

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-Star

 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

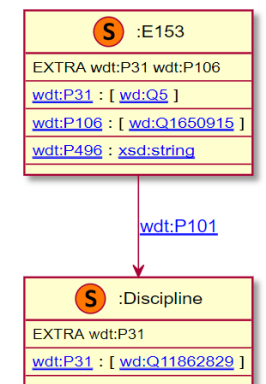
New implementation in rudof

XMI-based prototype allowed to edit the data models using
UML editors

HTML pages

Partial visualizations to improve usability browsing large
data models

3D prototype visualization of shapes schemas



Generated by [rdfshape](#)

Continuous integration with Shapes

Coexistence between ontologies/shapes

Shapes can validate the behaviour of inference systems

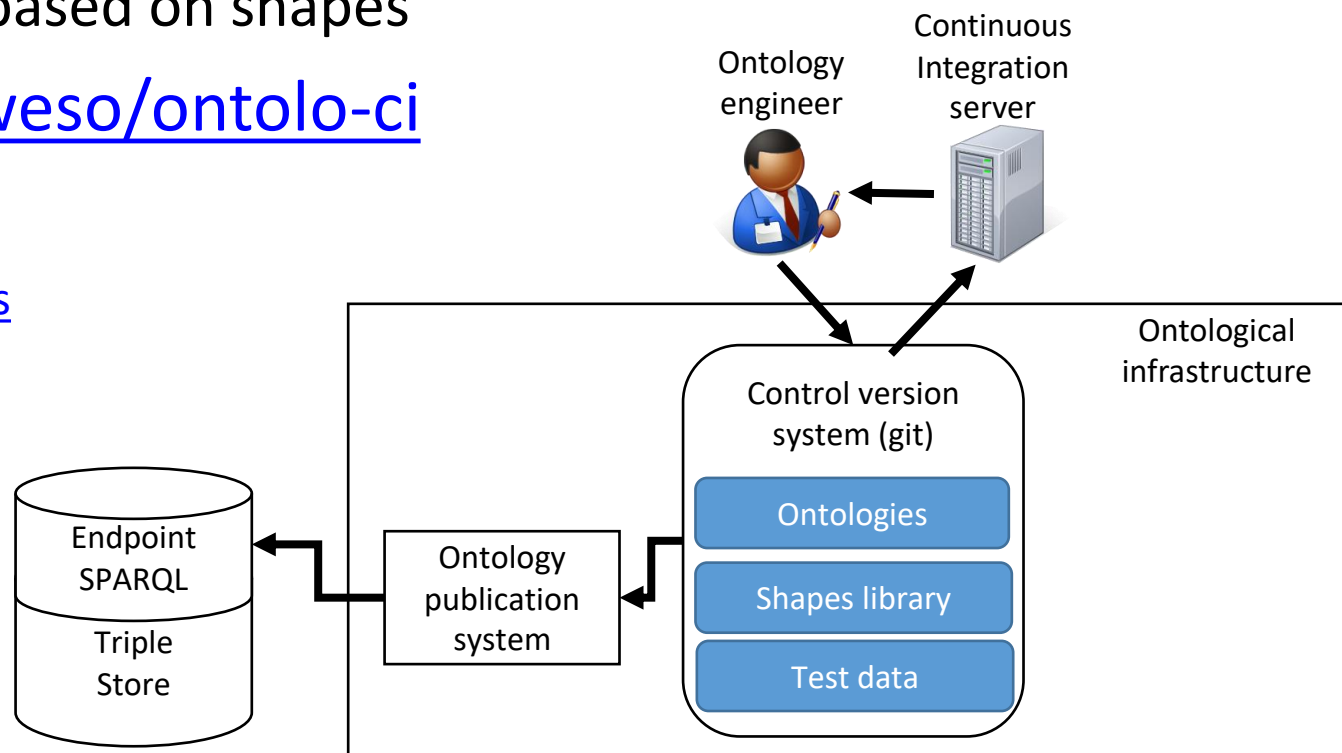
Shapes pre- and post- inference

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

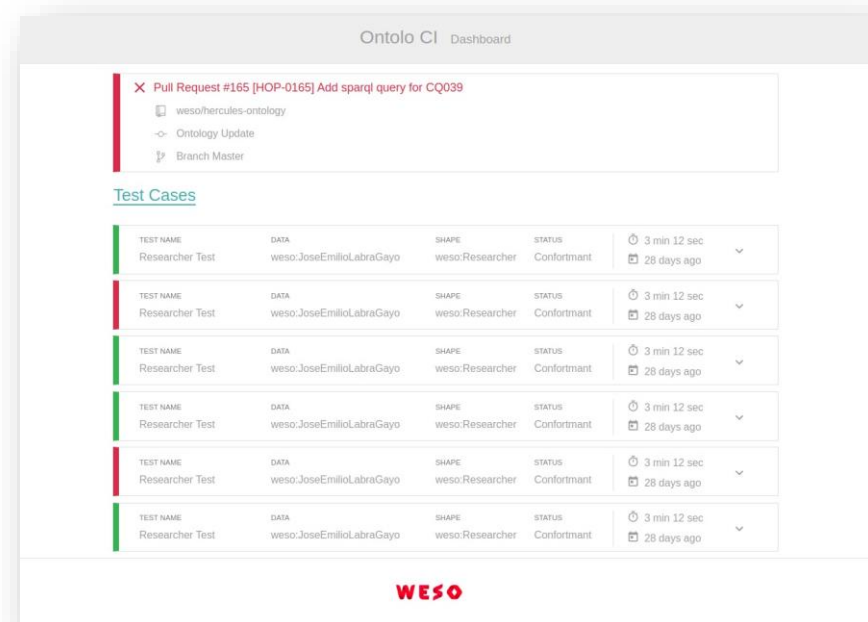
Ontolo-ci: <https://github.com/weso/ontolo-ci>

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`. 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.

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 dialog box titled 'Create sh:property for Person' is open, allowing the user to define a new property. The dialog includes fields for the predicate ('knows'), a checkbox for 'Also globally declare rdf:Property', a display name ('name'), a description ('Name of a person'), a count ('Unlimited [0..*]'), a node kind ('Literals'), a class (empty), a datatype ('xsd:string'), and a value shape (empty). The dialog has 'OK' and 'Cancel' buttons at the bottom.

*UserBook.shapes.ttl

Node Shape Form

Name: Person

- Annotations
- Constraints
 - sh:property
 - sh:sparql
- Targets
 - sh:targetClass
 - sh:targetObjectsOf
 - sh:targetSubjectsOf
 - sh:targetNode
 - sh:target
 - sh:deactivated
- Other Properties
 - dash:abstract
 - dash:applicableToClass
 - dash:closedByTypes
 - dash:defaultViewForRole
 - dash:resourceAction

Form | Browser | Diagram | Graph

Imports | Instances

http://datashapes.org/dash (owl:imports from /topBraid/SHACL/dash.ttl)

Create sh:property for Person

Predicate: knows

☐ Also globally declare rdf:Property

Display Name: name

Description: Name of a person

Count: Unlimited [0..*]

Node kind: Literals

Class:

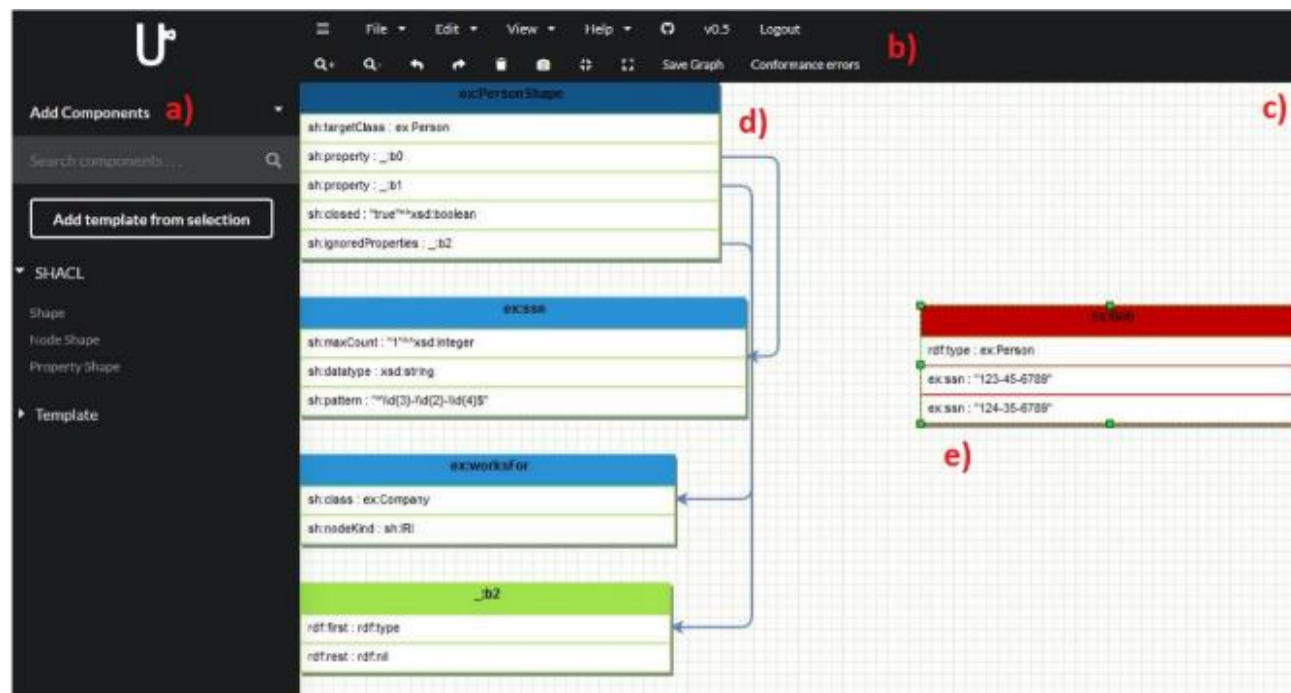
Datatype: xsd:string

Value shape:

OK Cancel

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, a hamburger menu, and links for 'YASHE' and 'About me'. A 'Fork me on GitHub' badge is visible in the top right corner of the interface.

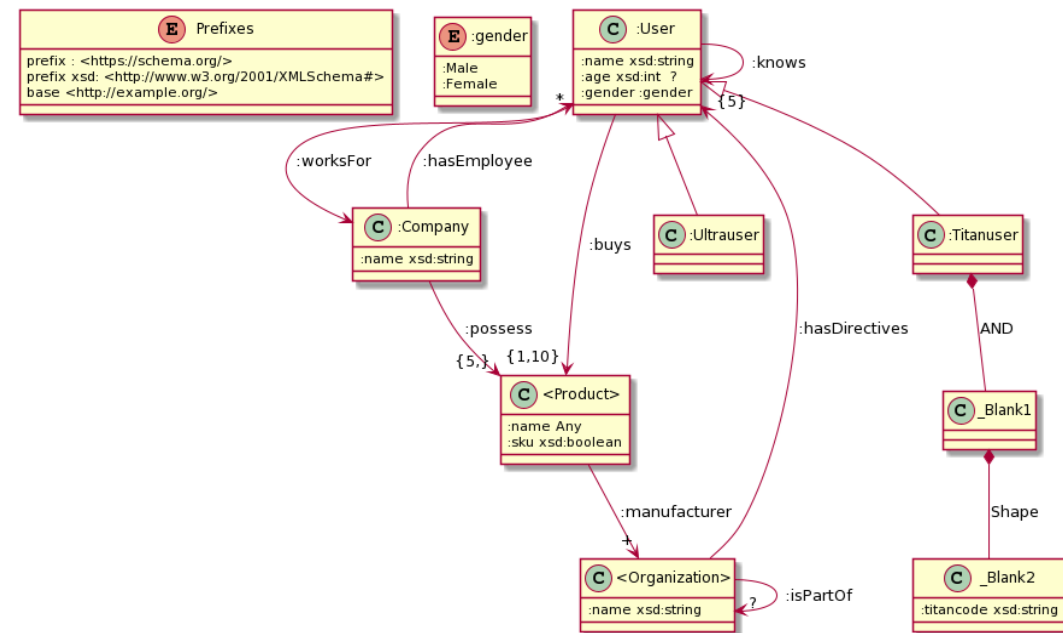
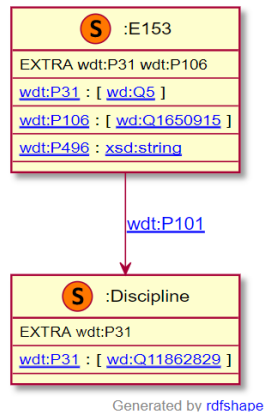
The interface is divided into two main columns:

- Visual Editor (Left Column):** Titled 'Assistant', it features a sidebar with icons for information, view, prefixes, undo, redo, save, download, and delete. The main area shows two 'Shape' definitions. The first shape, 'Researcher', is defined with the IRIRef 'Researcher' and five triples: (Prefix, wdt, P31), (Prefix, wdt, P106), (Prefix, wdt, P101), (Prefix, wdt, P496), and (Prefix, wdt, P1153). The second shape, 'Discipline', is defined with the IRIRef 'Discipline' and one triple: (Prefix, wdt, P31). Each triple is shown in a table with columns for Prefix, property (wdt), and value. Buttons for adding and deleting triples are present.
- Text Editor (Right Column):** Displays the corresponding ShEx text representation. The prefixes are: `PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>`, `PREFIX wd: <http://www.wikidata.org/entity/>`, and `PREFIX wdt: <http://www.wikidata.org/prop/direct/>`. The shapes are defined as: `<Researcher> { wdt:P31 IRI ; wdt:P106 IRI ? ; wdt:P101 @<Discipline>? ; wdt:P496 xsd:string ? ; wdt:P1153 xsd:string * ; }` and `<Discipline> { wdt:P31 IRI * ; }`.

Shapes visualization

Integrated in RDFShape/Wikishape

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



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



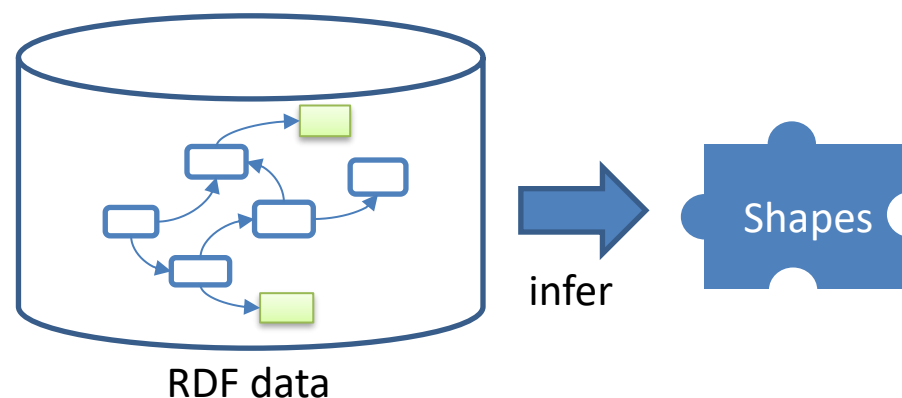
Generating Shapes from RDF data

Useful use case in practice

Some prototypes

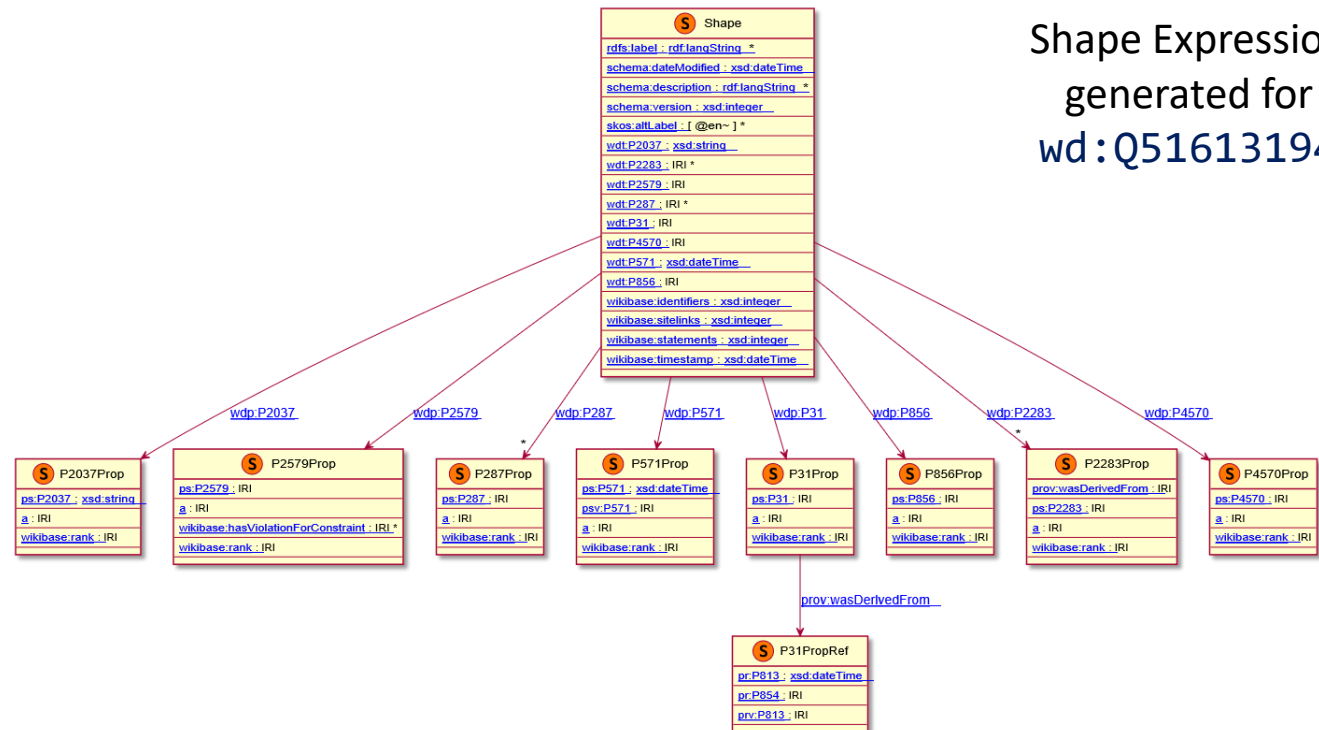
RDFShape: <http://rdfshape.weso.es>

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

Why 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

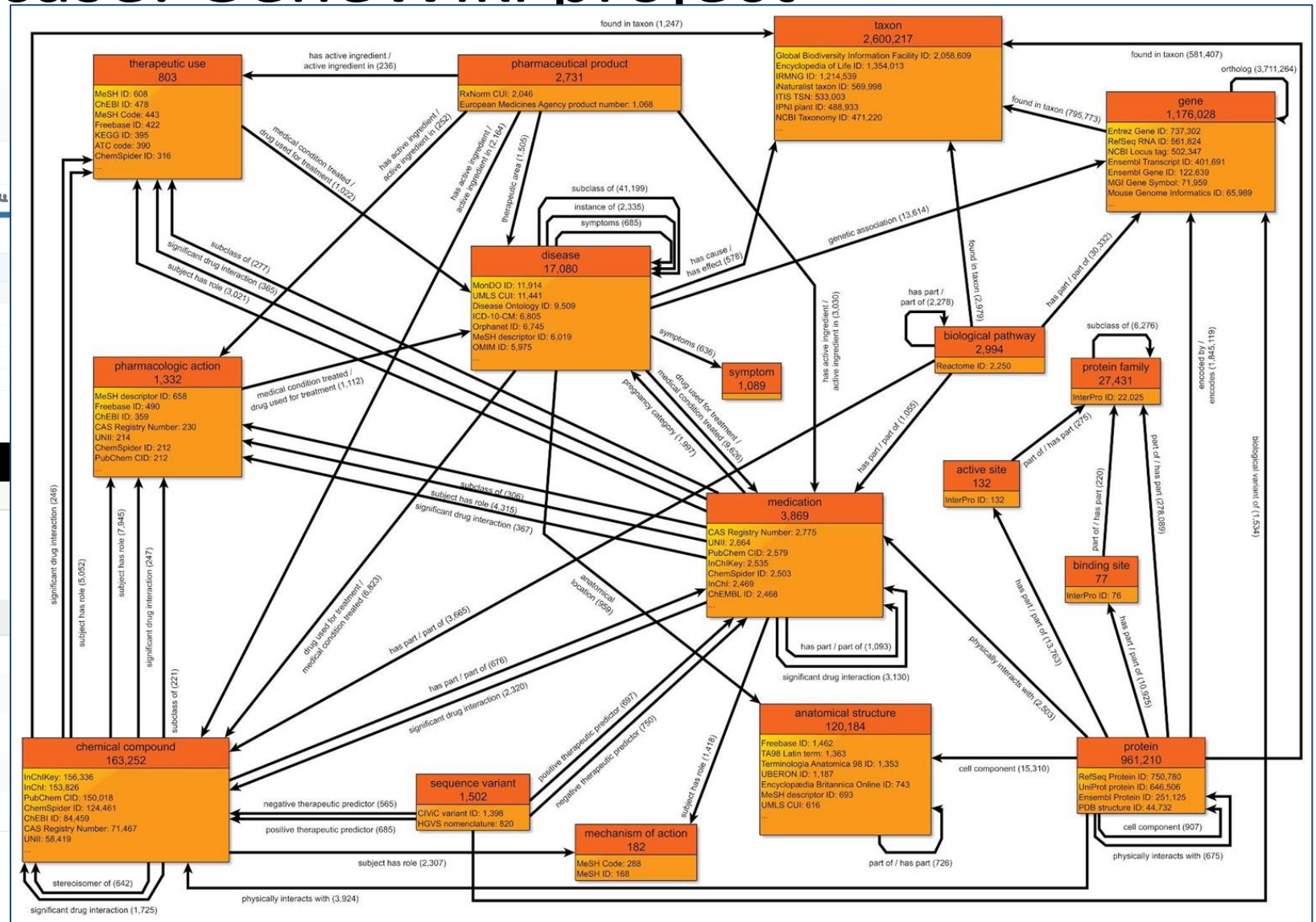
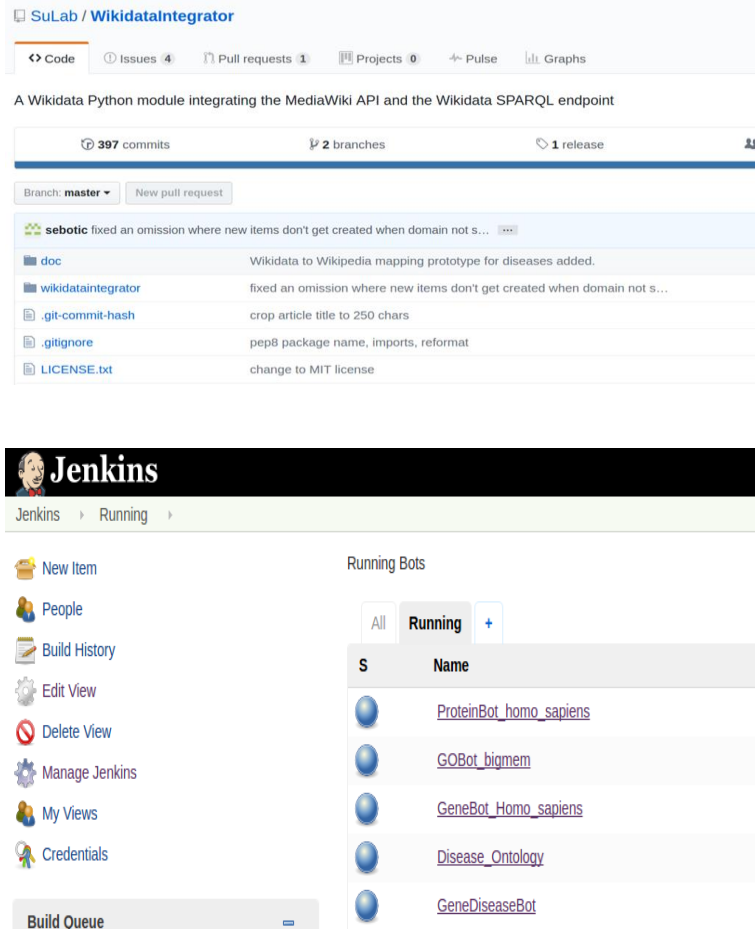
KG analysis

- Create and compare subsets from historical dumps

License combination

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

Motivating use case: GeneWiki project



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

}

}

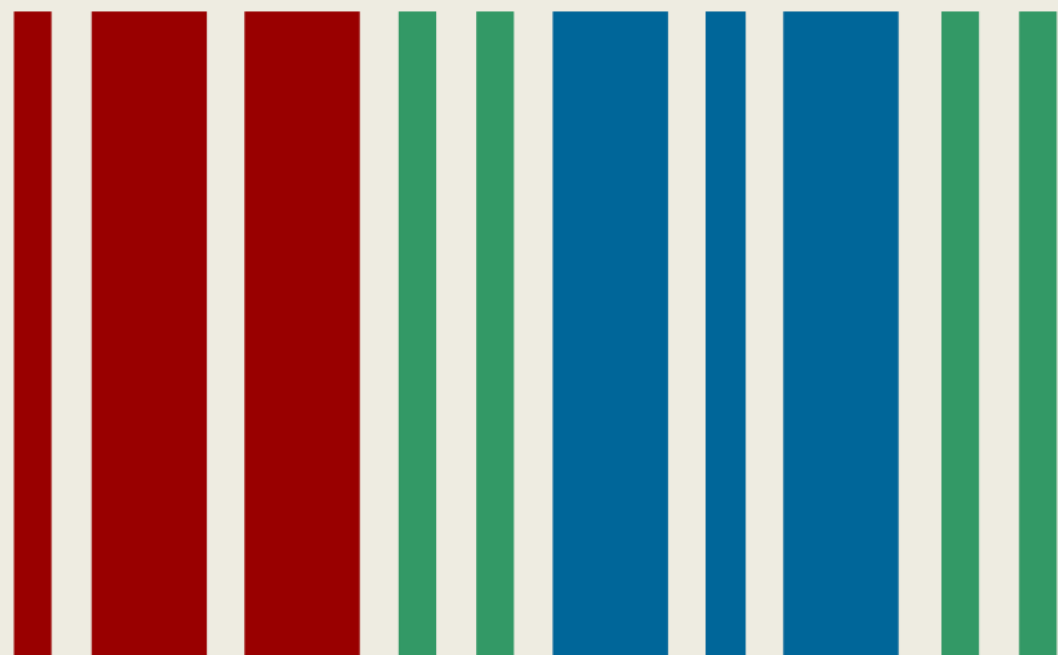
Wikidata subsetting efforts

Collaborative research driven by a practical need

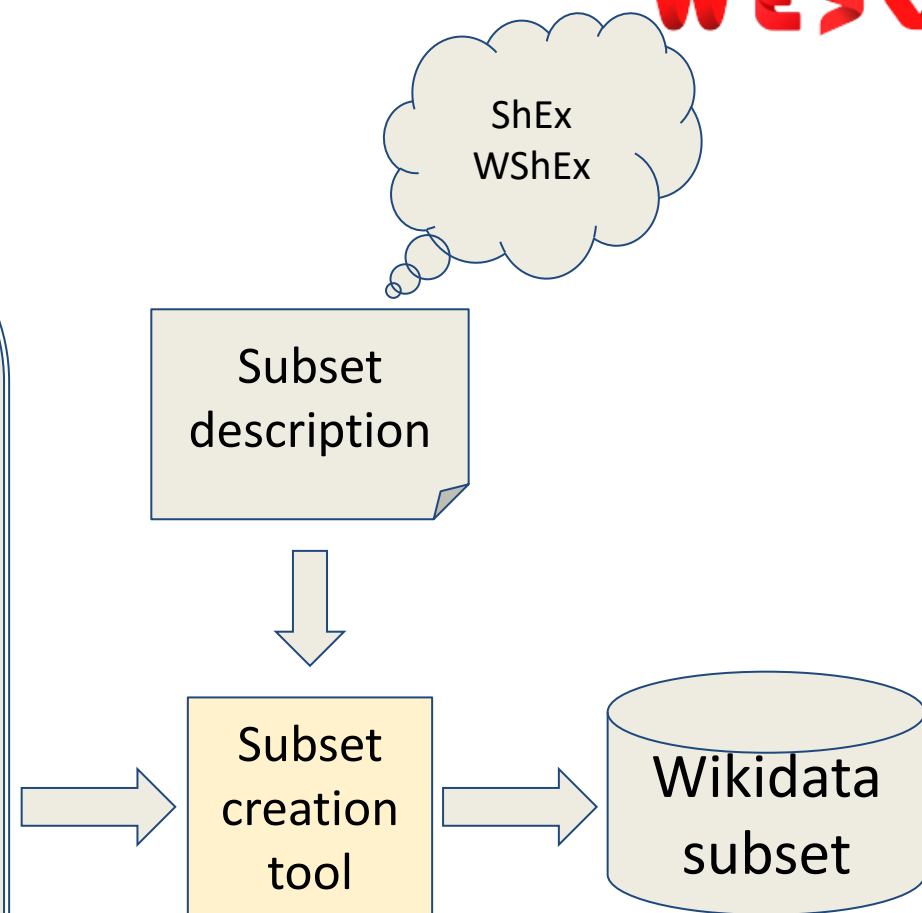
Most of the advances were triggered by SWAT4HCLS and Biohackathons

- 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



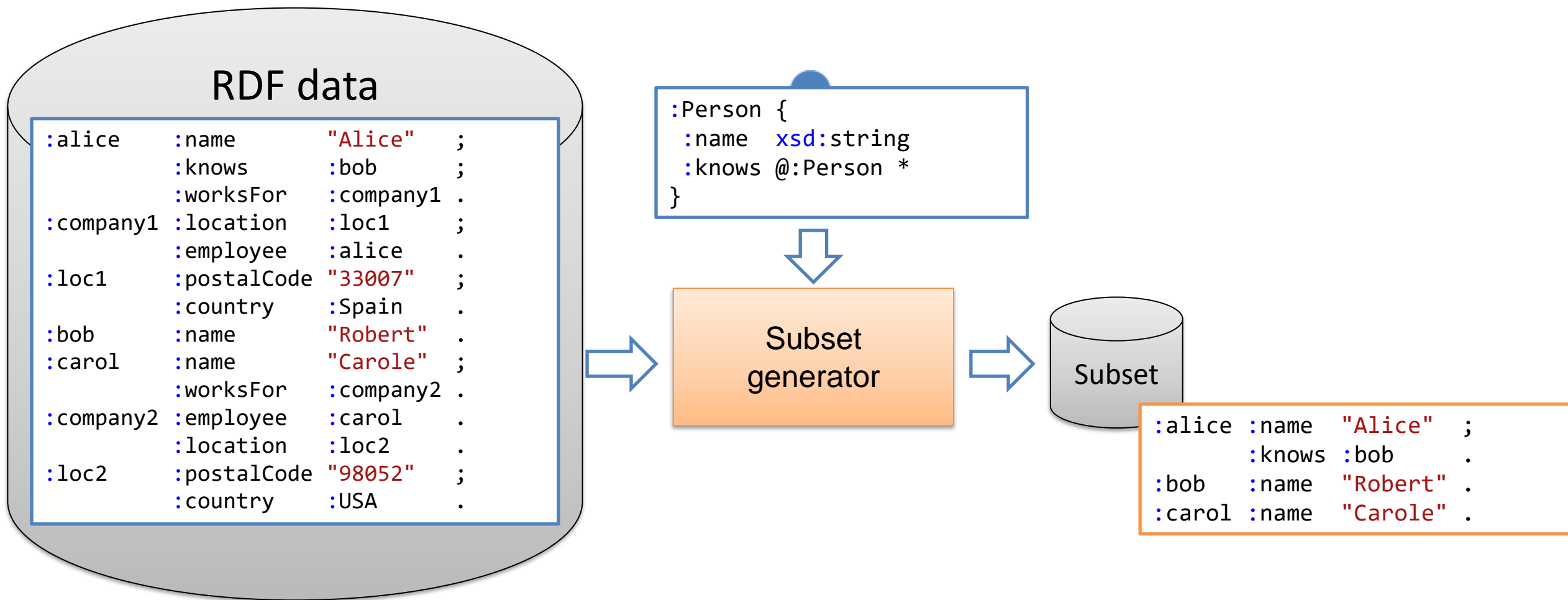
WIKIDATA



Subsetting based on ShEx

Generate subsets from ShEx

ShEx describes the contents of the expected subset



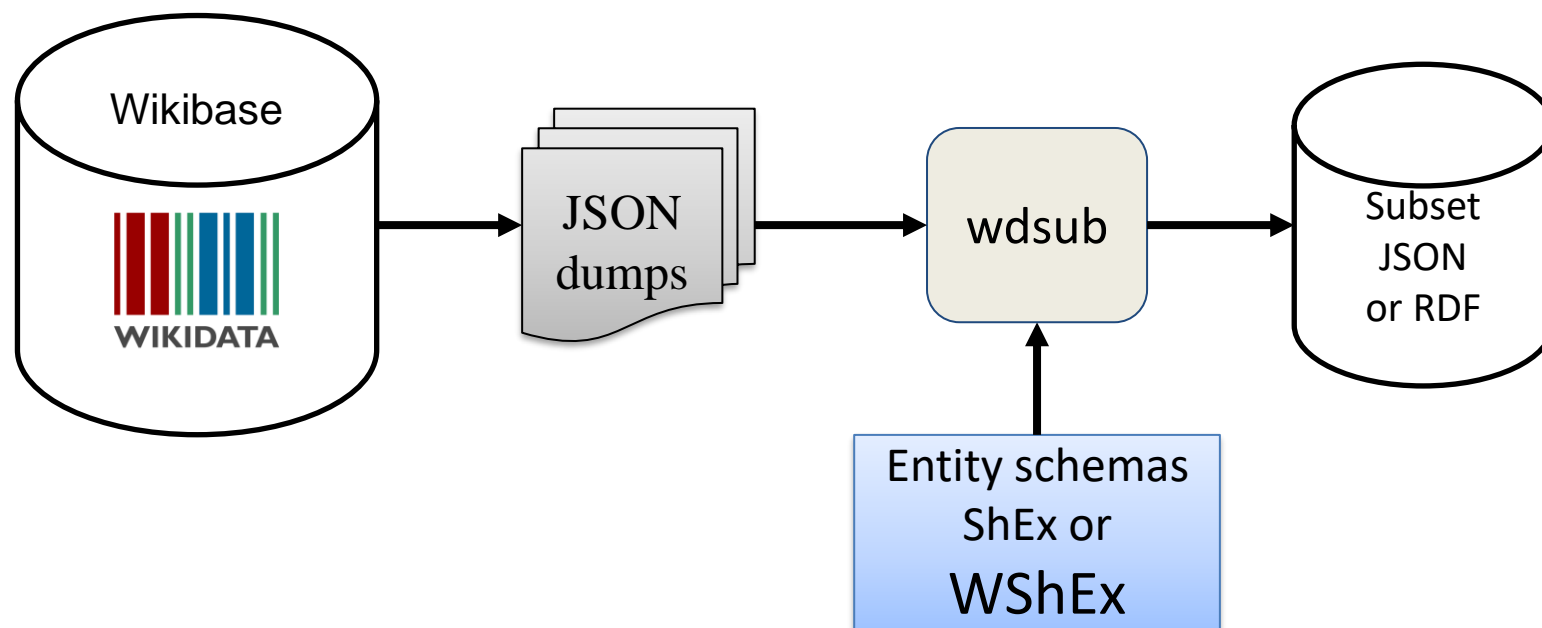
wdsub

Input:

- ShEx/WShEx schema
- Wikidata dumps in JSON

Output

- Dumps in JSON/RDF

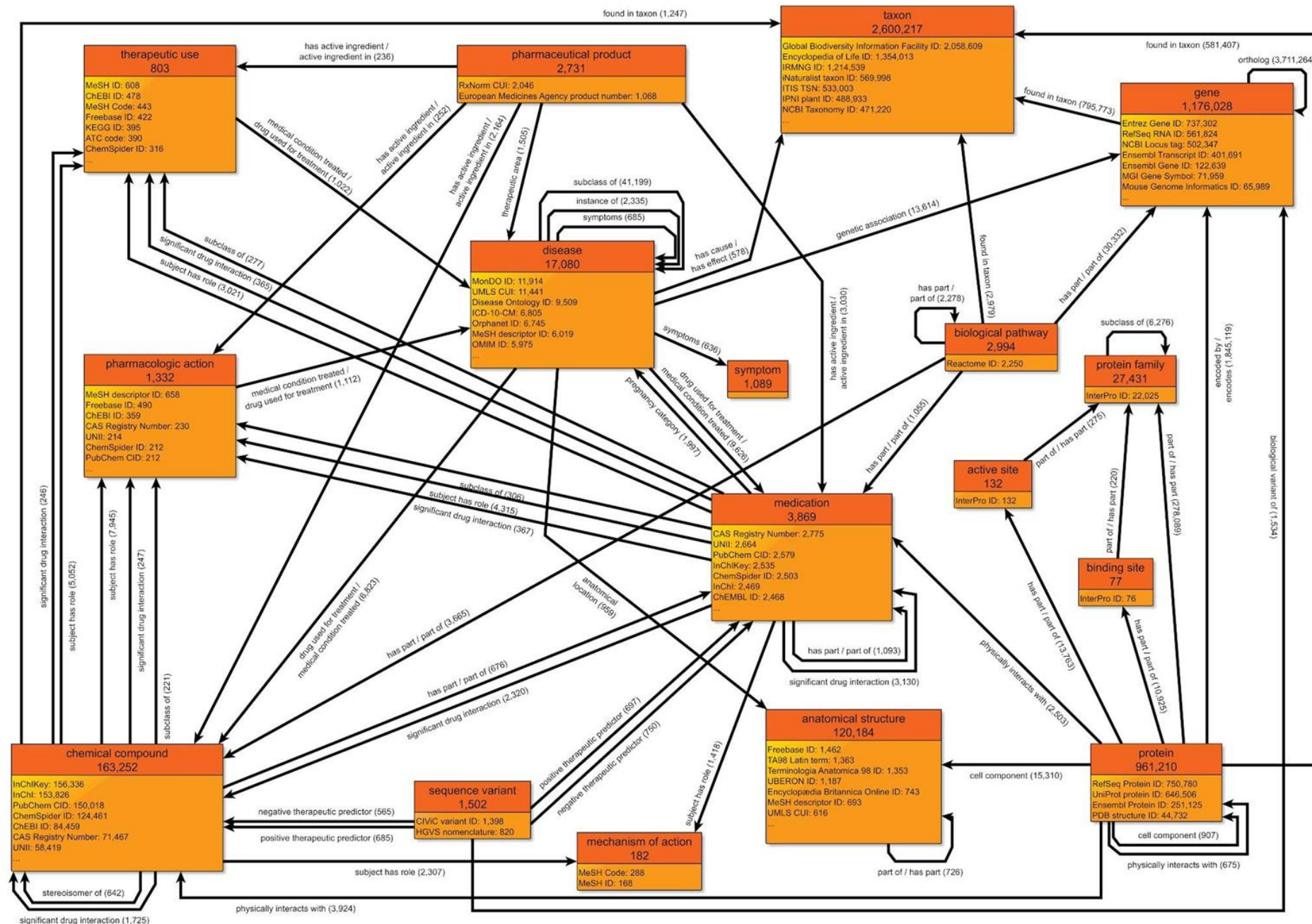


Link: <https://github.com/weso/wdsub>

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

GeneWiki project

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 the URI "aussies:HolgersAddress". The form is generated from an "Australian address shape". It features a toolbar with "Explore", "Modify", "Cancel", "Preview", and "Save Changes" buttons. The form is organized into two main sections: "Address" and "Contact".

Address Section:

- street address:** A text input field containing "3 Teewah Close".
- suburb:** A text input field containing "Kewarra Beach".
- state:** A dropdown menu currently showing "QLD".
- postal code:** A text input field containing "48791".

Contact Section:

- email:** Two text input fields, the first containing "holger@knublauch.com" and the second containing "holger@topquadrant.com".
- phone number:** A text input field.

Generating code from shapes

Domain model based on Shapes

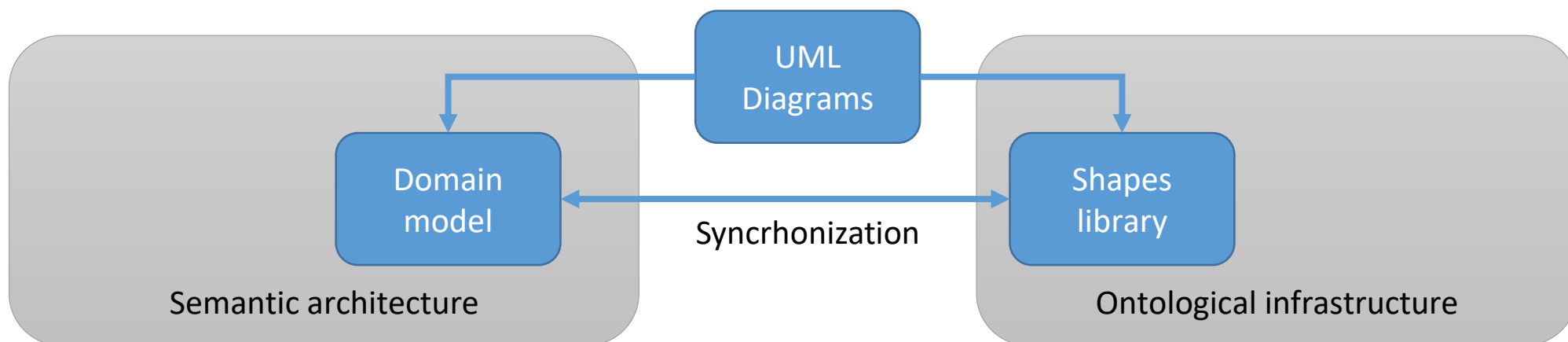
Clean architecture pattern

Domain model as central element

Simple classes (POJO): Plain Old Java Objects

Shapes synchronization

Application logic and services based on domain model

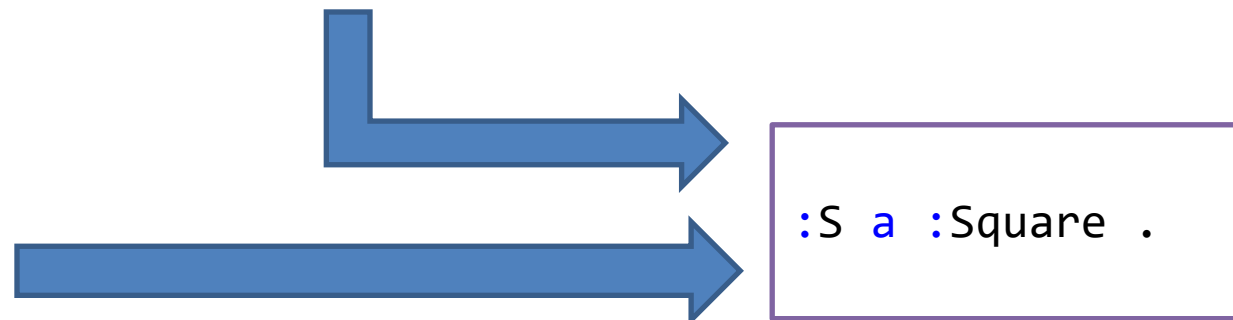


Shapes and rules

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