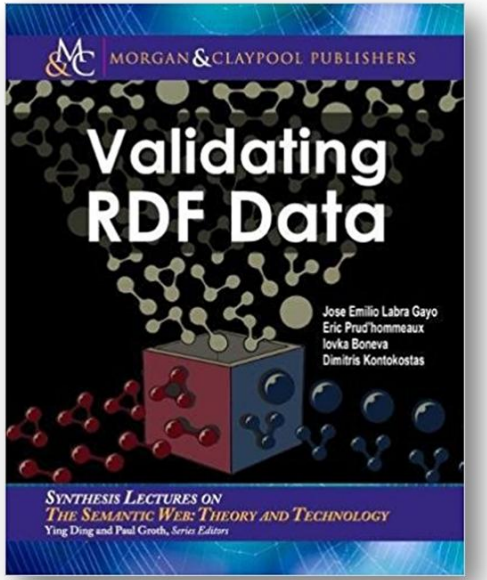
ShEx primer

Jose Emilio Labra Gayo
WESO Research group
University of Oviedo, Spain
https://labra.weso.es/

More information

Web page: http://shex.io
Semantics: http://shex.io/shex-semantics/
Primer: http://shex.io/shex-primer



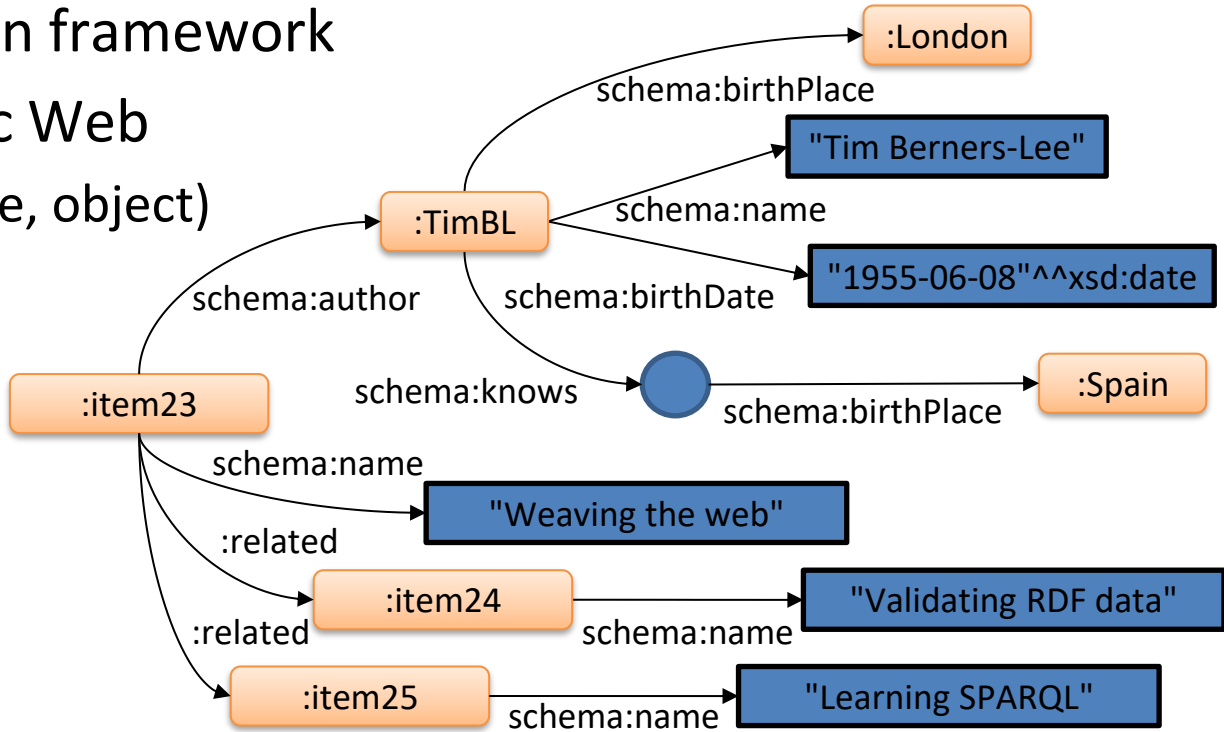
Jose E. Labra Gayo, Eric Prud'hommeaux, Iovka Boneva, Dimitris Kontokostas, *Validating RDF Data*, Synthesis Lectures on the Semantic Web, Vol. 7, No. 1, 1-328, DOI: 10.2200/S00786ED1V01Y201707WBE016, Morgan & Claypool (2018)
Online version: http://book.validatingrdf.com/

RDF graphs

RDF = resource description framework

Lingua franca of Semantic Web

- Triples: (subject, predicate, object)
- Predicates = URIs
  - Interoperability



Try it: https://rdfshape.weso.es/link/17089394117

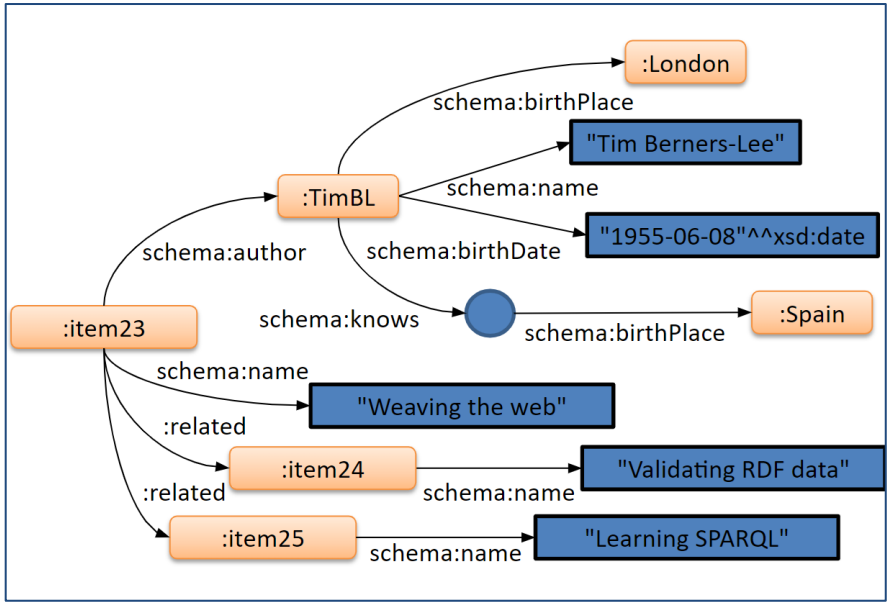
RDF syntaxes

One data model, several syntaxes: Turtle, N-Triples, JSON-LD, …

Turtle

```
prefix :       <http://example.org/>
prefix xsd:    <http://www.w3.org/2001/XMLSchema#>
prefix rdf:    <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
prefix schema: <http://schema.org/>

:item23 schema:name      "Weaving the web"      ;
        schema:author     :timbl                ;
        :related          :item24, :item25      .
:timbl  schema:name       "Tim Berners-Lee"     ;
        schema:birthDate  "1955-06-08"^^xsd:date ;
        schema:birthPlace :london               ;
        schema:knows      _:1                   .
_:1     schema:birthPlace :Spain                .
:item24 schema:name      "Validating RDF data"  .
:item25 schema:name      "Learning SPARQL"      .
```



Try it: https://rdfshape.weso.es/link/17089394117

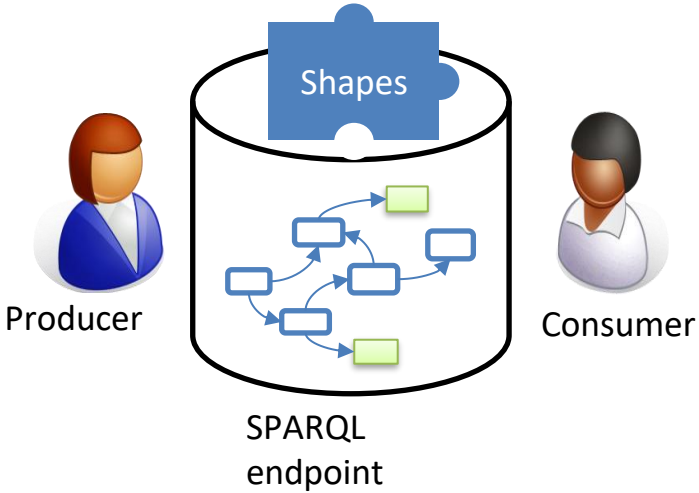
Why describe & validate RDF?

For producers

- Developers can understand the contents they are going to produce
- They can ensure they produce the expected structure
- Advertise and document the structure
- Generate interfaces

For consumers

- Understand the contents
- Verify the structure before processing it
- Query generation & optimization



Schemas for RDF?

RDF doesn’t impose a schema, but…

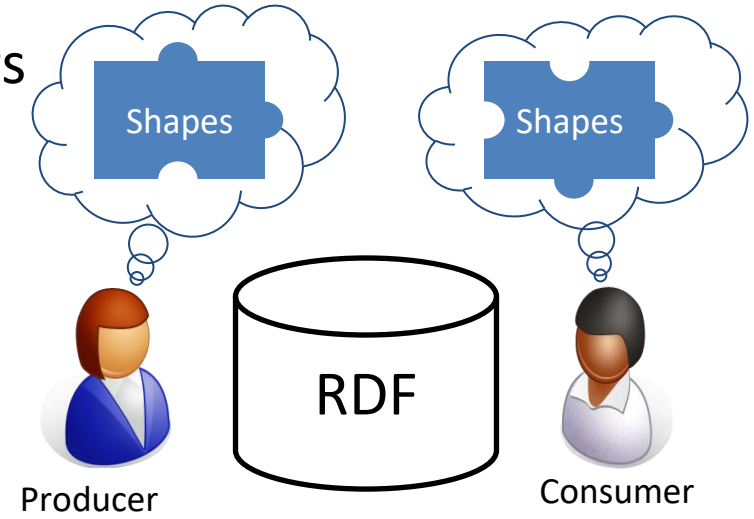
In practice, there are **implicit schemas**

Assumed by producers and consumers

Shapes make schemas explicit

Handle malformed/incomplete data

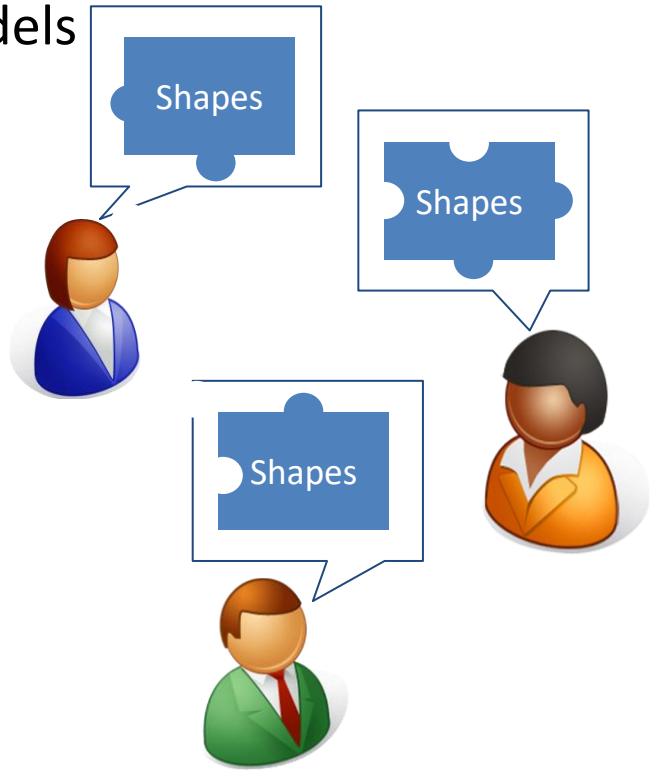
Avoid defensive programming



Focus discussions on what matters

Help domain experts define their own data models
- Understandable by domain experts
- …and machine processable

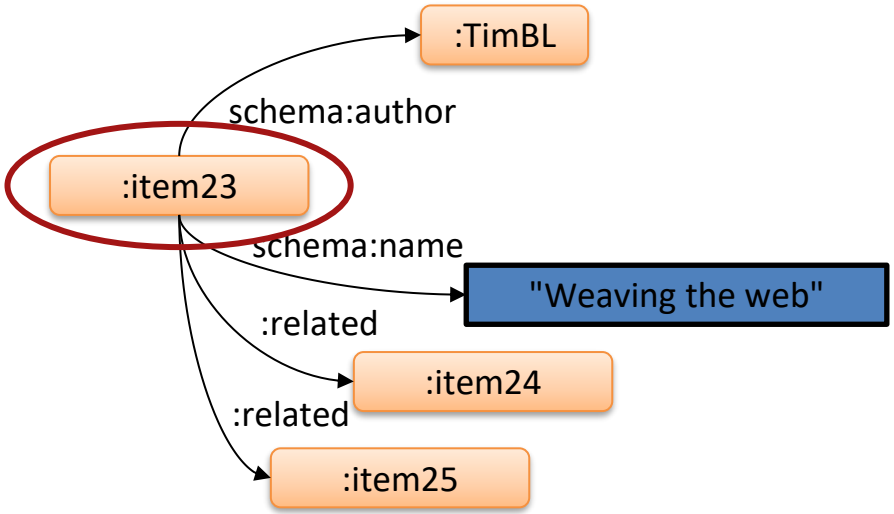
Initial motivation: clinical data models (FHIR)
- Distributed data model
  - Different location, authorities,...
- Extensible data models



What is a shape?

A shape describes

- The form of a node
- Incoming/outgoing arcs from a node
- Values associated with those arcs



RDF Node

```
:item23 schema:author :timbl              ;
        schema:name   "Weaving the web"   ;
        :related      :item24  , :item25 .
```

ShEx

```
:Book IRI and {
 schema:author @:Author    + ;
 schema:name   xsd:string    ;
 :related      @:Book      *
}
```

https://rdfshape.weso.es/link/17089392604

Evolution of ShEx

2013 RDF Validation Workshop

Conclusions of the workshop:

…need of a higher level, concise language for RDF Validation

ShEx initially proposed (v 1.0)

2014 W3C Data Shapes WG chartered

2017 ShEx 2.0 released as W3C Community group draft

2017 SHACL accepted as W3C recommendation

2019 ShEx 2.1 added support for imports

2019 ShEx adopted by Wikidata

2022 Added support for *extends*

2024 IEEE ShEx (*work in progress*)

Shape Expressions - ShEx

Goal: Concise and human-readable language

3 syntaxes:

- ShExC: Compact syntax, similar to Turtle or SPARQL
- ShExJ: JSON(-LD), for the spec
- ShExR: RDF, based on JSON-LD

Note: Round tripping is possible, convert from one to the other

Semantics inspired by regular expressions

ShEx libraries and demos

Implementations & libraries:

- shex.js: Javascript
- Jena-ShEx: Java
- SHaclEX: Scala (Jena/RDF4j)
- PyShEx: Python
- shex-java: Java
- Ruby-ShEx: Ruby
- RDF-Elixir: Elixir
- rudof: Rust 

Online demos & playgrounds

- ShEx-simple
- RDFShape
- Wikishape
- ShEx-Java
- ShExValidata

More info: http://shex.io

Simple example

Prefix declarations as in Turtle/SPARQL

```
prefix :       <http://example.org/>
prefix xsd:    <http://www.w3.org/2001/XMLSchema#>
prefix schema: <http://schema.org/>

:Book IRI AND {
 schema:name   xsd:string   ;
 :related      @:Book     *
}
```

Nodes conforming to `:Book` must

- Be `IRI`s and
- Have property `schema:name` with a value of type `xsd:string` (exactly one)
- Have property `:related` with values conforming to `:Book` (zero or more)

RDF Validation using ShEx

Schema

```
:Book IRI AND {
 :name    xsd:string   ;
 :related @:Book      *
}
```

RDF Data

```
:a   :name    "Title A" ;
     :related :b  .
:b   :related :a ;
     :name    "Title B".
:c   :name    "Title C1", "Title C2" .
:d   :name    234 .
:e   :namme   "Title E" .
:f   :name    "Title F" ;
     :related :a, _:1 .
_:1  :name    "Unknown title" .
```

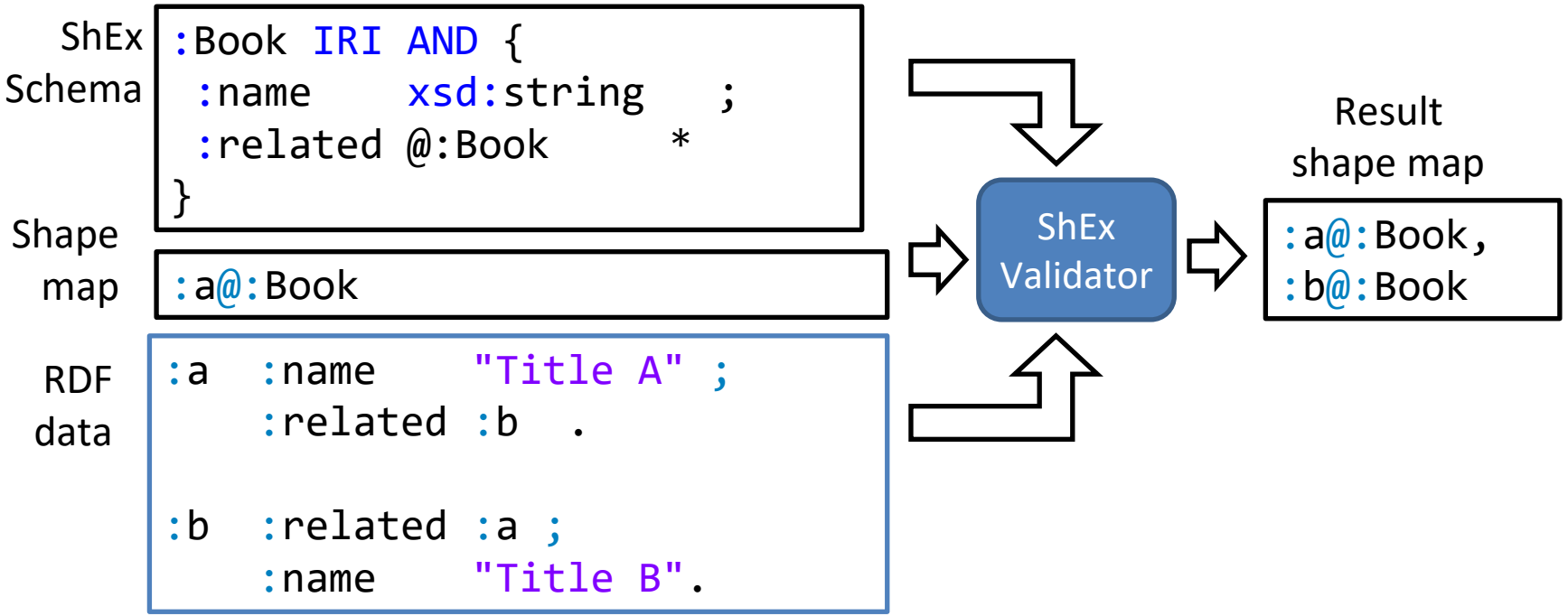
Shape map

```
:a@:Book   ✓
:b@:Book,  ✓
:c@:Book,  ✗
:d@:Book,  ✗
:e@:Book,  ✗
:f@:Book   ✗
```

Try it (RDFShape)
Try it (Simple ShExDemo)

Validation process

Input: RDF data, ShEx schema, Shape map
Output: Result shape map



Node constraints

Constraints over a node (without considering its neighbourhood)

```
:Book {
 :name           xsd:string
 :datePublished  xsd:date
 :numberOfPages  MinInclusive 1
 :author         @:Person
 :genre          [:Fiction :NonFiction ]
 :isbn           /isbn:[0-9X]{10}/
 :publisher      IRI
 :audio          .
 :maintainer     @:Person OR
                 @:Organization
}
:Person {}
:Organization {}
```

```
:item23
 :name           "Weaving the Web"                ;
 :datePublished  "2012-03-05"^^xsd:date           ;
 :numberOfPages  272                              ;
 :author         :timbl                           ;
 :genre          :NonFiction                      ;
 :isbn           "isbn:006251587X"                ;
 :publisher      <http://www.harpercollins.com/>  ;
 :audio          <http://audio.com/item23>        ;
 :maintainer     :alice                           .
```

Try it: (RDFShape)

Cardinalities

Inspired by regular expressions: +, ?, *, {m,n}

By default {1,1}

```
:Book {
 :name          xsd:string        ;
 :numberOfPages xsd:integer  ?    ;
 :author        @:Person     +    ;
 :publisher     IRI          ?    ;
 :maintainer    @:Person     {1,3} ;
 :related       @:Book       *
}
:Person {}
:Organization {}
```

```
:item23
  :name          "Weaving the Web"       ;
  :numberOfPages 272                     ;
  :author        :timbl, :markFischetti  ;
  :maintainer    :alice,:bob             .
```

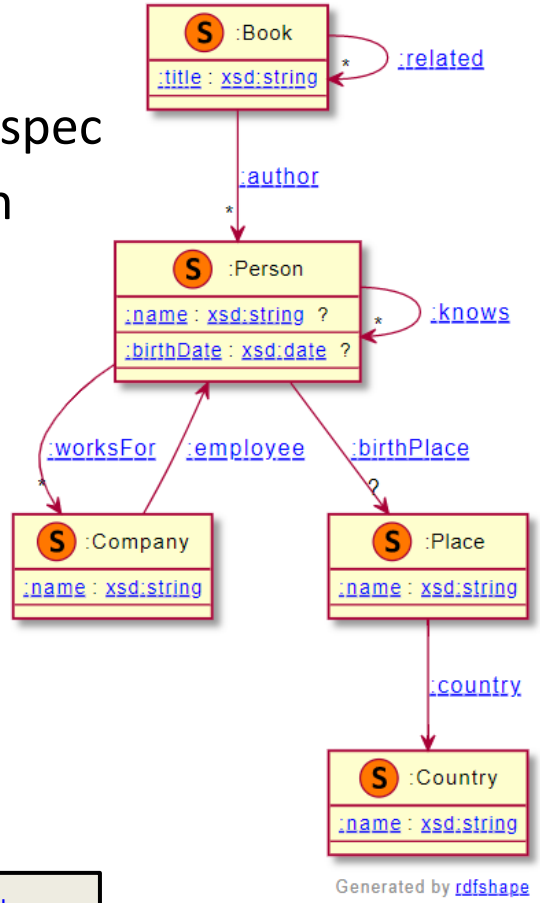
Try it: RDFShape

Recursive schemas

Support for recursive (cyclic) data models is part of the spec

Well formed semantics based on stratified negation

```
:Book    {  :title        xsd:string   ;
            :author       @:Person   * ;
            :related      @:Book     * }
:Person  {  :name         xsd:string ? ;
            :birthDate    xsd:date   ? ;
            :birthPlace   @:Place    ? ;
            :knows        @:Person   * ;
            :worksFor     @:Company  * }
:Place   {  :name         xsd:string   ;
            :country      @:Country    }
:Country {  :name         xsd:string   }
:Company {  :name         xsd:string   ;
            :employee     @:Person     }
```



Try it: RDFShape

Open/Closed content models

RDF semantics mostly presume open content models

Shape expressions are open by default

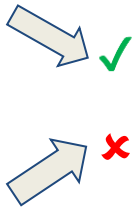
Enable extensibility

But…some use cases require closed content models

Added CLOSED keyword

```
:Book {
 :name     xsd:string   ;
}
```

```
:Book CLOSED {
 :name     xsd:string   ;
}
```



```
:a :name "Weaving the web" ;
   :isbn "006251587X"      .
```

Try it: RDFShape

Open/Closed properties

Property values are closed by default (closed properties)

```
:Book {
 :code /isbn:[0-9X]{10}/ ;
}
```

✓
```
:item23  :code "isbn:006251587X" .
```

✘
```
:item23  :code 23                .
```

Properties can be repeated

```
:Book {
  :code  /isbn:[0-9X]{10}/ ;
  :code  /isbn:[0-9]{13}/
}
```

```
:item23 :code "isbn:006251587X"    ,
        :code "isbn:9780062515872" .
```

EXTRA declares properties as open

```
:Book EXTRA :code {
 :code /isbn:[0-9X]{10}/ ;
}
```

```
:item23 :code "isbn:006251587X" ,
        :code 23                .
```

Try it: RDFShape

Triple expressions

“Unordered” regular expressions: Regular bag expressions

```
:Person {
 ( :name       xsd:string |
   :firstName xsd:string  + ;
   :lastName  xsd:string
 ) ;
 :birthDate   xsd:date    ?
}
```

```
(name |
  firstName + ;
  lastName
) ;
birthDate ?
```

```
(n | f + ; l ) ; b?
```

```
:alice :name      "Alice Cooper"          ;
       :birthDate "2010-02-23"^^xsd:date .

:bob   :firstName "Robert"                ;
       :lastName  "Smith"                 .

:carol :firstName "Carol"                 ;
       :lastName  "King"                  ;
       :firstName "Carole"                .
```

```
:alice → n
:bob   → f l
:carol → f l f
```

```
:dave  → n f
```

```
:dave :name      "Dave Navarro"           ;
      :firstName "Dave"                   .
```

Try it: RDFShape

Logical operators

Shape Expressions can be combined with AND, OR, NOT

```
:Book {
 :name   xsd:string ;
 :author @:Person OR @:Organization ;
}

:AudioBook @:Book AND {
 :name           MaxLength 20 ;
 :readBy         @:Person     ;
} AND NOT {
 :numberOfPages . +
}

:Person {}
:Organization {}
```

```
:item24 :name          "Weaving the Web" ;
        :author        :timbl            ;
        :readBy        :timbl            .
```

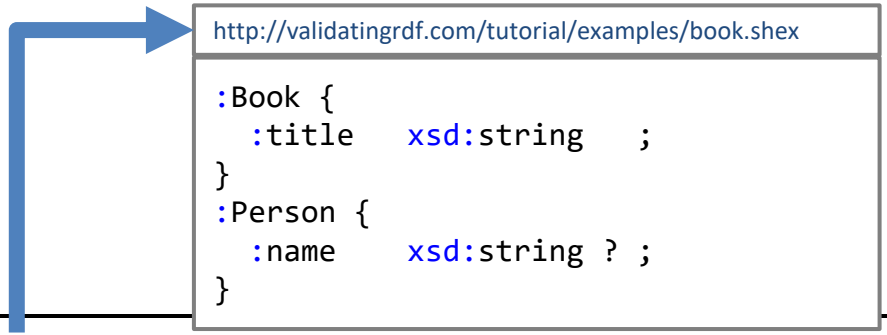
```
:item23 :name          "Weaving the Web" ;
        :author        :timbl            ;
        :numberOfPages 272               ;
        :readBy        :timbl            .
```

ShEx allows NOT combined with recursion (semantics based on stratified negation)

Try it: RDFShape

Importing schemas

`import` statement can be used to import schemas




```
import <https://www.validatingrdf.com/examples/book.shex>

:AudioBook @:Book AND {
 :title          MaxLength 20 ;
 :readBy         @:Person     ;
}
```

```
:item24 :name          "Weaving the Web" ;
        :author        :timbl            ;
        :readBy        :timbl            .
```

Try it: RDFShape

Machine processable annotations

Annotations based on RDF

Lots of applications, e.g. generate forms from shapes

```
prefix schema: <http://schema.org/>
prefix : <http://example.org/>
prefix xsd: <http://www.w3.org/2001/XMLSchema#>
prefix ui: <http://www.w3.org/ns/ui#>

start = @:User

:User {
 schema:name      xsd:string                    // ui:label “Name"       ;
 schema:birthDate xsd:dateTime                  // ui:label "Birth date" ;
 schema:gender    [ schema:Male schema:Female ] // ui:label "Gender"
}
```

Try it:
Eric’s demo
https://rdfshape.weso.es/link/17095003321

More complex Shape Maps

Shape Maps define which nodes validate with which shapes

Examples:

`{FOCUS a :Person}@:User`	https://rdfshape.weso.es/link/17095033013
`{FOCUS schema:name _}@:Book`	https://rdfshape.weso.es/link/17095014386

`SPARQL """`
`PREFIX schema: <http://schema.org/>`
`SELECT ?book WHERE {`
`?book schema:name ?name;`
`schema:author ?author`
`}"""@:Book`
	https://rdfshape.weso.es/link/17095031848

Inheritance model for ShEx

`extends` allows to reuse existing shapes adding new content

Handles closed properties and shapes

```
:Book {
 :name    xsd:string ;
 :author  @:Person   ;
 :code    /isbn:[0-9]{13}/   ;
 :code    /isbn:[0-9X]{10}/
}

:LibraryBook extends @:Book {
 :code    /internal:[0-9]*/ ;
}
```

```
:item23 :name        "Weaving the Web"    ;
        :author      :timbl               ;
        :code        "isbn:006251587X"    ;
        :code        "isbn:9780062515872" ;
        :code        "internal:234"       .
```

Other features

- Multiple inheritance
- Abstract shapes

More details: ***Shape Expressions with Inheritance***, Iovka Boneva, Jose Emilio Labra Gayo, Eric Prud'hommeaux, Katherine Thornton, Andra Waagmeester *Extended Semantic Web Conference*, Portoroz, Slovenia, 2025 – 2025
https://labra.weso.es/publication/2025_shex_inheritance/ **(Next Wednesday 16:20h)**

Try it:
https://rdfshape.weso.es/link/17487753484

ShEx vs SPARQL

SPARQL pros:

- Expressive
- Ubiquitous

SPARQL cons:

- Expressive
- Idiomatic - many ways to encode the same constraint
- Non recursive

```
<User> {
  schema:name    xsd:string ;
  schema:gender [ schema:Female schema:Male]
}
```

```
ASK {{ SELECT ?Person {
          ?Person schema:name ?o .
    } GROUP BY ?Person HAVING (COUNT(*)=1)
  }
  { SELECT ?Person {
          ?Person schema:name ?o .
     FILTER ( isLiteral(?o) &&
              datatype(?o) = xsd:string )
     } GROUP BY ?Person HAVING (COUNT(*)=1)
  }
  { SELECT ?Person (COUNT(*) AS ?c1) {
          ?Person schema:gender ?o .
    } GROUP BY ?Person HAVING (COUNT(*)=1)}
    { SELECT ?Person (COUNT(*) AS ?c2) {
          ?S schema:gender ?o .
      FILTER ((?o = schema:Female ||
               ?o = schema:Male))
    } GROUP BY ?Person HAVING (COUNT(*)=1)}
    FILTER (?c1 = ?c2)
}
```

3 syntaxes: ShExC, ShExJ, ShExR

ShExC

```
prefix :       <http://example.org/>
prefix xsd:    <http://www.w3.org/2001/XMLSchema#>
prefix schema: <http://schema.org/>

:Book {
 schema:name   xsd:string   ;
 :related      @:Book     *
}
```

ShExR (RDF, Turtle)

```
:Book a sx:ShapeDecl ;
  sx:shapeExpr [ a sx:Shape ;
   sx:expression [ a sx:EachOf ;
    sx:expressions (
      [ a sx:TripleConstraint ;
        sx:predicate schema:name ;
        sx:valueExpr  [ a sx:NodeConstraint ;
                        sx:datatype xsd:string
                      ] ]
      [ a sx:TripleConstraint ;
        sx:predicate :related ;
        sx:valueExpr  [ a sx:NodeConstraint ;
        sx:valueExpr :Book ] ] ) ] ] .
```

It's posible to roudtrip from each one

ShExJ (JSON LD)

```
{ "type" : "Schema",
  "@context" : "http://www.w3.org/ns/shex.jsonld",
  "shapes" : [ {
      "type" : "Shape",
      "id" : "http://example.org/Book",
      "expression" : {
        "type" : "EachOf",
        "expressions" : [ {
            "type" : "TripleConstraint",
            "predicate" : "http://schema.org/name",
            "valueExpr" : {
              "type" : "NodeConstraint",
              "datatype" : "http://www.w3.org/2001/XMLSchema#string"
            }
          },
          { "predicate" : "http://example.org/related",
            "valueExpr" : "http://example.org/Book",
            "min" : 0,
            "max" : -1,
            "type" : "TripleConstraint"
          }
        ] } } ] }
```

ShEx from a more theoretical point of view

Grammar divided in two main blocks

se Shape expressions

- Describe nodes

te Triple expressions

- Describe the neighbourhood of nodes
- Incoming/outgoing arcs
- Regular *Bag* Expressions

Recursion and negation

- Stratified negation

$\mathcal{S}$	::=	$l \mapsto se^*$	
se	::=	IRI | BNode | ...	Node constraints
	|	cond	A boolean condition on nodes
	|	se_1 AND se_2	Conjunction
	|	se_1 OR se_2	Disjunction
	|	NOT se	Negation
	|	$@l$	Shape label reference for $l \in \Lambda$
	|	{ te }	Triple expression te
te	::=	$te_1; te_2$	Each of te_1 and te_2
	|	$te_1 \mid te_2$	Some of te_1 or te_2
	|	$\_ \xrightarrow{p} @l$	Triple with predicate p that conforms to shape expression identified by l
	|	$te*$	Zero or more te

Summary

- More ShEx features
  - Stems, named expressions, nested shapes, semantic actions, …
- And ShEx tools
  - Inference ShEx from data (sheXer), editors, KG subsets, …
- ShEx and SHACL compared (see later)
  - Different underlying philosophy
    - ShEx more inspired on grammars than on constraints
  - Separation of concerns
    - Structure definition (ShEx) ≠ Ontology (OWL)
    - Structure definition (ShEx) ≠ Node/shape selection (ShapeMaps)

ShEx Tools

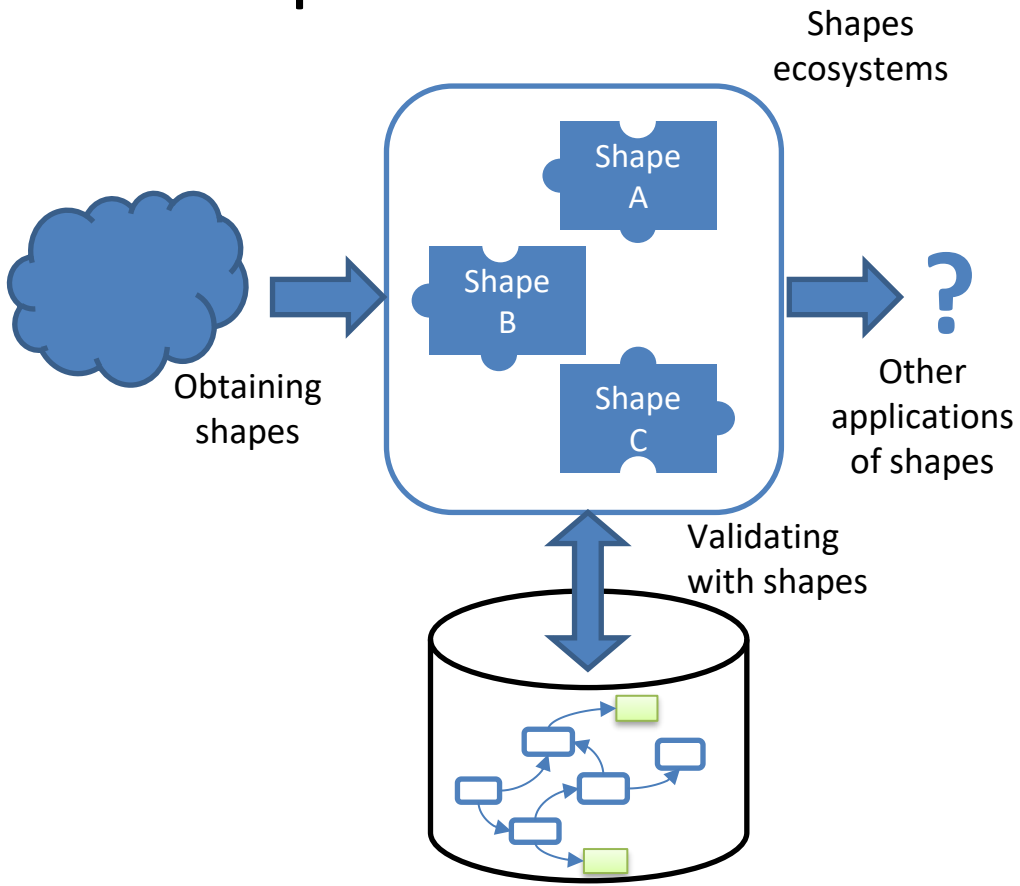
Tools landscape

Validating with shapes

Obtaining shapes

Other applications of shapes

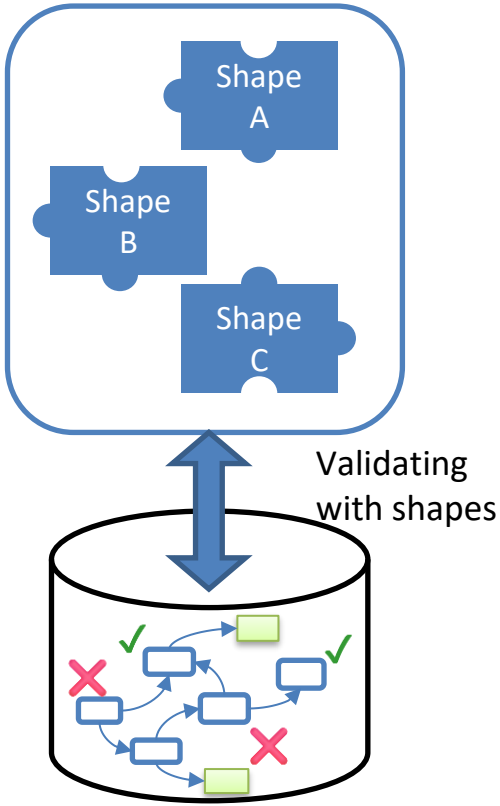
Shapes ecosystems



Validating with shapes

Libraries and online demos

Continuous integration with Shapes



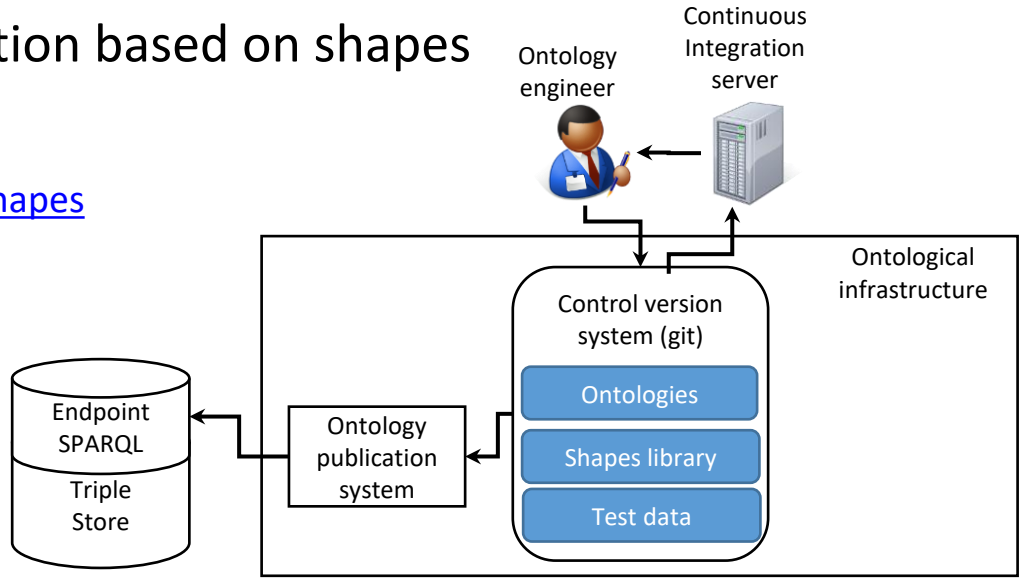
Continuous integration with Shapes

Coexistence between ontologies/shapes

- Shapes can validate the behaviour of inference systems
- Shapes pre- and post- inference
- TDD and continuous integration based on shapes

Gene Ontology Shapes:

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



Continuous integration with Shapes

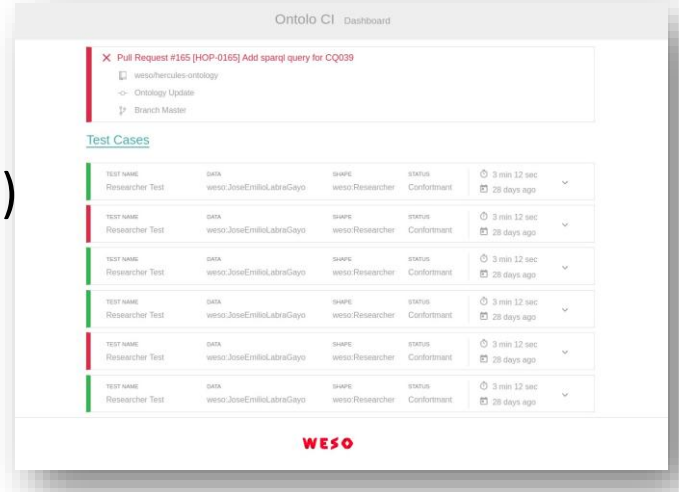
Ontolo-ci: https://www.weso.es/ontolo-ci/

Developed as part of HERCULES-Ontology

Test-Driven-Development applied to ontologies

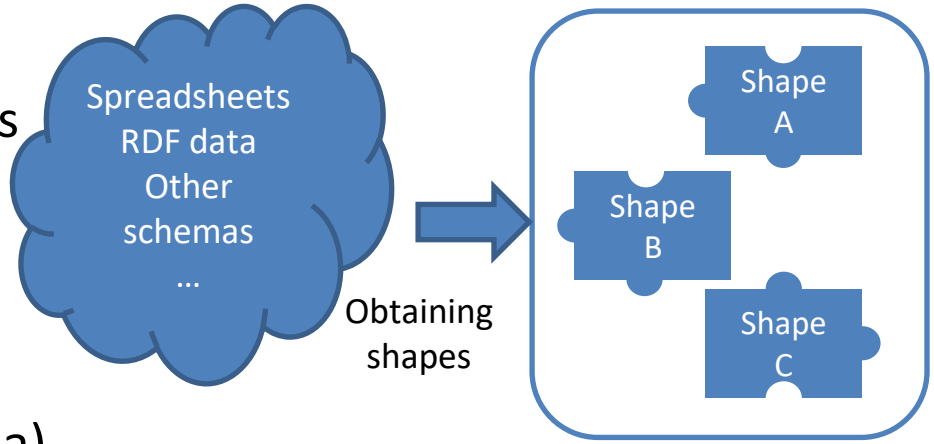
Input:

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



Obtaining shapes

- Shapes editors
  - Text-based editors
  - Visual editors and visualizers
- Obtaining shapes from...
  - Spreadsheets
  - RDF data
  - Other schemas (XML Schema)



Text-based editors

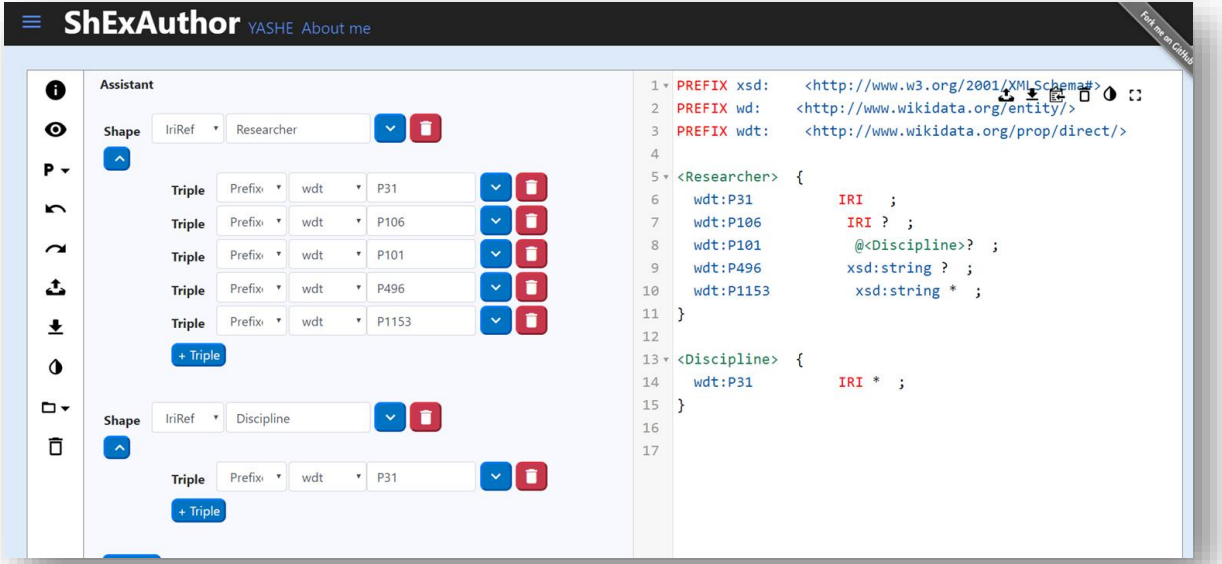
YaSHE: Forked from YASGUI: http://www.weso.es/YASHE/

- Syntax highlighting
- Auto-completion

Shapes author tools: ShEx Author

ShEx-Author: Inspired by Wikidata Query Service

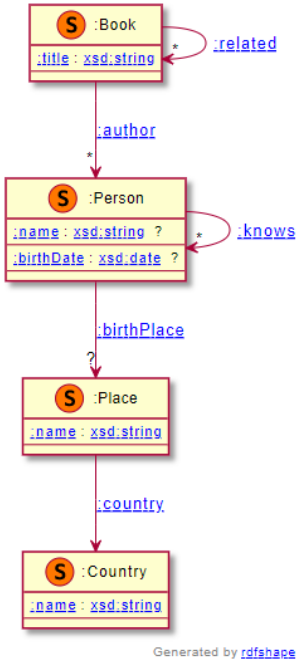
2 column: Visual one synchronized with text based



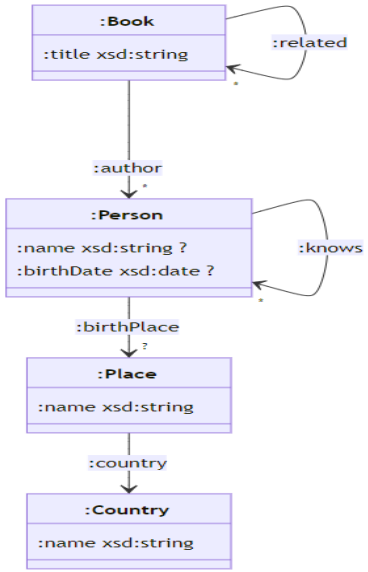
Shapes visualizations

Several options integrated in RDFShape/Wikishape

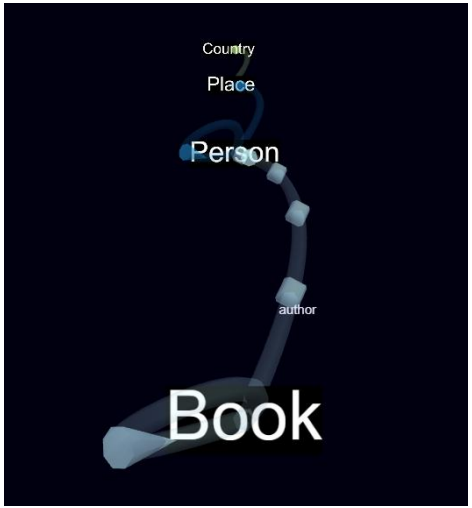
[UMLSHaclEX](#)
UML diagrams using graphviz



[ShUMLex](#)
UML through XMI



[3DShEx](#)
Visualization of shapes in 3D



Shapes from Spreadsheets

ShExCSV: CSV representation of Shapes

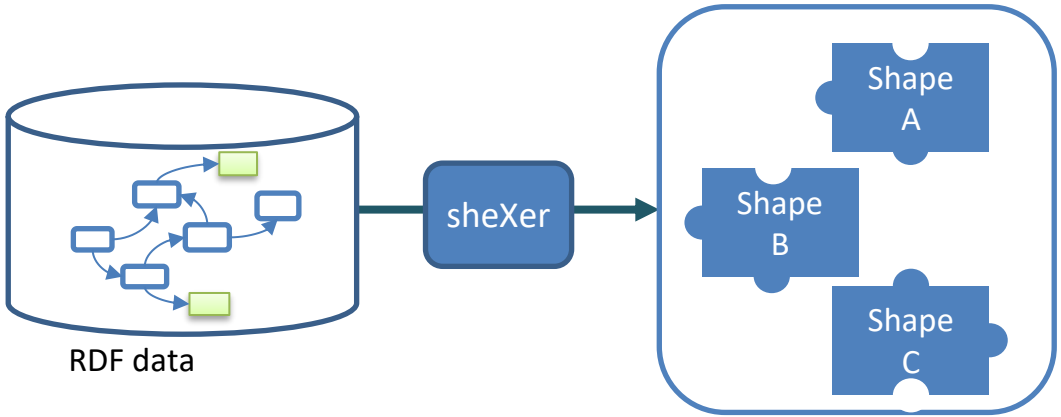
Hermes: ShExCSV processor, https://github.com/weso/hermes

DC Tabular Application Profiles
(https://tap2shex.readthedocs.io/)

Generating Shapes from RDF data

In practice, a lot of RDF data is already there

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

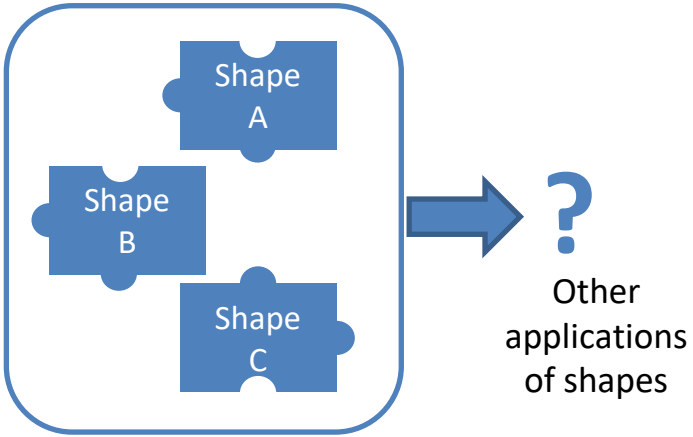


Other applications of Shapes

UIs and shapes

Generating code from Shapes

Generate subsettings



UI and shapes: ShapeForms

```
prefix schema: <http://schema.org/>
prefix : <http://example.org/>
prefix xsd: <http://www.w3.org/2001/XMLSchema#>
prefix ui: <http://www.w3.org/ns/ui#>

start = @:User

:User {
 schema:name  xsd:string                    // ui:label “Name" ;
 schema:birthDate xsd:dateTime              // ui:label "Birth date" ;
 schema:gender [ schema:Male schema:Female ] // ui:label "Gender" ;
}
```

Shapes

Validates/describes

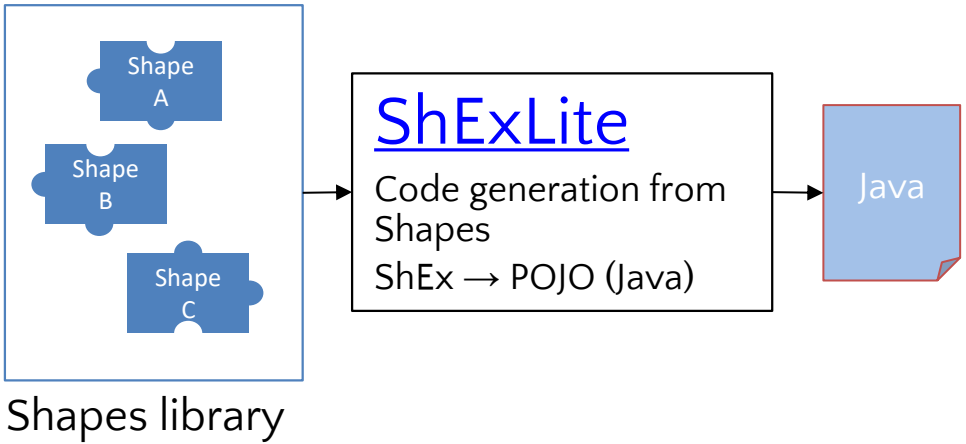
RDF data

Try it:
Eric’s demo
https://rdfshape.weso.es/link/16480473402

Generating code from shapes

Generate domain model from shapes

Java code generation (POJOs) from those shapes

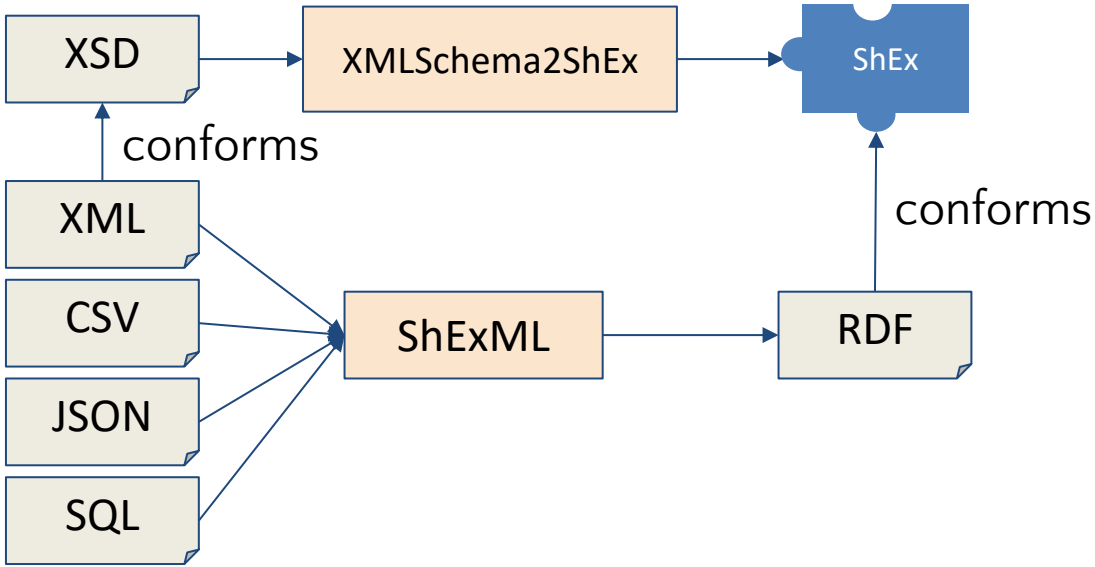


Shapes for data integration

XMLSchema2ShEx: Convert XML Schemas to shapes

ShExML: Domain specific language to convert data to RDF

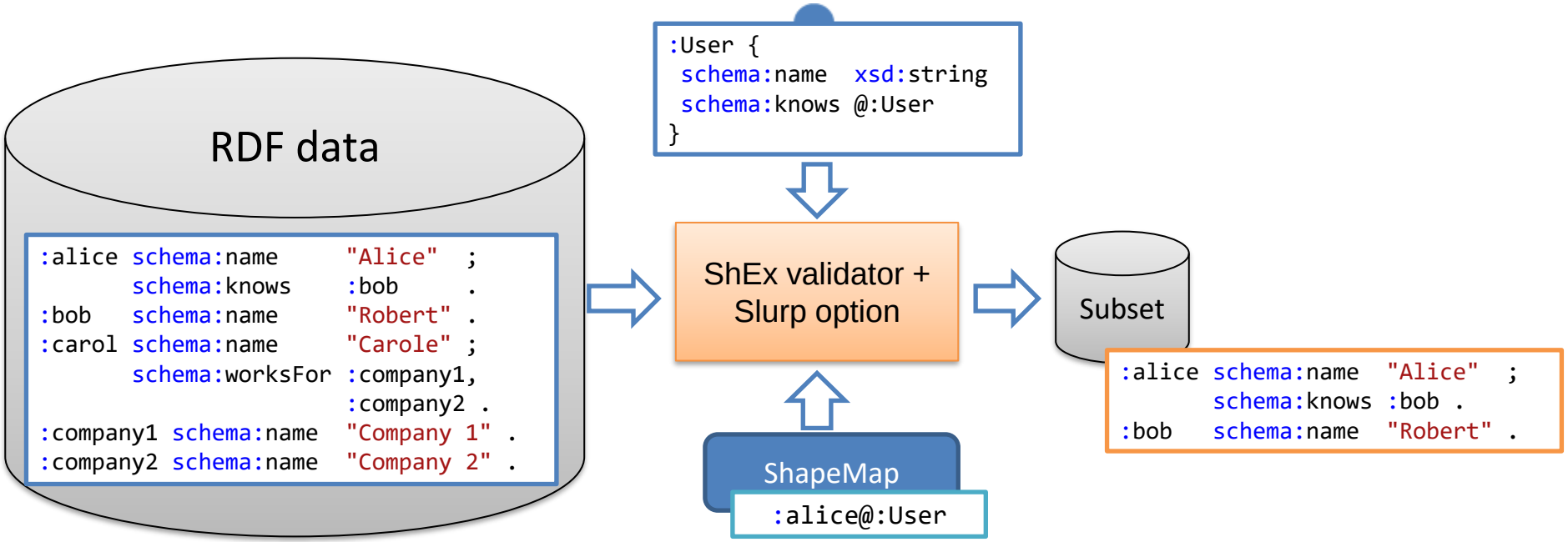
Input formats: CSV, XML, JSON, SQL



Subsetting based on Shapes

Generate subsets from ShEx

Slurp option: when validating, collect visited nodes/triples



Some ShEx use cases and future work

Wikidata, FHIR, SOLID

Wikidata

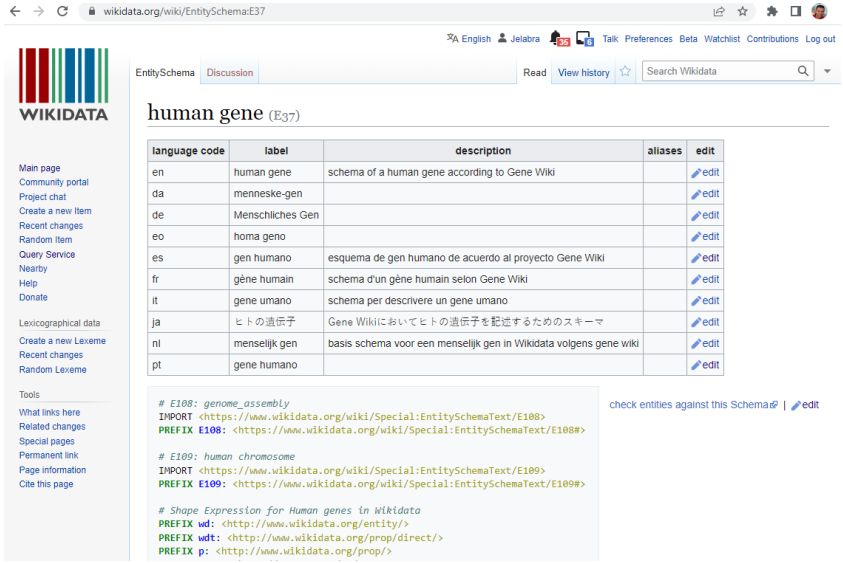
In May, 2019, Wikidata announced ShEx adoption

New namespace for schemas

Example (Human Gene): https://www.wikidata.org/wiki/EntitySchema:E37

Wikibase also contains entity schemas

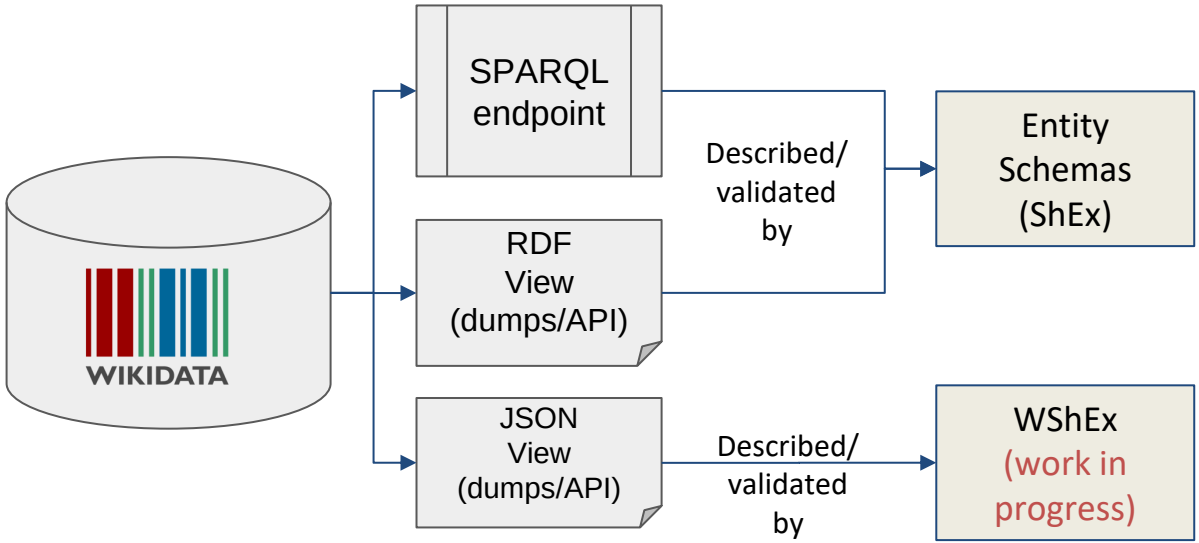
Online demo: wikishape



Wikidata entity schemas

Entity schemas describe RDF view

Available through SPARQL endpoint, RDF dumps



Shapes ecosystems

Wikidata provides a whole ShEx ecosystem

Entity schemas evolve and relate between each other

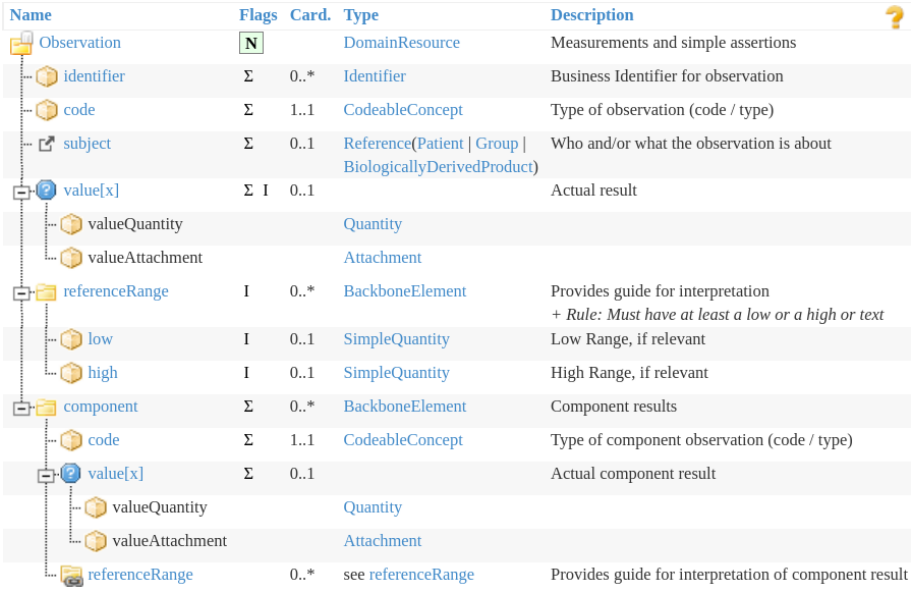
- Directory: https://www.wikidata.org/wiki/Wikidata:Database_reports/EntitySchema_directory

Some challenges

- Different schemas for the same type of entities?
  - Some schemas stress some aspects while others stress others
- Evolution of schemas
- Searching entity schemas

FHIR use case

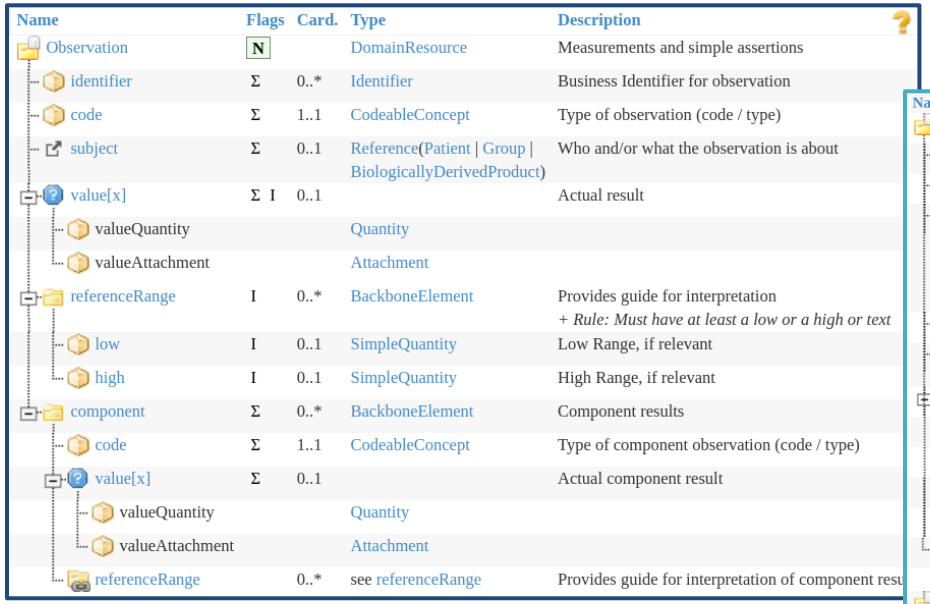
- Clinical information model
  - Observations
  - Procedures
  - Medications
  - + 150 other things
- Custom schema language
  - Use-case tailored expressivity
  - Available as JSON structure
    - (not JSON-schema)
- Developer-friendly
  - Pretty HTML
- e.g. Observation
  - https://hl7.org/fhir/observation#resource



Name	Flags	Card.	Type	Description
Observation	N		DomainResource	Measurements and simple assertions
identifier	Σ	0..*	Identifier	Business Identifier for observation
code	Σ	1..1	CodeableConcept	Type of observation (code / type)
subject	Σ	0..1	Reference(Patient | Group | BiologicallyDerivedProduct)	Who and/or what the observation is about
value[x]	Σ I	0..1		Actual result
valueQuantity			Quantity	
valueAttachment			Attachment	
referenceRange	I	0..*	BackboneElement	Provides guide for interpretation *+ Rule: Must have at least a low or a high or text*
low	I	0..1	SimpleQuantity	Low Range, if relevant
high	I	0..1	SimpleQuantity	High Range, if relevant
component	Σ	0..*	BackboneElement	Component results
code	Σ	1..1	CodeableConcept	Type of component observation (code / type)
value[x]	Σ	0..1		Actual component result
valueQuantity			Quantity	
valueAttachment			Attachment	
referenceRange		0..*	see referenceRange	Provides guide for interpretation of component result

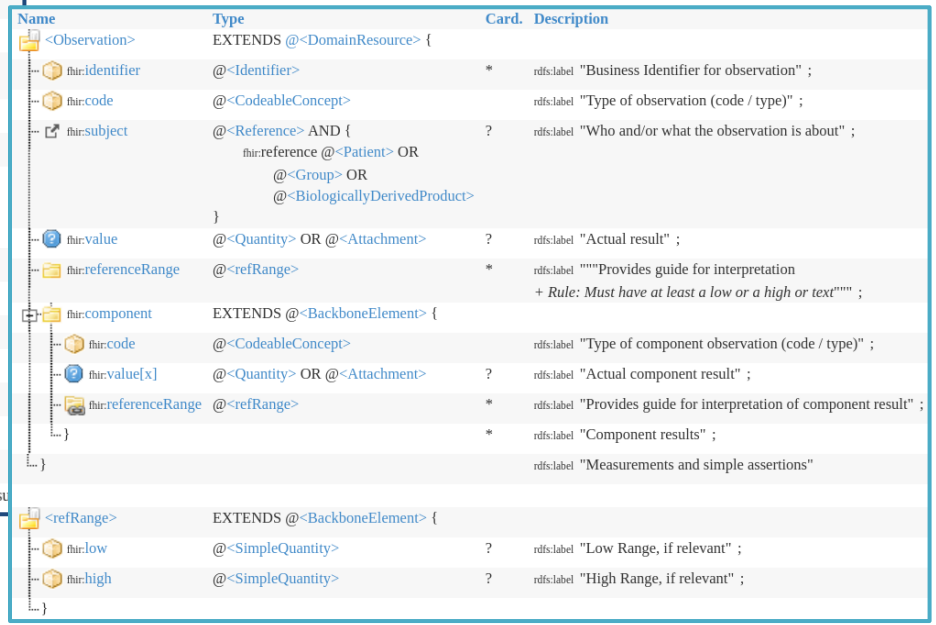
FHIR Observation

Standard FHIR tree representation…



Name	Flags	Card.	Type	Description
Observation	N		DomainResource	Measurements and simple assertions
identifier	Σ	0..*	Identifier	Business Identifier for observation
code	Σ	1..1	CodeableConcept	Type of observation (code / type)
subject	Σ	0..1	Reference(Patient | Group | BiologicallyDerivedProduct)	Who and/or what the observation is about
value[x]	Σ I	0..1		Actual result
valueQuantity			Quantity	
valueAttachment			Attachment	
referenceRange	I	0..*	BackboneElement	Provides guide for interpretation *+ Rule: Must have at least a low or a high or text*
low	I	0..1	SimpleQuantity	Low Range, if relevant
high	I	0..1	SimpleQuantity	High Range, if relevant
component	Σ	0..*	BackboneElement	Component results
code	Σ	1..1	CodeableConcept	Type of component observation (code / type)
value[x]	Σ	0..1		Actual component result
valueQuantity			Quantity	
valueAttachment			Attachment	
referenceRange		0..*	see referenceRange	Provides guide for interpretation of component resu

…expressed as "Graphical" ShEx



Name	Type	Card.	Description
<Observation>	EXTENDS @<DomainResource> {		
fhir:identifier	@<Identifier>	*	rdfs:label "Business Identifier for observation" ;
fhir:code	@<CodeableConcept>		rdfs:label "Type of observation (code / type)" ;
fhir:subject	@<Reference> AND { fhir:reference @<Patient> OR @<Group> OR @<BiologicallyDerivedProduct> }	?	rdfs:label "Who and/or what the observation is about" ;
fhir:value	@<Quantity> OR @<Attachment>	?	rdfs:label "Actual result" ;
fhir:referenceRange	@<refRange>	*	rdfs:label """Provides guide for interpretation *+ Rule: Must have at least a low or a high or text*""" ;
fhir:component	EXTENDS @<BackboneElement> {		
fhir:code	@<CodeableConcept>		rdfs:label "Type of component observation (code / type)" ;
fhir:value[x]	@<Quantity> OR @<Attachment>	?	rdfs:label "Actual component result" ;
fhir:referenceRange	@<refRange>	*	rdfs:label "Provides guide for interpretation of component result" ;
}		*	rdfs:label "Component results" ;
}			rdfs:label "Measurements and simple assertions"
<refRange>	EXTENDS @<BackboneElement> {		
fhir:low	@<SimpleQuantity>	?	rdfs:label "Low Range, if relevant" ;
fhir:high	@<SimpleQuantity>	?	rdfs:label "High Range, if relevant" ;
}			

Name	Flags	Card.	Type	Description
Observation	N		DomainResource	Measurements and simple assertions
identifier	Σ	0..*	Identifier	Business Identifier for observation
code	Σ	1..1	CodeableConcept	Type of observation (code / type)
subject	Σ	0..1	Reference(Patient | Group | BiologicallyDerivedProduct)	Who and/or what the observation is about
value[x]	Σ I	0..1		Actual result
valueQuantity			Quantity	
valueAttachment			Attachment	
referenceRange				
low				
high				
component				
code				
value[x]				
valueQuantity				
valueAttachment				
referenceRange				

Name	Type	Card.	Description
<Observation>	EXTENDS @<DomainResource> {		
fhir:identifier	@<Identifier>	*	rdfs:label "Business Identifier for observation" ;
fhir:code	@<CodeableConcept>		rdfs:label "Type of observation (code / type)" ;
fhir:subject	@<Reference> AND { fhir:reference @<Patient> OR @<Group> OR @<BiologicallyDerivedProduct> }	?	rdfs:label "Who and/or what the observation is about" ;
fhir:value	@<Quantity> OR @<Attachment>	?	rdfs:label "Actual result" ;
fhir:referenceRange	@<refRange>	*	rdfs:label """Provides guide for interpretation *+ Rule: Must have at least a low or a high or text*""" ;
fhir:component	EXTENDS @<BackboneElement> {		
fhir:code	@<CodeableConcept>		rdfs:label "Type of component observation (code / type)" ;

FHIR/RDF ShEx

- tree-view *is* machine-readable schema
- maps to intuitions
- captures use-case semantics (closed, slicing)

```
<Observation> EXTENDS @<DomainResource> CLOSED {
  fhir:identifier       @<Identifier> *          // rdfs:label "Business Identifier for observation";
  fhir:code             @<CodeableConcept>       // rdfs:label "Type of observation (code / type)";
  fhir:subject          @<Reference> AND {
    fhir:reference      @<Patient> OR @<Group> OR @<BiologicallyDerivedProduct>
  } ?                                            // rdfs:label "Who and/or what the observation is about";
  fhir:value            @<Quantity> OR @<Attachment> ?  // rdfs:label "Actual result";
  fhir:referenceRange   @<refRange> *            // rdfs:label """Provides guide for interpretation
                                                    + Rule: Must have at least a low or a high or text""";
  fhir:component        EXTENDS @<BackboneElement> CLOSED {
    fhir:code           @<CodeableConcept>       // rdfs:label "Type of component observation (code / type)";
    fhir:value          @<Quantity> OR @<Attachment> ? // rdfs:label "Actual component result";
    fhir:referenceRange @<refRange> *            // rdfs:label "Provides guide to interpret component result";
  } *                                            // rdfs:label "Component results";
}                                                // rdfs:label "Measurements and simple assertions"
<refRange> EXTENDS @<BackboneElement> CLOSED {
  fhir:low      @<SimpleQuantity> ?              // rdfs:label "Low Range, if relevant";
  fhir:high     @<SimpleQuantity> ?              // rdfs:label "High Range, if relevant";
}
```

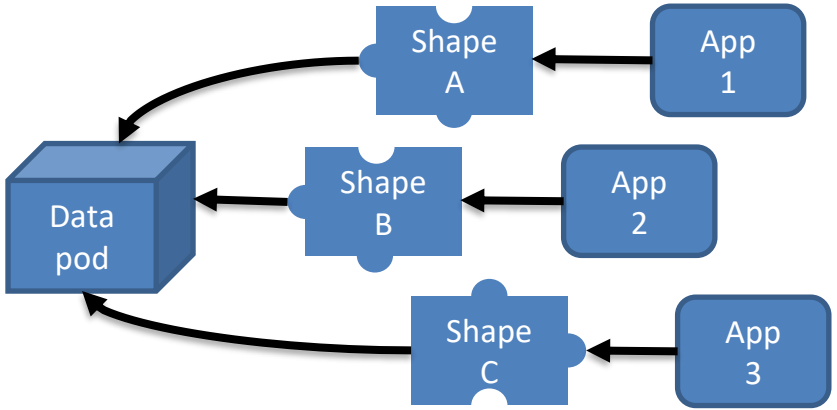
Solid project

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/

Shape trees for SOLID

- Map graphs in a resource hierarchy to Shapes
- Shape trees provides a "typing" for a filesystem
- Deliverable of the Solid Application Interoperability Task Force
- Provide validation on create, update and delete operations

https://shapetrees.org/TR/specification/

End of presentation