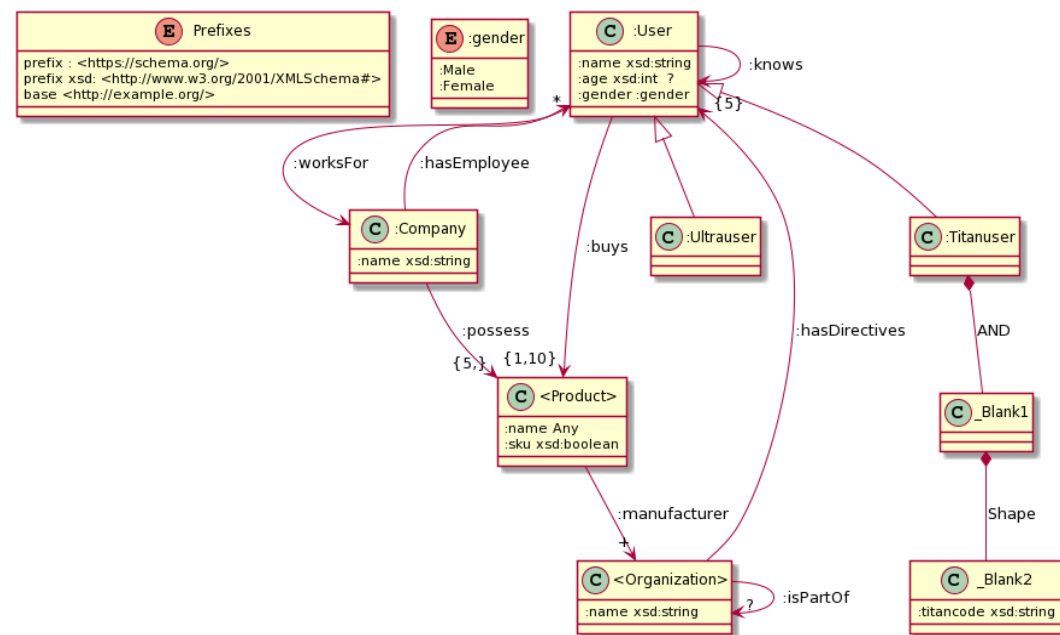
Shapes visualization

Integrated in RDFShape/Wikishape

- [UMLSHacLEX](#) UML diagrams for ShEx
- [ShUMLex](#): Conversion to UML through XMI



Generated by [rdfshape](#)



Shapes from spreadsheets

SKOS-Play was used at ELI to generate SHACL shapes from Excel

ShExstatements: <https://shexstatements.toolforge.org/>

ShExCSV: CSV representation of Shapes

Hermes: ShExCSV processor, <https://github.com/weso/hermes>

Creating data models using spreadsheets

[DCTAP](#) data models (templates in XLSX, CSV), recently added to rudof



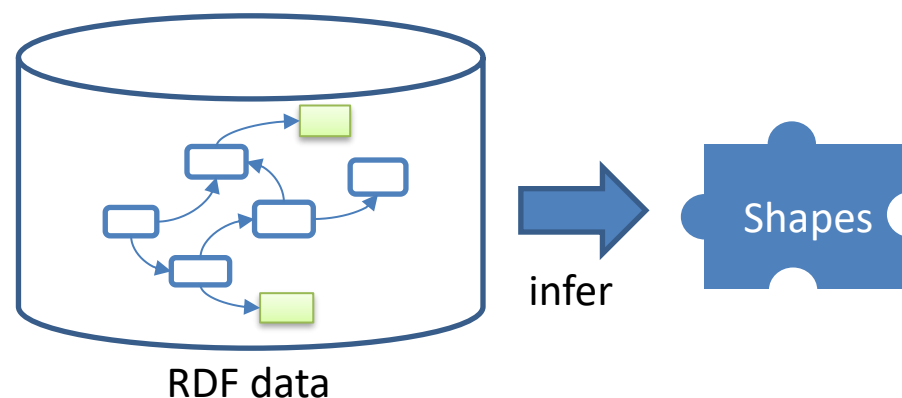
Generating Shapes from RDF data

Useful use case in practice

Some prototypes

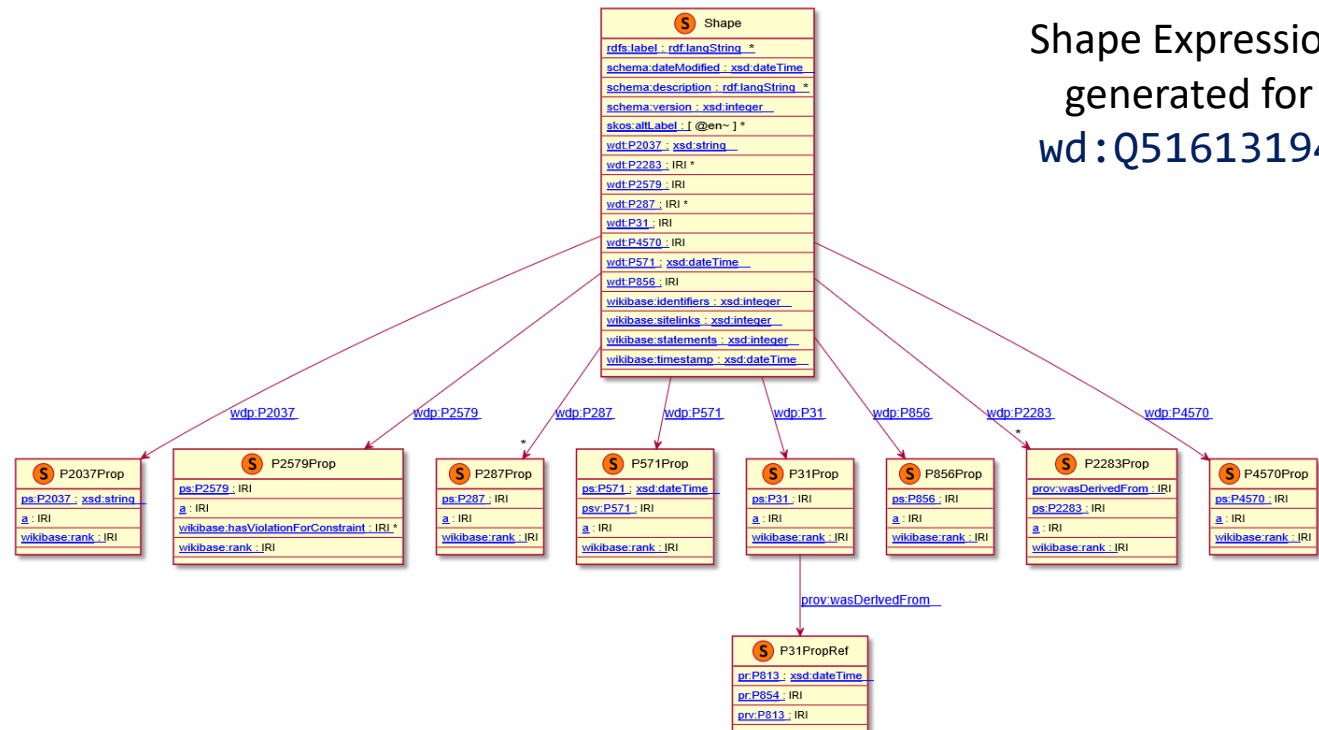
RDFShape: <http://rdfshape.weso.es>

sheXer: <http://shexer.weso.es/>



Shapes from data: RDFShape

RDFShape/Wikishape implement a basic prototype to derive Shapes from RDF data



Shapes from data: sheXer

sheXer: <https://github.com/DaniFdezAlvarez/sheXer>

Implemented as a Python library

Supports Shapes generation from large SPARQL endpoints

- Can generate shapes from sampling

ShEx consolidator can be used for large RDF data

Application example:
Creating Knowledge Graphs subsets

Some current problems...

Large knowledge graphs are difficult to handle

- SPARQL queries require computational resources

- Endpoints usually impose limits (timeouts...)

Contents are continually evolving

- Results of SPARQL queries now may be different later

- Research based on large KGs difficult to be reproducible

Some applications for KG subsets (1)

Performance

- Enable SPARQL queries that don't work with whole KG

Data integration

- Create domain specific subsets and integrate with other data

Reproducibility

- Snapshots of KG contents that can be cited & reused

Some applications for KGs subsets (2)

Transformation and enrichment

- Add and integrate content from different KGs

KG in your pocket

- Create mobile apps based on some subset

KG analysis

- Create and compare subsets from historical dumps

License combination

- Combine subsets from KG with some license (CC0) with others

Source: Wikidata as a knowledge graph for the life sciences, A. Waagmeester et al, <https://elifesciences.org/articles/52614>

Reusing turned out to be difficult

Example: counting number of genes and associated elements:

- Proteins
- Chromosomes
- Disease
- Taxons

Wikidata Query Service

Examples

Help

More tools

Query Builder

1

PREFIX wd: <http://www.wikidata.org/entity/>

2

PREFIX wdt: <http://www.wikidata.org/prop/direct/>

3

4

SELECT (COUNT(?gene) AS ?count_gene) ?count_P688_protein ?count_P1057_chromosome ?count_P2293_disease ?count_P703_taxon

5

WHERE

6

{ ?gene wdt:P31 wd:Q7187

7

{ SELECT (COUNT(?y) AS ?count_P688_protein)

8

WHERE

9

{ ?x wdt:P31 wd:Q7187 ;

10

wdt:P688 ?y .

11

?y wdt:P31 wd:Q8054

12

}

13

}

14

{ SELECT (COUNT(?y) AS ?count_P1057_chromosome)

15

WHERE

16

{ ?x wdt:P31 wd:Q7187 ;

17

wdt:P1057 ?y .

18

?y wdt:P31 wd:Q37748

19

}

20

}

21

{ SELECT (COUNT(?y) AS ?count_P2293_disease)

22

WHERE

23

{ ?x wd

24

wd

25

?y wd

26

}

27

}

28

{ SELECT ((

29

WHERE

30

{ ?x wd

31

wd

32

?y wd

33

}

34

}

35

}

36

GROUP BY ?count_

Query timeout limit reached

SPARQL-QUERY: queryStr=PREFIX wd: <http://www.wikidata.org/entity/>

PREFIX wdt: <http://www.wikidata.org/prop/direct/>

SELECT (COUNT(?gene) AS ?count_gene) ?count_P688_protein ?count_P1057_chromosome ?count_P2293_disease ?count_P703_taxon

WHERE

{ ?gene wdt:P31 wd:Q7187

{ SELECT (COUNT(?y) AS ?count_P688_protein)

WHERE

{ ?x wdt:P31 wd:Q7187 ;

wdt:P688 ?y .

?y wdt:P31 wd:Q8054

}

}

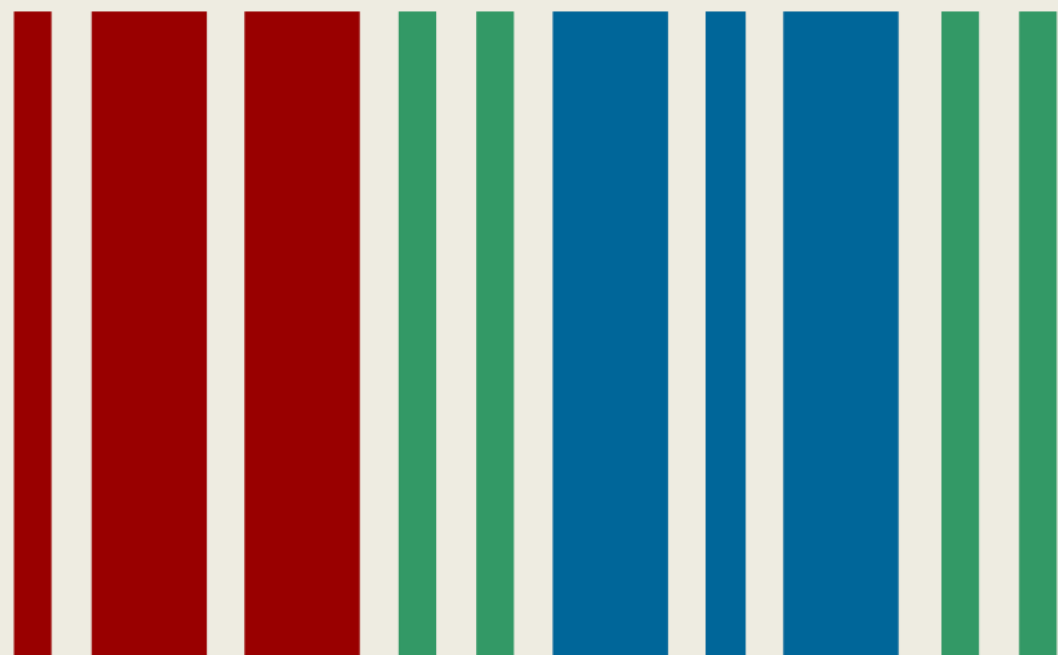
Wikidata subsetting efforts

Collaborative research driven by a practical need

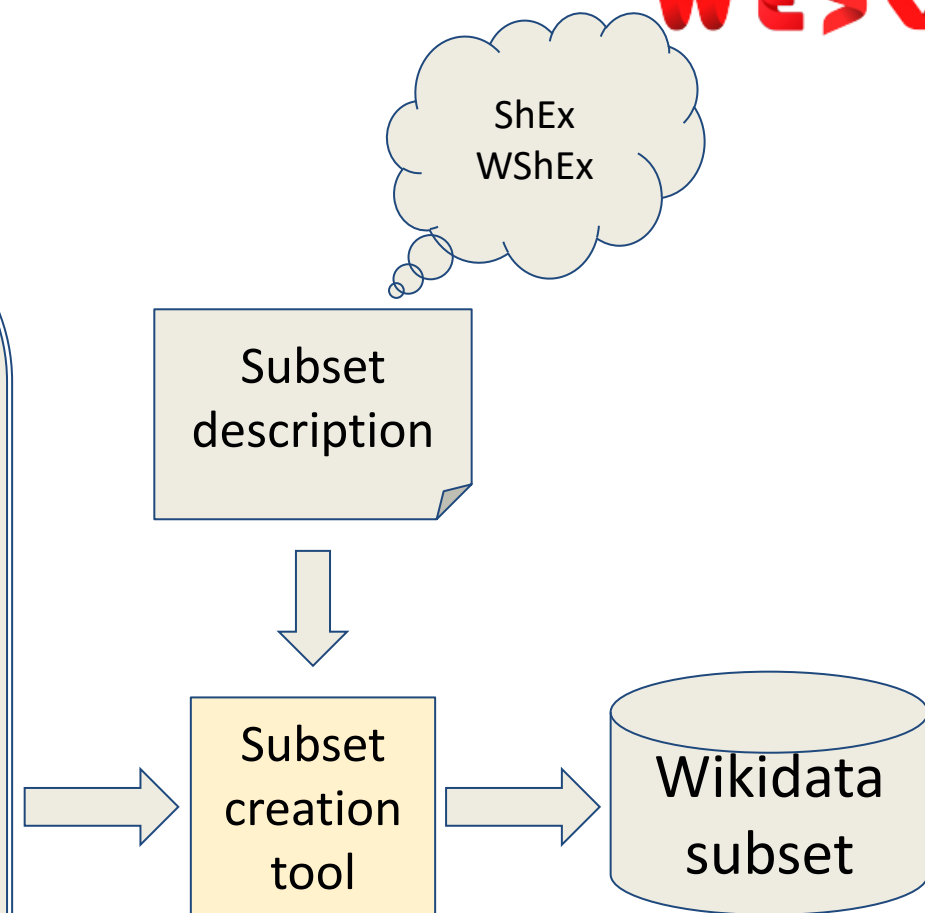
Most of the advances were triggered by SWAT4HCLS and Biohackathons

- 12th SWAT4HCLS conference, 2019. [Wikidata:WikiProject Schemas/Subsetting - Wikidata](#)
- Europe Biohackathon, 2020, [project 35](#) [[preprint](#)]
- Europe Biohackathon, 2021, [project 21](#)
- Europe Biohackathon, 2022 [project 11](#), [[preprint](#)]
- Japan Biohackathon, 2023 [[preprint](#)]
- 2023, Paper: [Wikidata subsetting: approaches, tools and evaluation](#), accepted at Semantic Web Journal

Problem statement



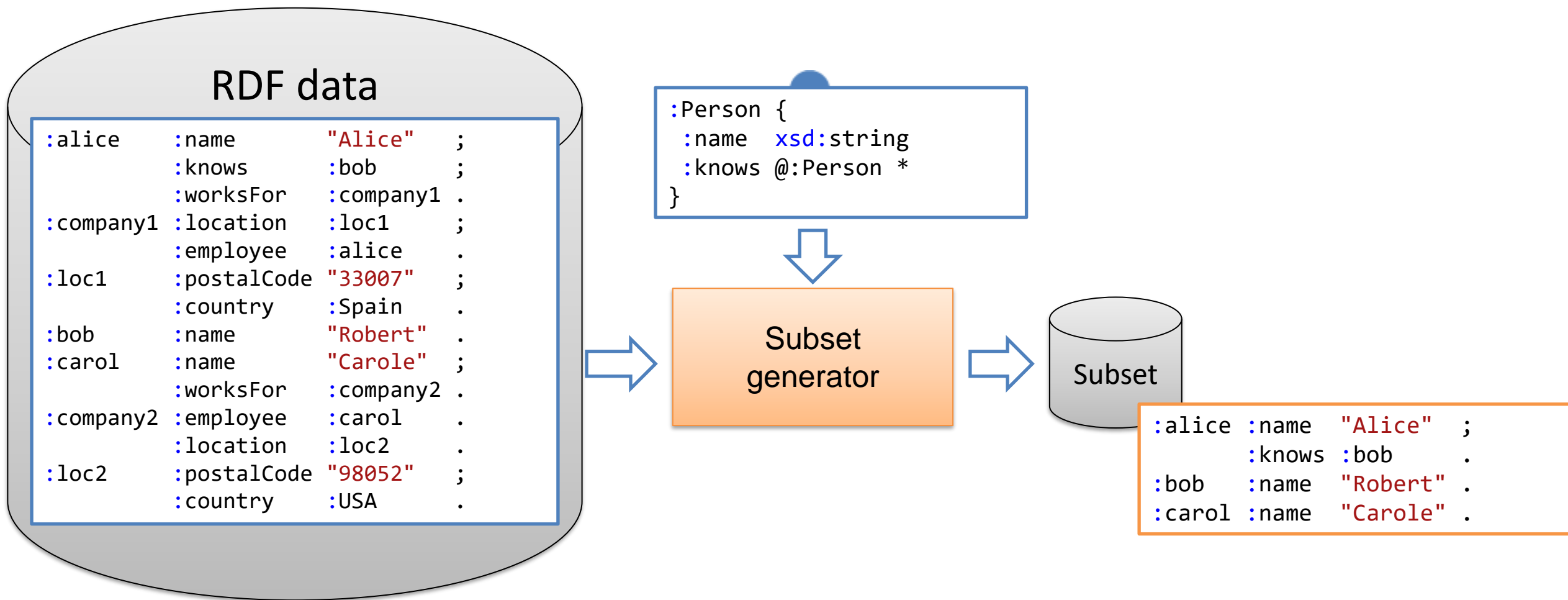
WIKIDATA



Subsetting based on ShEx

Generate subsets from ShEx

ShEx describes the contents of the expected subset



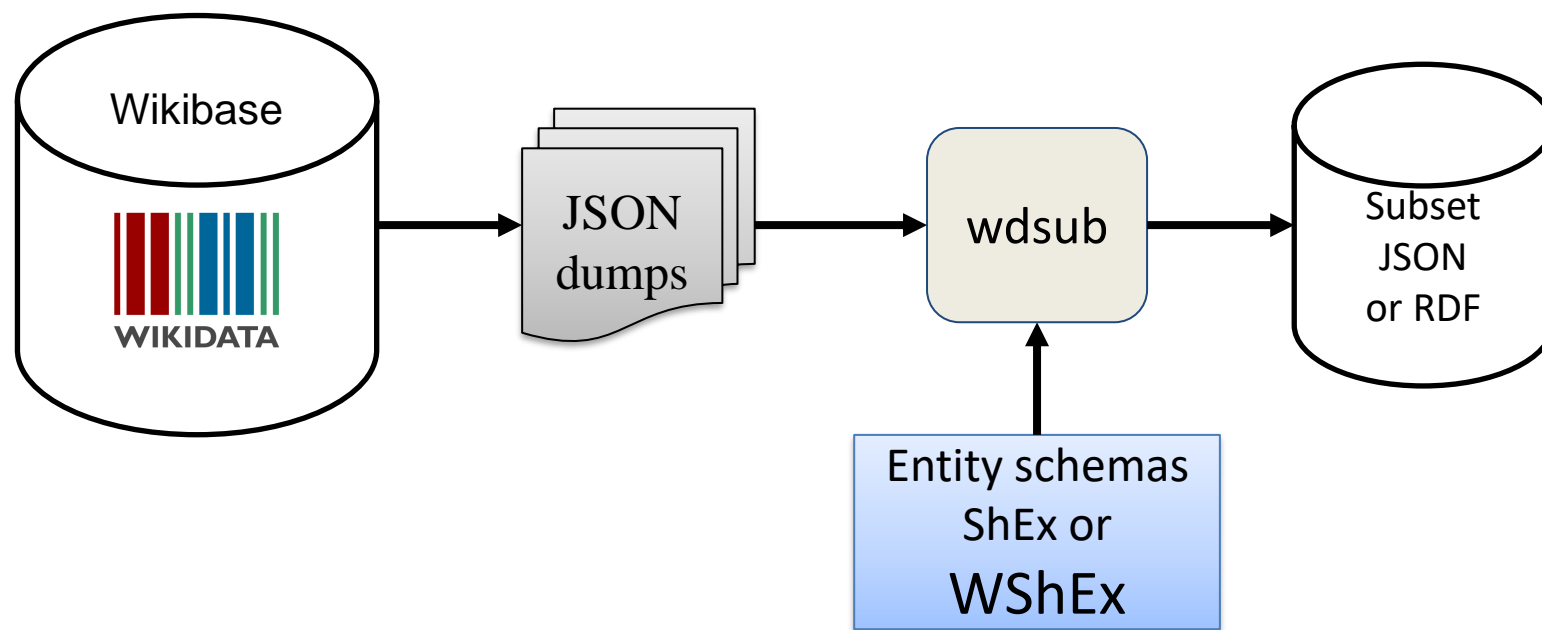
wdsub

Input:

- ShEx/WShEx schema
- Wikidata dumps in JSON

Output

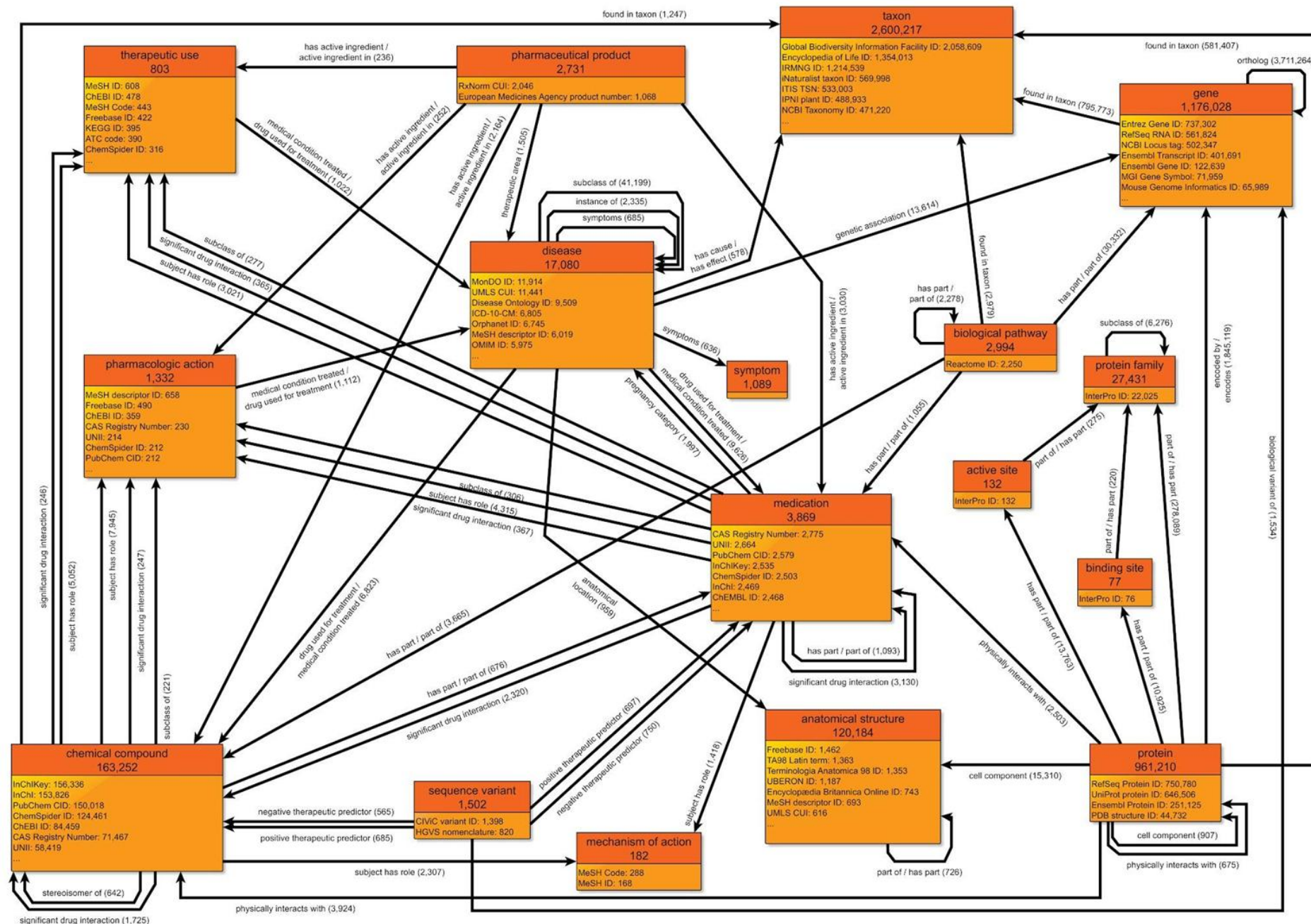
- Dumps in JSON/RDF



Link: <https://github.com/weso/wdsub>

Docker (CLI): [wesogroup/wdsub:0.0.32](https://github.com/weso/wdsub)

Data model



GeneWiki subset

GeneWiki experiments

- [ShEx schema \(E258\)](#)

```

start= @:active_site OR
       @:anatomical_structure OR
       . . .
       @:gene OR
       . . .

:active_site EXTRA wdt:P31 {
  rdfs:label [ @en ] ;
  wdt:P31 [ wd:Q423026 ] ;
  wdt:P361 @:protein_family * ;
  wdt:P527 @:protein_family * ;
}
. . .
:gene EXTRA wdt:P31 {
  rdfs:label [ @en ] ;
  wdt:P31 [ wd:Q7187 ] ;
  wdt:P684 @:gene * ; # ortholog (P684)
  wdt:P2293 @:disease * ; # genetic association (P2293)
  wdt:P703 @:taxon * ; # found in taxon (P703)
  wdt:P1057 @:chromosome * ; # chromosome (P1057)
  wdt:P682 @:biological_process * ; # biological process (P682)
  wdt:P688 @:protein * ; # encodes (P688)
}
. . .

```

Results about GeneWiki experiment

| Class | 2015 | 2016 | 2017 | 2018 | 2019 | 2020 | 2021 | 2022 | Wikidata |
|--------------------------|---------|---------|---------|---------|---------|---------|---------|---------|-----------|
| active_site | 0 | 0 | 132 | 132 | 132 | 132 | 132 | 132 | 132 |
| anatomical_structure | 4 | 62 | 470 | 483 | 614 | 732 | 738 | 812 | 746 |
| binding_site | 0 | 0 | 76 | 76 | 76 | 77 | 77 | 77 | 76 |
| biological_pathway | 0 | 0 | 425 | 2754 | 2929 | 3279 | 3429 | 3486 | 3554 |
| biological_process | 11 | 12 | 31263 | 31222 | 42058 | 43417 | 42061 | 41857 | 42449 |
| cellular_component | 1 | 1 | 4017 | 4081 | 4239 | 4298 | 4137 | 4139 | 4211 |
| chemical_compound | 19144 | 21128 | 156718 | 157018 | 157685 | 1050488 | 1201719 | 1245041 | 1249719 |
| chromosome | 0 | 0 | 149 | 152 | 432 | 9167 | 9224 | 9223 | 9224 |
| disease | 124 | 931 | 9578 | 9926 | 11439 | 13197 | 5395 | 5607 | 5698 |
| gene | 17 | 20 | 679372 | 677836 | 811574 | 1196193 | 1196334 | 1211506 | Timed-Out |
| medication | 46 | 2127 | 2459 | 2472 | 2699 | 3210 | 3336 | 3424 | 3450 |
| molecular_function | 0 | 0 | 9413 | 9801 | 11258 | 11226 | 10940 | 10898 | 11246 |
| pharmaceutical_product | 0 | 0 | 1067 | 1067 | 2725 | 2754 | 2759 | 2774 | 2784 |
| protein_domain | 2 | 3 | 9581 | 8847 | 9348 | 10770 | 11274 | 11709 | 11736 |
| protein_family | 0 | 212 | 20912 | 20632 | 22240 | 22170 | 23277 | 24204 | 24266 |
| protein | 118 | 166 | 450785 | 487781 | 579979 | 980520 | 985755 | 988099 | Timed-Out |
| sequence_variant | 0 | 0 | 1411 | 918 | 774 | 724 | 695 | 686 | 686 |
| supersecondary_structure | 0 | 0 | 687 | 687 | 688 | 688 | 694 | 696 | 696 |
| symptom | 16 | 235 | 273 | 283 | 328 | 366 | 319 | 335 | 343 |
| taxon | 1920049 | 2121404 | 2213907 | 2318731 | 2492613 | 2769303 | 2929068 | 3478871 | 3491430 |

Other examples

The paper contains other other studies about

Multilingual extraction

```
start= @:gene OR
       @:taxon

:gene EXTRA wdt:P31 {
  rdfs:label      [ @en @es @fa @nl ] ;
  schema:description [ @en @es @fa @nl ] ;
  skos:altLabel    [ @en @es @fa @nl ] * ;
  wdt:P31          [ wd:Q7187 ] ;
  wdt:P703         @:taxon * ;
}

:taxon EXTRA wdt:P31 {
  rdfs:label      [ @en @es @fa @nl ] ;
  schema:description [ @en @es @fa @nl ] ;
  skos:altLabel    [ @en @es @fa @nl ] * ;
  wdt:P31          [ wd:Q16521 ] ;
}
```

Qualifiers and references

```
. . .
<gene> EXTRA wdt:P31 {
  rdfs:label [ @en ] ? ;
  wdt:P31 [ wd:Q7187 ] * ;      # is instance of (P31) gene (Q7187)
  wdt:P351 . ? ;                # has one or no Entrez Gene ID (P351)
  p:P2293 {
    ps:P2293 @<disease> + ;
    prov:wasDerivedFrom @<References> + ;
  } + ;
}
. . .
```

Subsets based on Pregel algorithm

Pregel algorithm = parallel algorithm proposed at Google

“think like a vertex”

2 prototype implementations (work in progress)

- Scala: [SparkWDSUB](#)
 - We were able to create subsets in 36 minutes with large cluster (512 cores)
- Rust: [pregel-rs](#) based on Polars and
 - It can process Wikidata-dumps
 - Recently applied to large RDF data: UniProt

More information at [paper](#):

- [Creating Knowledge Graphs Subsets using Shape Expressions](#), J. Labra, arXiv:2110.11709
- [Using Pregel to create Knowledge Graphs subsets described by non-recursive Shape Expressions](#), A. Préstamo, J. Labra, accepted at KGSWC' 23

UIs and shapes

Shapes can provide hints to generate user interfaces/forms

SHACL core defines a basic vocabulary: sh:group, sh:order, ...

ShEx annotations can also be used to define UI declarations

Example: UI ontology annotations

UIs and Shapes: ShExPath and ShEx-Forms

ShEx Path can be used to point to parts of a ShEx schema

<https://shexspec.github.io/spec/ShExPath>

ShEx generated forms demo based on UI ontology:

<https://ericprud.github.io/shex-form/?manifestURL=examples/manifest.json>

UIs and shapes: TopQuadrant

Form generation from SHACL

DASH vocabulary:

<http://datashapes.org/forms.html>

The screenshot shows a web form titled "Holger's Address" with a tabbed interface. The active tab is "Holger's Address", and there are buttons for "Explore", "Modify", "Cancel", "Preview", and "Save Changes". A dropdown menu shows "Australian address shape". The form is organized into sections: "Address" and "Contact".

Address Section:

- street address:** 3 Teewah Close
- suburb:** Kewarra Beach
- state:** QLD
- postal code:** 48791

Contact Section:

- email:** holger@knublauch.com, holger@topquadrant.com
- phone number:** (empty field)

Generating code from shapes

Domain model based on Shapes

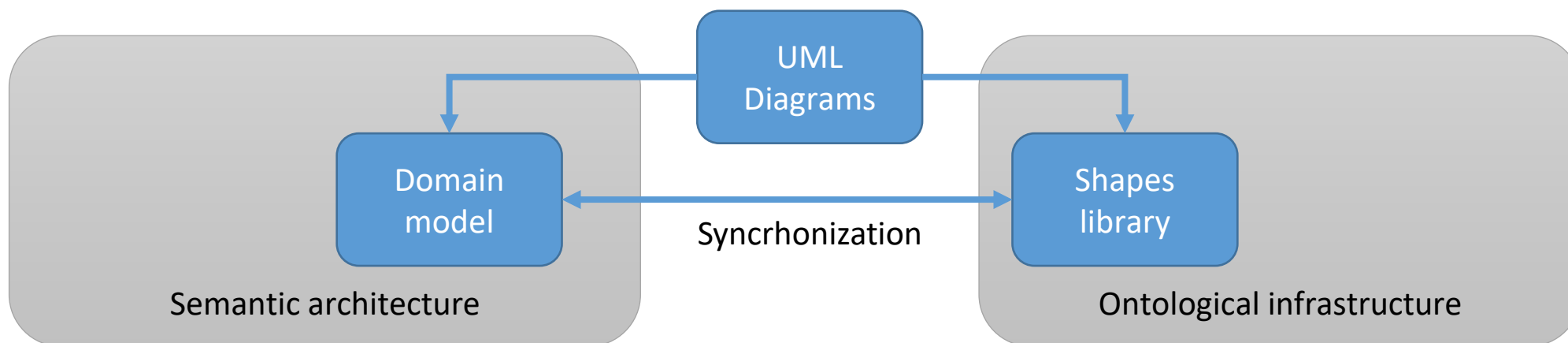
Clean architecture pattern

Domain model as central element

Simple classes (POJO): Plain Old Java Objects

Shapes synchronization

Application logic and services based on domain model

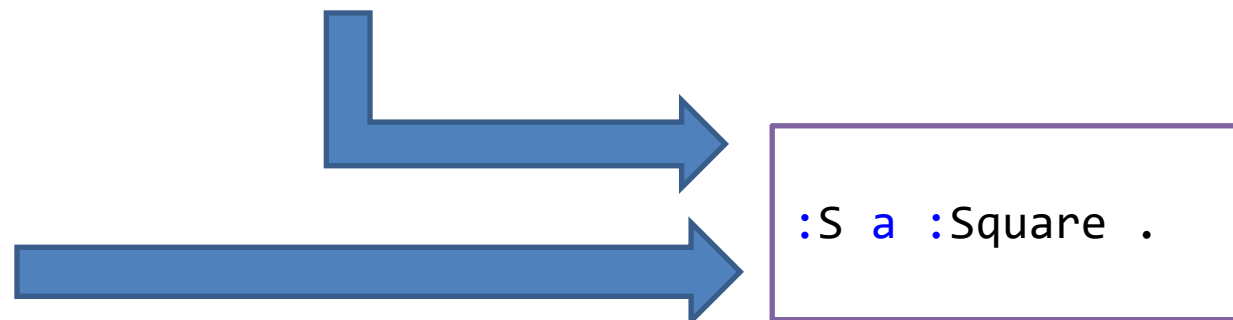


Shapes and rules

SHACL Advanced Features describes SHACL rules

```
:Rectangle a rdfs:Class, sh:NodeShape ;  
  rdfs:label "Rectangle" ;  
  sh:property [ sh:path :height ;  
    sh:datatype xsd:integer ;  
    sh:maxCount 1 ; sh:minCount 1 ;  
    sh:name "height" ] ;  
  sh:property [ sh:path :width ;  
    sh:datatype xsd:integer ;  
    sh:maxCount 1 ; sh:minCount 1 ;  
    sh:name "width" ; ] ;  
  sh:rule [ a sh:TripleRule ;  
    sh:subject sh:this ;  
    sh:predicate rdf:type ;  
    sh:object :Square ;  
    sh:condition :Rectangle ;  
    sh:condition [  
      sh:property [  
        sh:path :width ;  
        sh>equals :height ;  
      ] ; ] ; ] .
```

```
:I a :Rectangle .  
:N a :Rectangle ;  
  :height 2 ;  
  :width 3 .  
:S a :Rectangle ;  
  :height 4 ;  
  :width 4 .
```



Shapes ecosystems

Wikidata provides a whole ShEx ecosystem

Entity schemas can evolve and relate between each other

Directory: https://www.wikidata.org/wiki/Wikidata:Database_reports/EntitySchema_directory

Different schemas for the same entities?

Some schemas stress some aspects while others stress others

Evolution of schemas

Searching entity schemas

Conclusions

Shapes can be a very important aspect of Knowledge Graphs

Graph data is flexible and shapes also

Shapes ecosystems with:

- Prescriptive shapes

- Descriptive shapes

- Suggestive shapes (can *suggest* properties to employ)

Domain experts are a key aspect of this

END OF PRESENTATION

Acknowledgments

Awesome Semantic Shapes:

<https://github.com/w3c-cg/awesome-semantic-shapes>

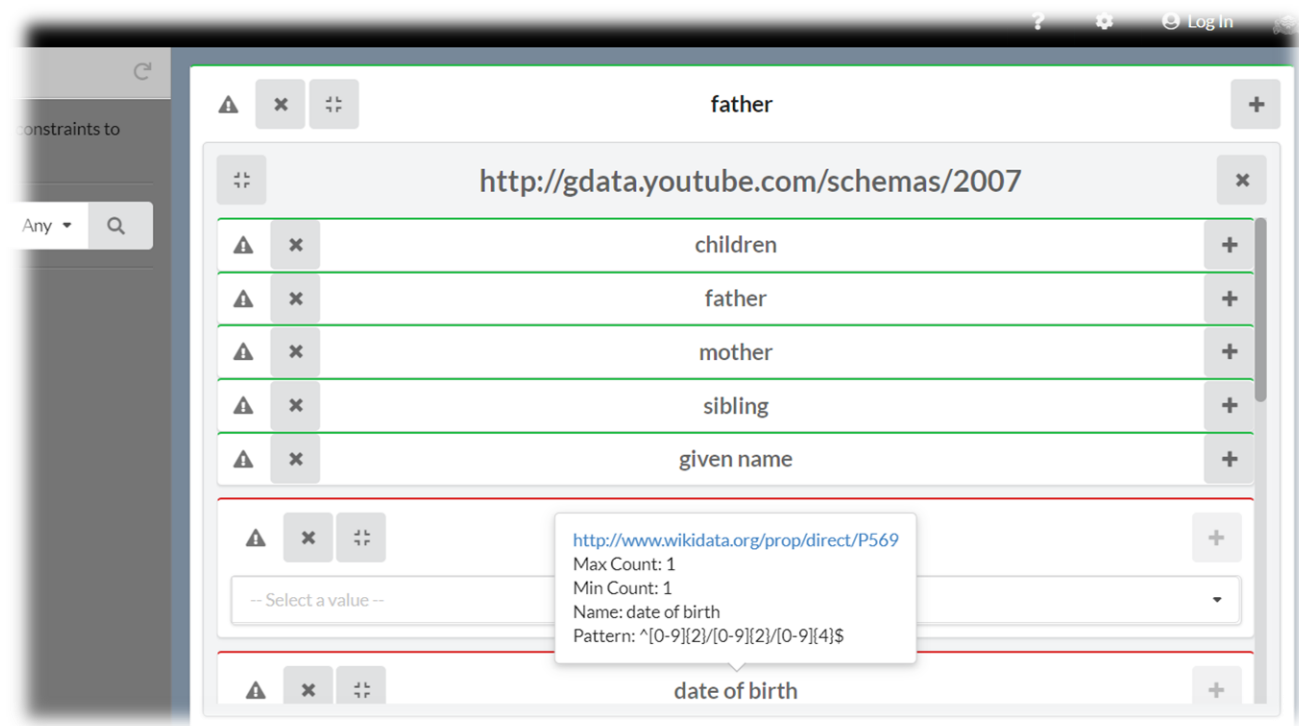
Special thanks to Vladimir Alexiev for starting it

People from ShEx community group: Tom Baker, Kat Thornton,
Andra Waagmeester,...

UIs and shapes: Schímatos

<http://schimatos.org/>

It will be presented at ISWC20

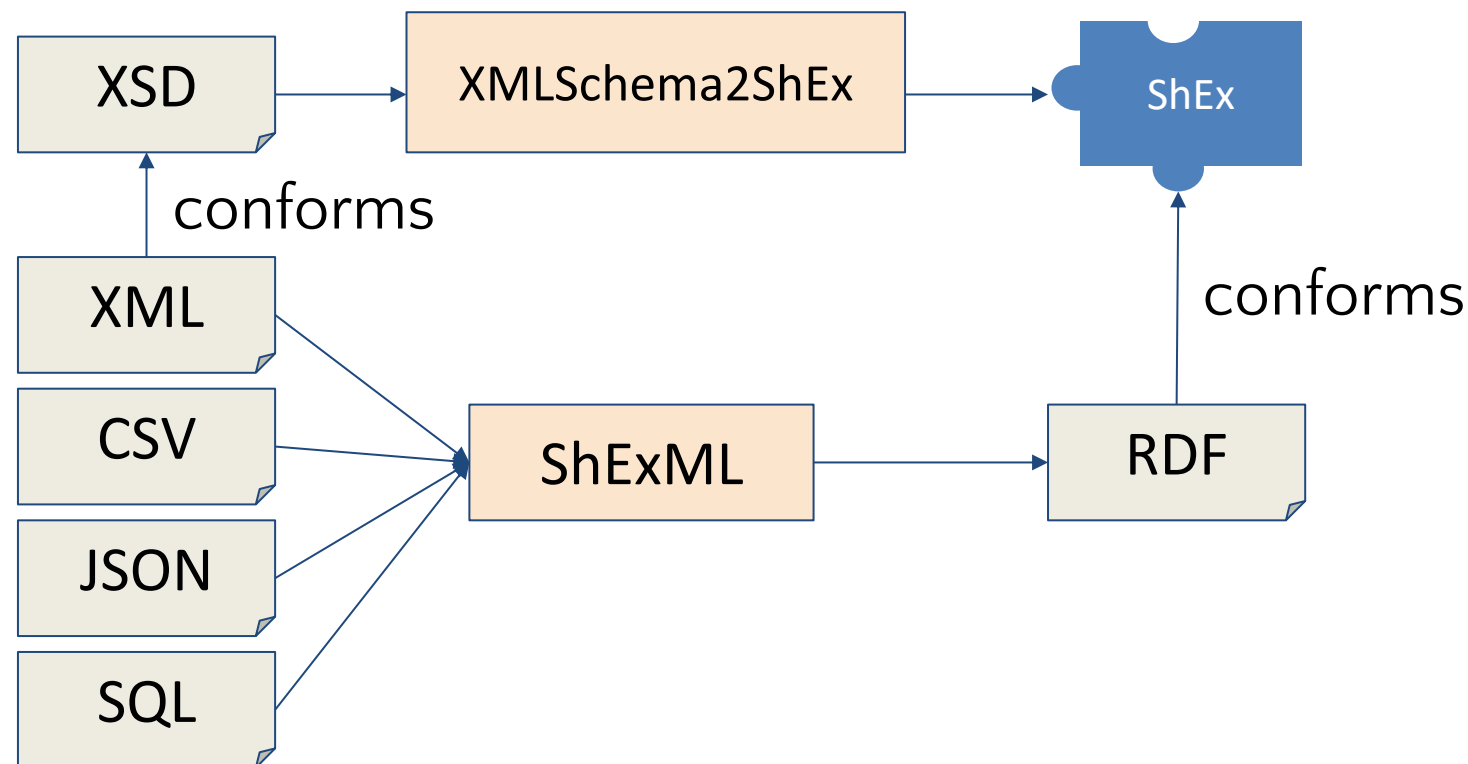


Shapes for data integration

[XMLSchema2ShEx](#): Convert XML Schemas to shapes

[ShExML](#): Domain specific language to convert data to RDF

Input formats: CSV, XML, JSON, SQL



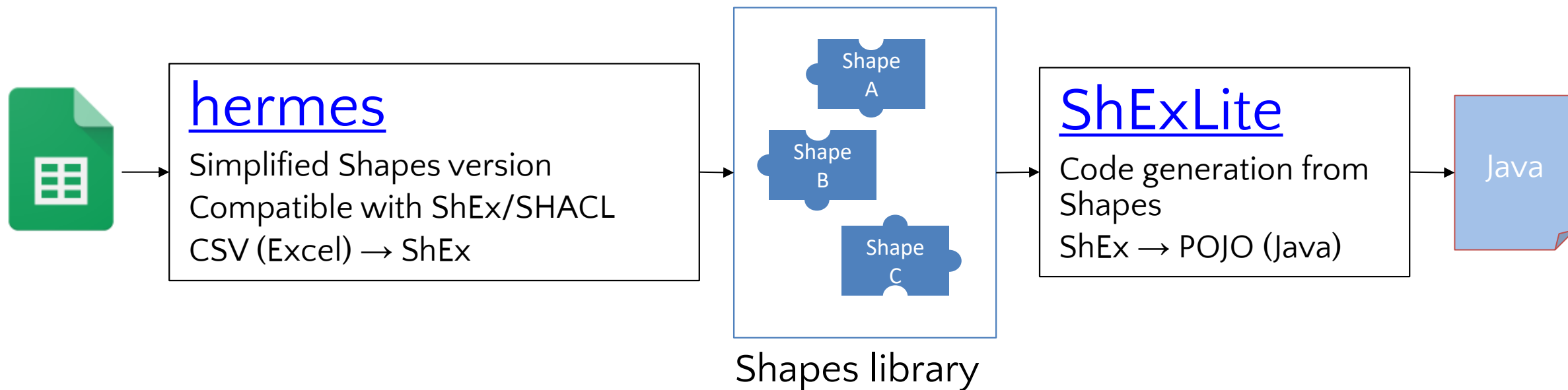
Generating code from shapes

Generate domain model from shapes

Entities (pseudo-shapes) defined with Excel (Google spreadsheets)

Shapes generation from those templates

Java code generation (POJOs) from those shapes



Shapes from data: ShapeDesigner

<https://gitlab.inria.fr/jdusart/shexjapp>

The screenshot displays the ShapeDesigner application interface. The top menu bar includes File, Schema, Graph, Pattern, Query, and Help. Below the menu, there are tabs for Conception and ShEx Validation. The main workspace is divided into three sections: Patterns, Queries, and Schema.

Patterns Section:

| Name | Description |
|--|--|
| Wikidata direct properties | { wdt:~ __; } |
| Wikidata properties and their qualifiers | { p:P~ {ps:~ __; psn:~ __; a[__]; } }; |
| All properties | { p:P~ {ps:~ __; psn:~ __; a[__]; } wdt:~ __; ~ __ } |
| provenance for population | { p:P1082 {ps:~ __; pq:~ __; prov:~ __ } } |

Buttons: Delete, Duplicate, Add, Edit

Queries Section:

| Name | Description |
|----------------------------------|---|
| Select all nodes with given type | SELECT ?x WHERE { ?x wdt:P31 <replace-this-by-iri-of-...> } |
| Select 10 big city | SELECT ?x WHERE { ?x wdt:P31 wd:Q1549591. } LIMIT 10 |

Buttons: Delete, Duplicate, Add, Edit

Schema Section:

```
Schema:
1 PREFIX wdt: <http://www.wikidata.org/entity/statement/>
2 PREFIX ontolex: <http://www.w3.org/ns/lemon/ontolex#>
3 PREFIX wdata: <http://www.wikidata.org/wiki/Special:EntityData/>
4 PREFIX p: <http://www.wikidata.org/prop/>
5 PREFIX wd: <http://www.wikidata.org/entity/>
6 PREFIX wdno: <http://www.wikidata.org/prop/novalue/>
7 PREFIX wdref: <http://www.wikidata.org/reference/>
8 PREFIX ps: <http://www.wikidata.org/prop/statement/>
9 PREFIX pq: <http://www.wikidata.org/prop/qualifier/>
10 PREFIX cc: <http://creativecommons.org/ns#>
11 PREFIX wdt: <http://www.wikidata.org/prop/direct/>
12 PREFIX wdv: <http://www.wikidata.org/value/>
13 PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
14 PREFIX dct: <http://purl.org/dc/terms/>
15 PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
16 PREFIX prov: <http://www.w3.org/ns/prov#>
17 PREFIX wdtm: <http://www.wikidata.org/prop/direct-normalized/>
18 PREFIX pqv: <http://www.wikidata.org/prop/qualifier/value/>
19 PREFIX prv: <http://www.wikidata.org/prop/reference/value/>
20 PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
21 PREFIX psv: <http://www.wikidata.org/prop/statement/value/>
22 PREFIX skos: <http://www.w3.org/2004/02/skos/core#>
23 PREFIX geo: <http://www.opengis.net/ont/geosparql#>
24 PREFIX bd: <http://www.bigdata.com/rdf#>
25 PREFIX pqn: <http://www.wikidata.org/prop/qualifier/value-normalized/>
26 PREFIX pr: <http://www.wikidata.org/prop/reference/>
27 PREFIX prn: <http://www.wikidata.org/prop/reference/value-normalized/>
28 PREFIX psn: <http://www.wikidata.org/prop/statement/value-normalized/>
29 PREFIX schema: <http://schema.org/>
30 PREFIX wikibase: <http://wikiba.se/ontology#>
31 PREFIX demo: <http://inria.fr/ShapeDesigner/demo>

32
33
34
35 # Result for pattern { p:P1082 {ps:~ __; pq:~ __; prov:~ __ } } and query SELECT ?x WHERE { ?x wdt:P31 wd:Q1549591. } LIMIT 10
36 demo:BigCityPopulation {
37   # population; frequency: [10/10]
38   p:P1082 {
39     # population; frequency: [99/99]
40     psv:P1082 [ <http://www.wikidata.org/value~> ] ;
41     # population; frequency: [99/99]
42     ps:P1082 xsd:decimal ;
43     # point in time; frequency: [99/99]
44     pqv:P585 [ <http://www.wikidata.org/value~> ] + ;
45     # point in time; frequency: [99/99]
46     pq:P585 xsd:dateTime + ;
47     prov:wasDerivedFrom [ <http://www.wikidata.org/reference~> ] * ;
48     # determination method; frequency: [28/99]
49     pq:P459 [ <http://www.wikidata.org/entity~> ] ? ;
50     # publisher; frequency: [1/99]
51     pq:P123 [ <http://www.wikidata.org/entity~> ] ?
52   } +
53 }
```

Other uses of Shapes

UIs and shapes

Generating code from Shapes

Shapes and rules

Data portals

In 2013, at WESO, we were hired to develop some data portals

Examples: WebIndex (Web Foundation)

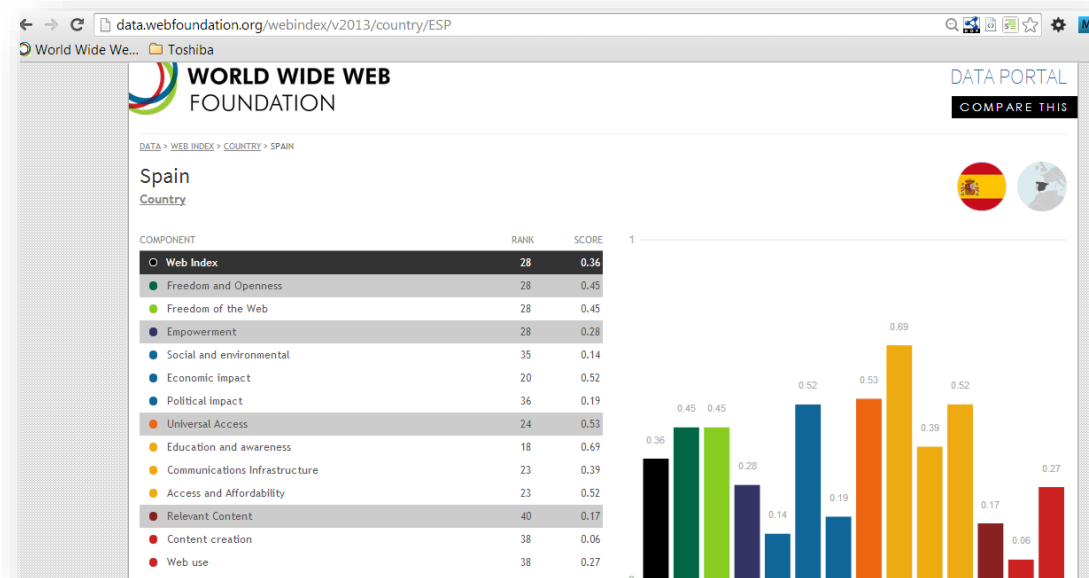
One of the first applications of ShEx

Measure WWW's contribution to development and human rights by country

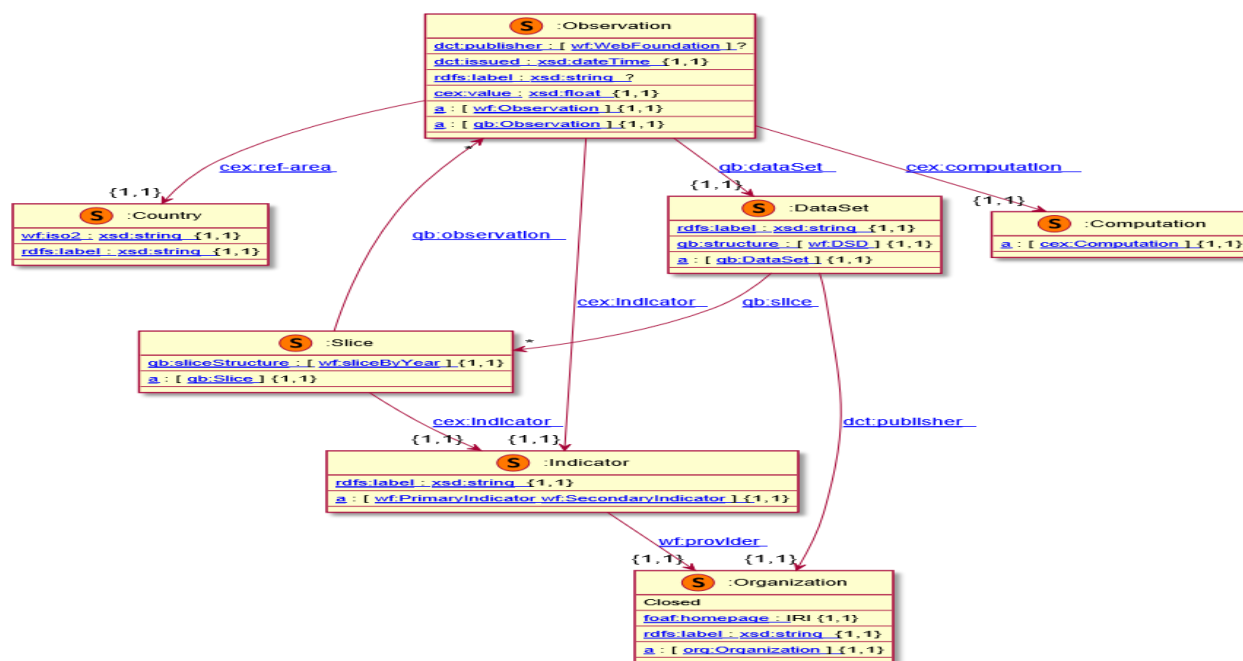
Developed by the Web Foundation

Content of data portal = statistical observations

We employed RDF Data Cube vocabulary (qb:Observation)



Simplified WebIndex data model



Lessons learnt from ShEx usage at WebIndex

1. Documentation of linked data portal

Human-readable, machine processable

<http://weso.github.io/wiDoc>

2. Team communication

Communicate the developers which shapes they had to generate

3. Validation

For example: check if a value of type `qb:Observation` had shape `<Observation>`

4. Reuse

Another data portal was later developed for <http://landportal.org> base on observations

Easy to reuse and adapt the data model

Same types (`qb:Observation`) but different structure

<http://weso.github.io/landportalDoc/data/>

Wikidata and wikibase



In May, 2019, Wikidata announced ShEx adoption

New namespace for schemas

Example:

<https://www.wikidata.org/wiki/EntitySchema:E2>

Wikibase also contains entity schemas

Online demo: [wikishape](#)

EntitySchema Discussion Read View history Search devwiki

Disease (E2)

| language code | label | description | aliases | edit |
|---------------|---------|--|-----------------------|----------------------|
| en | Disease | Shape Expression for diseases coming from Disease Ontology in Wikidata (Source: https://github.com/andrawaag/Genewiki-ShEx/blob/master/diseases/Wikidata-Disease-Ontology.shex) | Sickness Illness | edit |



Solid project

SOLID (SOcial Linked Data): Promoted by Tim Berners-Lee

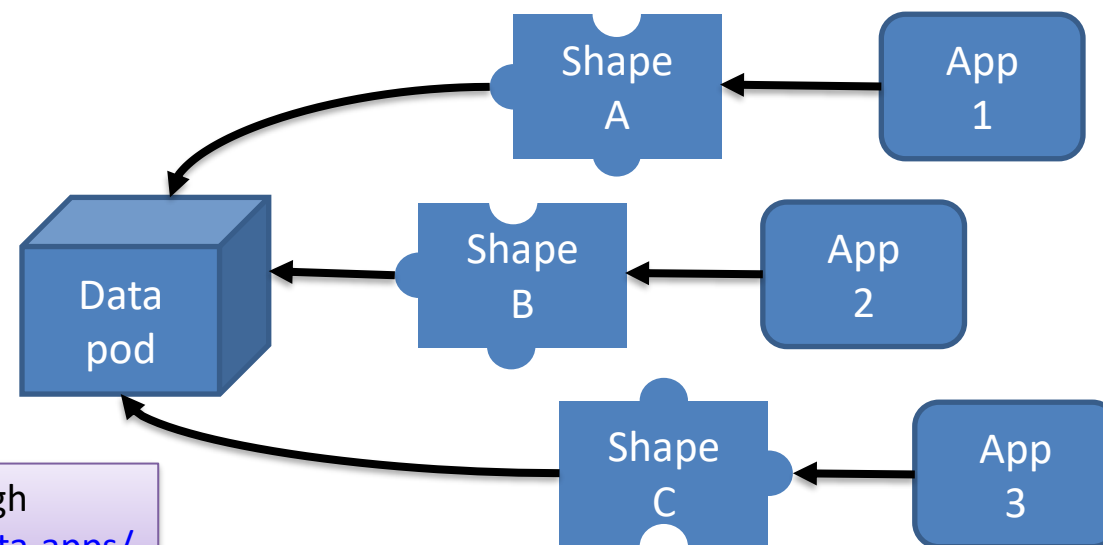
Goal: Re-decentralize the Web

- Separate data from apps

- Give users more control about their data

- Internally using linked data & RDF

Shapes needed for interoperability



"...I just can't stop thinking about shapes.", Ruben Verborgh

<https://ruben.verborgh.org/blog/2019/06/17/shaping-linked-data-apps/>

Other use cases

HL7 FHIR.

Example: <https://www.hl7.org/fhir/observation.html>

ELI validator

SHACL shapes obtained from Excel sheets:

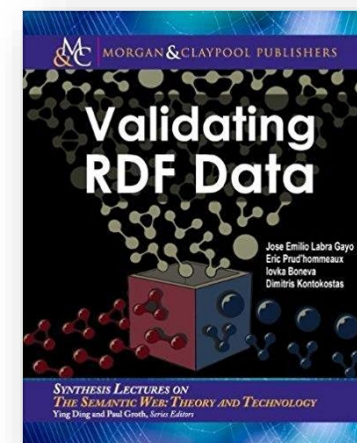
<https://webgate.ec.europa.eu/eli-validator/home>

SHACL adoption supported by Top Quadrant

See: <https://www.topquadrant.com/technology/shacl/>

More info:

Chapter 6 of Validating RDF data: <http://book.validatingrdf.com/bookHtml012.html>



Tools: challenges and perspectives

Validating with shapes

Obtaining shapes

Other applications of shapes

Shapes ecosystems

Validating with shapes

Libraries and command line validators

Online demos

Integrated in ontology editors

Continuous integration with Shapes