Declarative Construction and Validation of Knowledge Graphs

Half-day tutorial at K-CAP23

Ana Iglesias-Molina[1] and Xuemin Duan[2]

[1]Ontology Engineering Group (OEG) – Universidad Politécnica de Madrid
[2]Declarative Languages and Artificial Intelligence (DTAI) – KU Leuven

Agenda

- 13:00 – 13:10 Introduction
- 13:10 – 15:00 Declarative Knowledge Graph Construction
- 15:00 – 15:30 Break
- 15:30 – 17:20 Declarative Knowledge Graph Validation
- 17:20 – 17:30 Conclusions

Knowledge Graph Validation

15:30 - 17:20

Outline:

- SHACL background
- Write SHACL by hand
- RML2SHACL

SHACL Background

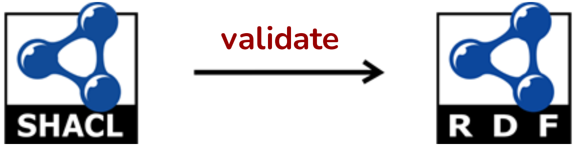
SHACL and ShEx

- Shapes Constraint Language (SHACL)
  - developed by the W3C RDF Data Shapes Working Group with the goal to "produce a language for defining structural constraints on RDF graphs."
  - the first public draft was published in 2015
  - proposed as a W3C Recommendation in 2017
- Shapes Expression Language (ShEx)
  - developed in late 2013 with the goal to provide a human-readable syntax for OSLC Resource Shapes.

SHACL

- validating RDF graphs against a set of constraints in shapes expressed in terms of RDF

- SHACL processor validates whether the nodes in RDF satisfies the constraints and return validation report



SHACL Concepts

Data Graph; Shapes Graph; Node Shape; Property Shape; target declarations; focus node; property path, property value; constraints

Shapes Graph and Data Graph

- an RDF graph containing shapes defining constriants
- RDF graphs that are validated

```
:StudentShape a sh:NodeShape ;
  sh:targetClass :Student ;
  sh:nodeKind sh:IRI ;
  sh:property :NameShape .

:NameShape a sh:PropertyShape ;
  sh:path :name ;
  sh:datatype xsd:string ;
  sh:minCount 1.
```

Shapes Graph and Data Graph

- an RDF graph containing shapes defining constriants
- RDF graphs that are validated

```
:r001 a :Student ;    #passes
  :name “Alice” .

:r002 a :Student ;    #passes
  :name “Bob” ;
  :id “r002” .

:r003 a :Student ;    #fails
  :id “r003” .
```

Shapes Graph and Data Graph

- an RDF graph containing shapes defining constriants
- RDF graphs that are validated

```
:StudentShape a sh:NodeShape ;
  sh:targetClass :Student ;
  sh:nodeKind sh:IRI ;
  sh:property :NameShape .

:NameShape a sh:PropertyShape ;
  sh:path :name ;
  sh:datatype xsd:string ;
  sh:minCount 1.
```

```
:r001 a :Student ;    #passes
  :name “Alice” .

:r002 a :Student ;    #passes
  :name “Bob” ;
  :id “r002” .

:r003 a :Student ;    #fails
  :id “r003” .
```

Node Shape, Target Declaration, and Focus Node

- specify constraints that need to be met with respect to focus nodes declared by target

```
:StudentShape a sh:NodeShape ;
  sh:targetClass :Student ;
  sh:nodeKind sh:IRI ;
  sh:property :NameShape .

:NameShape a sh:PropertyShape ;
  sh:path :name ;
  sh:datatype xsd:string ;
  sh:minCount 1.
```

```
:r001 a :Student ;    #passes
  :name “Alice” .

:r002 a :Student ;    #passes
  :name “Bob” ;
  :id “r002” .

:r003 a :Student ;    #fails
  :id “r003” .
```

Target Declaration and Focus Node

- sh:targetClass

```
:StudentShape a sh:NodeShape ;
  sh:targetClass :Student ;
  sh:nodeKind sh:IRI.
```

```
:r001 a :Student ;#passes
  :name “Alice” .
```

- sh:targetNode

```
:StudentShape a sh:NodeShape ;
  sh:targetNode :r001 ;
  sh:nodeKind sh:IRI.
```

```
:r001 a :Student ;#passes
  :name “Alice” .
```

- sh:targetSubjectsOf

```
:StudentShape a sh:NodeShape ;
  sh:targetSubjectsOf :name ;
  sh:nodeKind sh:IRI.
```

```
:r001 a :Student ;#passes
  :name “Alice” .
```

- sh:targetObjectsOf

```
:StudentShape a sh:NodeShape ;
  sh:targetObjectsOf :name ;
  sh:nodeKind sh:IRI.
```

```
:r001 a :Student ; #fails
  :name “Alice” .
```

Property Shape, Property Path, and Property Value

- primarily apply to the property value

- reached by focus node vis property path

```
:StudentShape a sh:NodeShape ;
  sh:targetClass :Student ;
  sh:nodeKind sh:IRI ;
  sh:property :NameShape .

:NameShape a sh:PropertyShape ;
  sh:path :name ;
  sh:datatype xsd:string ;
  sh:minCount 1.
```

```
:r001 a :Student ;   #passes
  :name “Alice” .

:r002 a :Student ;   #passes
  :name “Bob” ;
  :id “r001” .

:u001 a :Teacher ;   #passes
  :name “Carol” .
```

No Target No Validation?

```
:StudentShape a sh:NodeShape ;
  sh:targetClass :Student ;
  sh:nodeKind sh:IRI .
```

```
:StudentShape a sh:NodeShape ;
  sh:targetClass :Student ;
  sh:nodeKind sh:IRI ;
  sh:property :NameShape .

:NameShape a sh:PropertyShape ;
  sh:path :name ;
  sh:datatype xsd:string ;
  sh:minCount 1.
```

```
:StudentShape a sh:NodeShape ;

  sh:nodeKind sh:IRI .
```

```
:StudentShape a sh:NodeShape ;

  sh:nodeKind sh:IRI ;
  sh:property :NameShape .

:NameShape a sh:PropertyShape ;
  sh:path :name ;
  sh:datatype xsd:string ;
  sh:minCount 1.
```

SHACL Constraints

- can be divided into
  - **SHACL Core**
  - **SHACL-SPARQL**

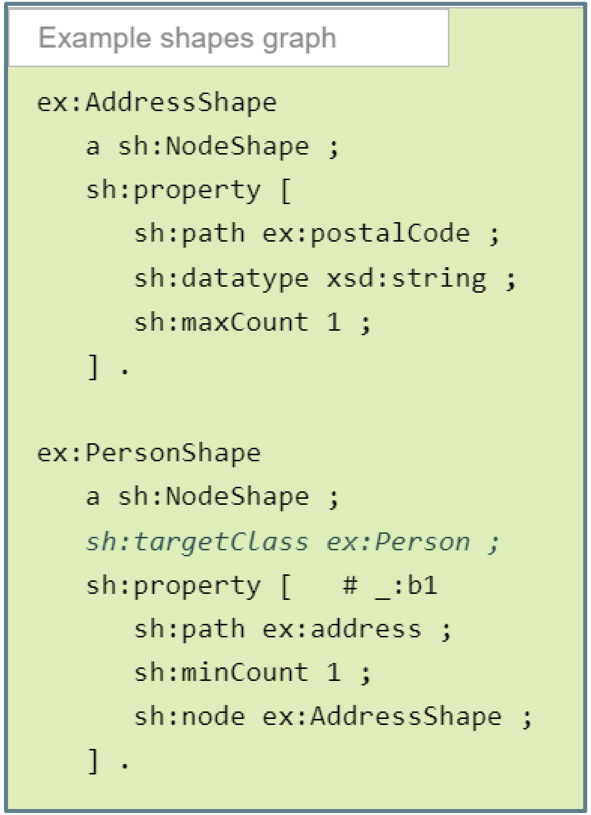
  Some works extend SHACL with (a) advanced features such as rules and complex expressions (called SHACL-Javascript).

Built-in SHACL Core Constraints

Components	**Parameters**
Value types	sh:class, sh:datatype, sh:nodeKind
Cardinality	sh:minCount, sh:maxCount
Value range	sh:minExclusive, sh:minInclusive, sh:minExclusive, sh:minInclusive
String-based	sh:minLength, sh:maxLength, sh:pattern, sh:languageIn,sh:uniqueLang
Property Pair	sh:equals, sh:disjoint, sh:lessThan, sh:lessThanOrEquals
Logical	sh:not, sh:and, sh:or, sh:xone
Shape-based	sh:node, sh:property, sh:qualifiedValueShape, sh:qualifiedMinCount, sh:qualifiedMaxCount
Other	sh:closed, sh:ignoredProperties, sh:hasValue, sh:in
Non-validating	sh:name, sh:description, sh:order, sh:group

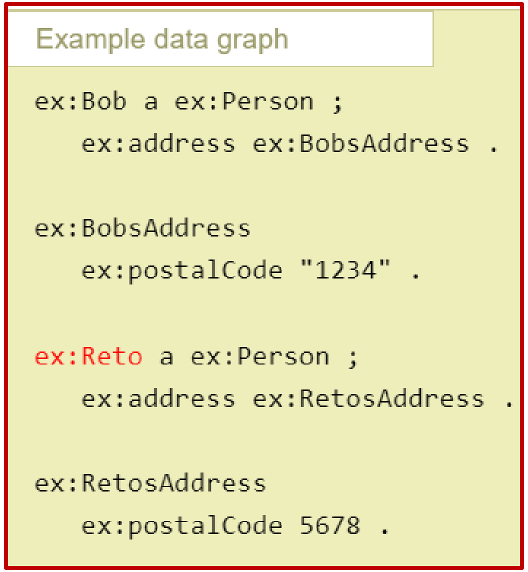
https://www.w3.org/TR/shacl/

sh:node & sh:property



Example shapes graph

```
ex:AddressShape
   a sh:NodeShape ;
   sh:property [
      sh:path ex:postalCode ;
      sh:datatype xsd:string ;
      sh:maxCount 1 ;
   ] .

ex:PersonShape
   a sh:NodeShape ;
   sh:targetClass ex:Person ;
   sh:property [   # _:b1
      sh:path ex:address ;
      sh:minCount 1 ;
      sh:node ex:AddressShape ;
   ] .
```

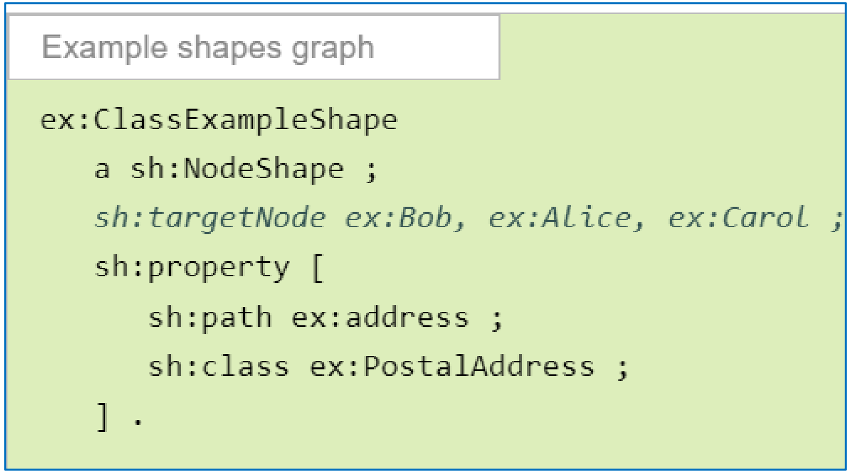


Example data graph

```
ex:Bob a ex:Person ;
   ex:address ex:BobsAddress .

ex:BobsAddress
   ex:postalCode "1234" .

ex:Reto a ex:Person ;
   ex:address ex:RetosAddress .

ex:RetosAddress
   ex:postalCode 5678 .
```

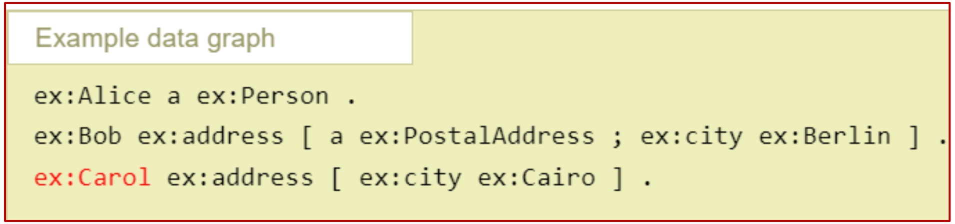
https://www.w3.org/TR/shacl/

sh:class



Example shapes graph

```
ex:ClassExampleShape
    a sh:NodeShape ;
    sh:targetNode ex:Bob, ex:Alice, ex:Carol ;
    sh:property [
        sh:path ex:address ;
        sh:class ex:PostalAddress ;
    ] .
```

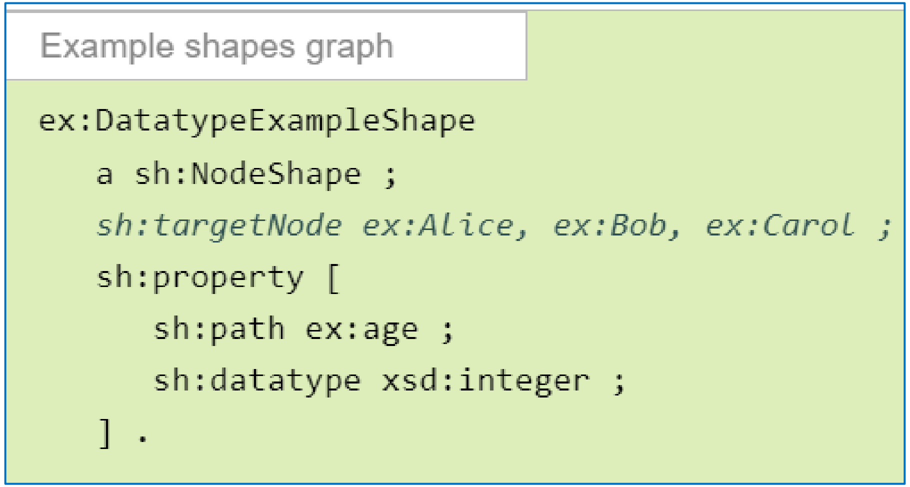


Example data graph

```
ex:Alice a ex:Person .
ex:Bob ex:address [ a ex:PostalAddress ; ex:city ex:Berlin ] .
ex:Carol ex:address [ ex:city ex:Cairo ] .
```

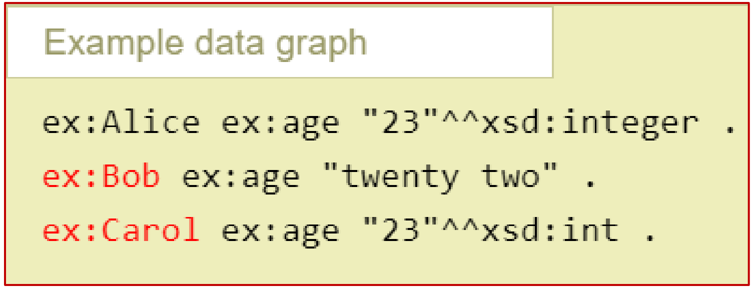
https://www.w3.org/TR/shacl/

sh:datatype



Example shapes graph

```
ex:DatatypeExampleShape
	a sh:NodeShape ;
	sh:targetNode ex:Alice, ex:Bob, ex:Carol ;
	sh:property [
		sh:path ex:age ;
		sh:datatype xsd:integer ;
	] .
```

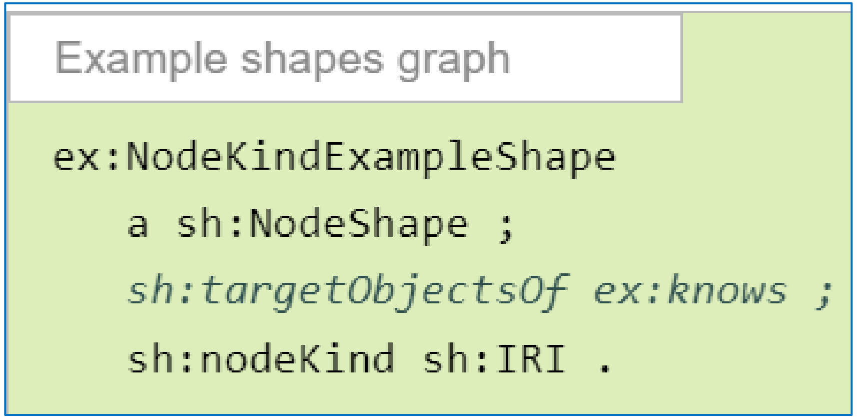


Example data graph

```
ex:Alice ex:age "23"^^xsd:integer .
ex:Bob ex:age "twenty two" .
ex:Carol ex:age "23"^^xsd:int .
```

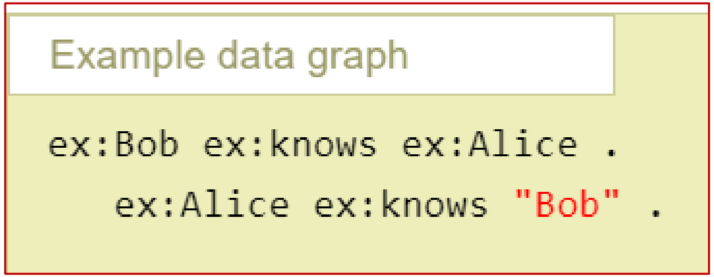
https://www.w3.org/TR/shacl/

sh:nodeKind



Example shapes graph

```
ex:NodeKindExampleShape
    a sh:NodeShape ;
    sh:targetObjectsOf ex:knows ;
    sh:nodeKind sh:IRI .
```

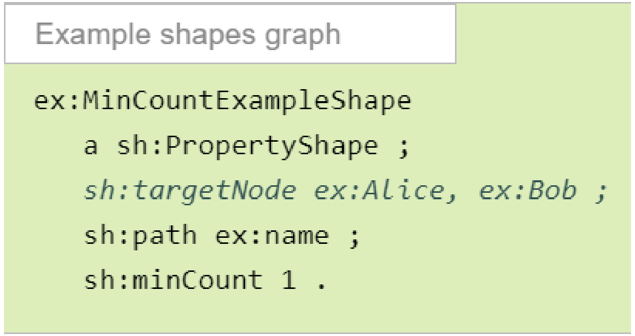


Example data graph

```
ex:Bob ex:knows ex:Alice .
   ex:Alice ex:knows "Bob" .
```

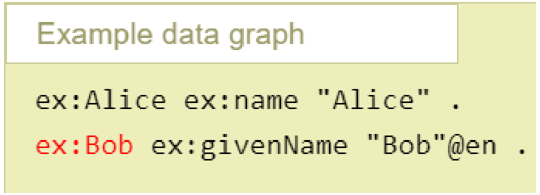
https://www.w3.org/TR/shacl/

sh:minCount & sh:maxCount



Example shapes graph

```
ex:MinCountExampleShape
    a sh:PropertyShape ;
    sh:targetNode ex:Alice, ex:Bob ;
    sh:path ex:name ;
    sh:minCount 1 .
```

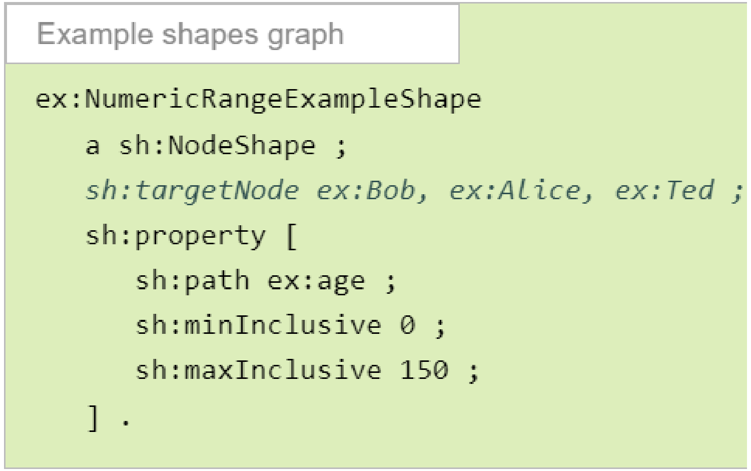


Example data graph

```
ex:Alice ex:name "Alice" .
ex:Bob ex:givenName "Bob"@en .
```

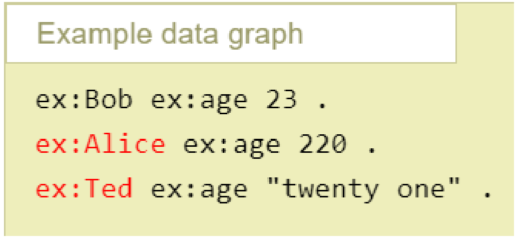
https://www.w3.org/TR/shacl/

sh:minExclusive & sh:minInclusive & sh:minExclusive & sh:minInclusive



Example shapes graph

```
ex:NumericRangeExampleShape
   a sh:NodeShape ;
   sh:targetNode ex:Bob, ex:Alice, ex:Ted ;
   sh:property [
      sh:path ex:age ;
      sh:minInclusive 0 ;
      sh:maxInclusive 150 ;
   ] .
```

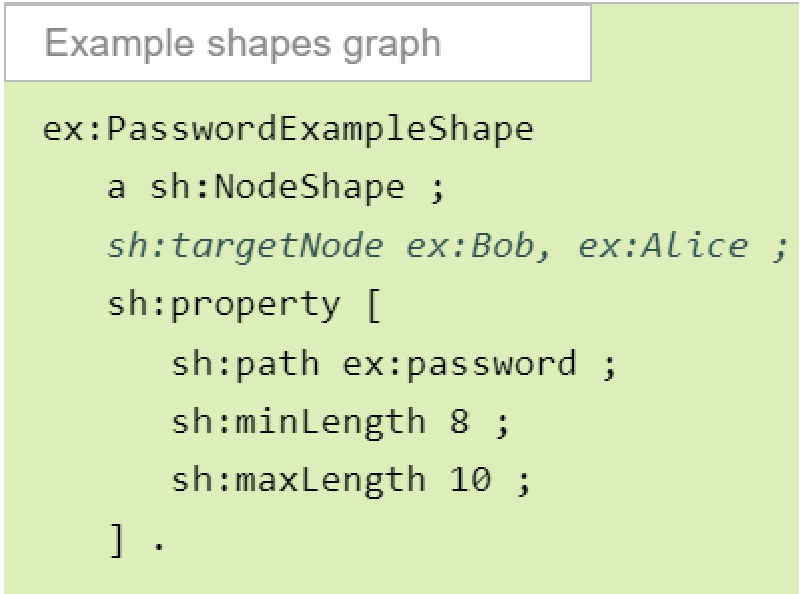


Example data graph

```
ex:Bob ex:age 23 .
ex:Alice ex:age 220 .
ex:Ted ex:age "twenty one" .
```

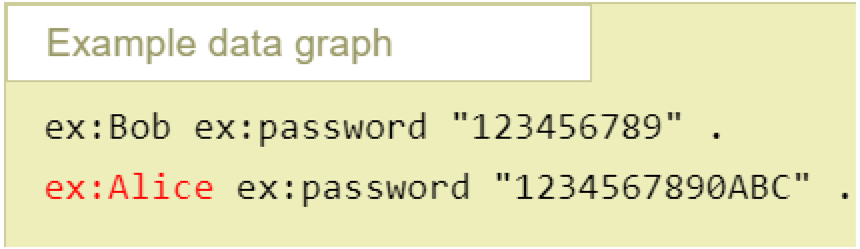
https://www.w3.org/TR/shacl/

sh:minLength & sh:maxLength



Example shapes graph

```
ex:PasswordExampleShape
    a sh:NodeShape ;
    sh:targetNode ex:Bob, ex:Alice ;
    sh:property [
        sh:path ex:password ;
        sh:minLength 8 ;
        sh:maxLength 10 ;
    ] .
```

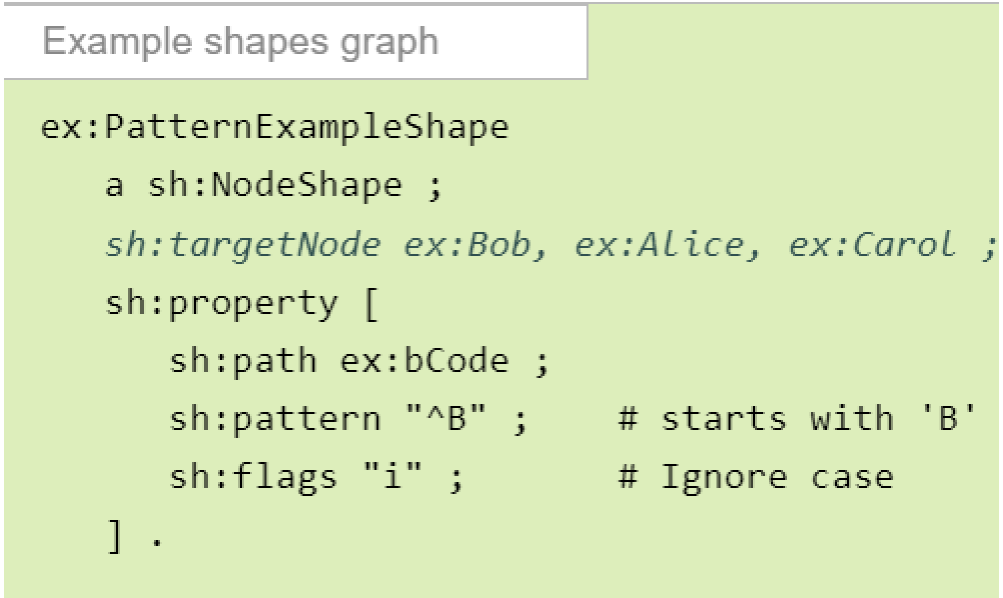


Example data graph

```
ex:Bob ex:password "123456789" .
ex:Alice ex:password "1234567890ABC" .
```

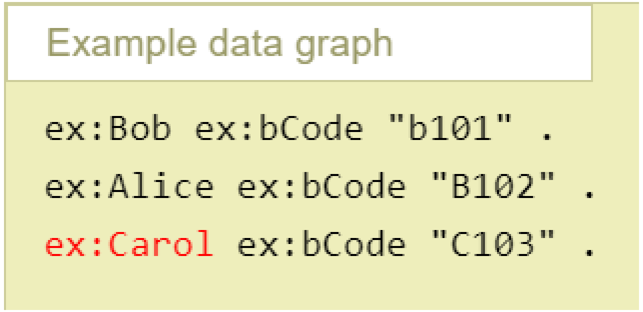
https://www.w3.org/TR/shacl/

sh:pattern



Example shapes graph

```
ex:PatternExampleShape
    a sh:NodeShape ;
    sh:targetNode ex:Bob, ex:Alice, ex:Carol ;
    sh:property [
        sh:path ex:bCode ;
        sh:pattern "^B" ;     # starts with 'B'
        sh:flags "i" ;        # Ignore case
    ] .
```

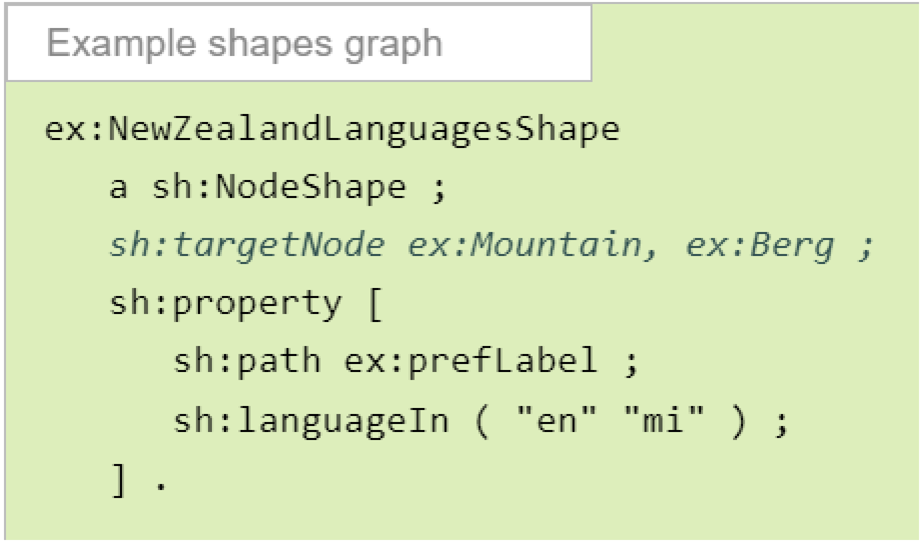


Example data graph

```
ex:Bob ex:bCode "b101" .
ex:Alice ex:bCode "B102" .
ex:Carol ex:bCode "C103" .
```

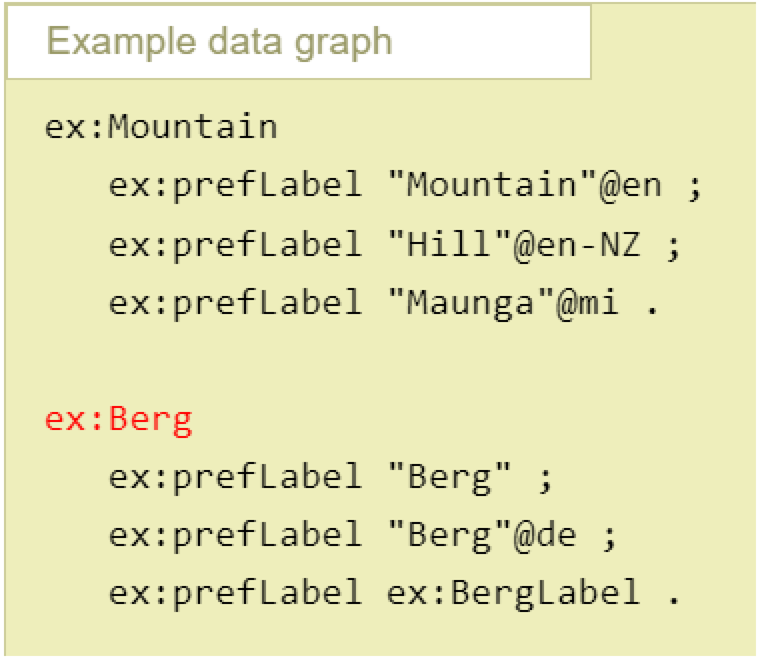
https://www.w3.org/TR/shacl/

sh:languageIn & sh:uniqueLang



Example shapes graph

```
ex:NewZealandLanguagesShape
   a sh:NodeShape ;
   sh:targetNode ex:Mountain, ex:Berg ;
   sh:property [
      sh:path ex:prefLabel ;
      sh:languageIn ( "en" "mi" ) ;
   ] .
```

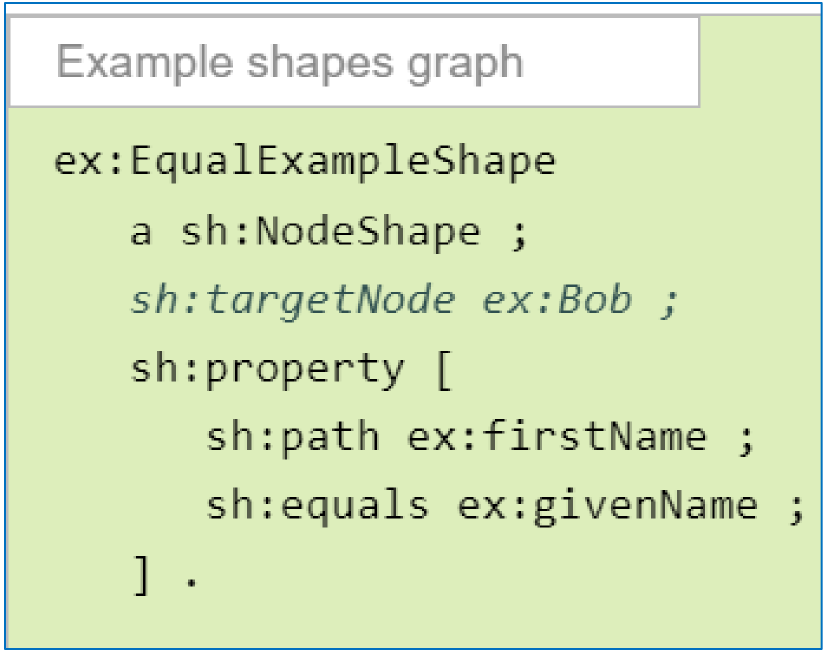


Example data graph

```
ex:Mountain
   ex:prefLabel "Mountain"@en ;
   ex:prefLabel "Hill"@en-NZ ;
   ex:prefLabel "Maunga"@mi .

ex:Berg
   ex:prefLabel "Berg" ;
   ex:prefLabel "Berg"@de ;
   ex:prefLabel ex:BergLabel .
```

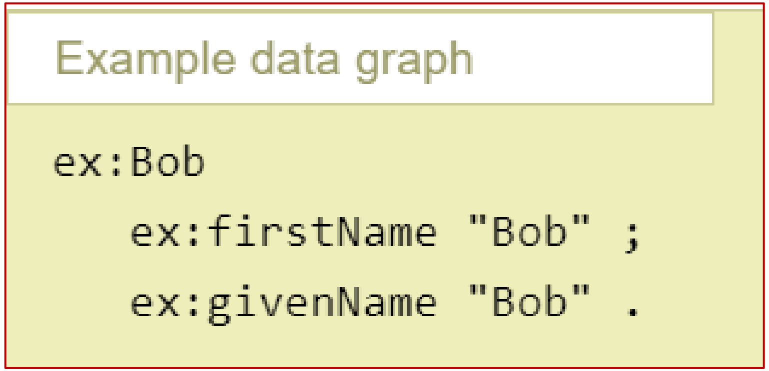
https://www.w3.org/TR/shacl/

sh:equals & disjoint



Example shapes graph

```
ex:EqualExampleShape
	a sh:NodeShape ;
	sh:targetNode ex:Bob ;
	sh:property [
		sh:path ex:firstName ;
		sh:equals ex:givenName ;
	] .
```

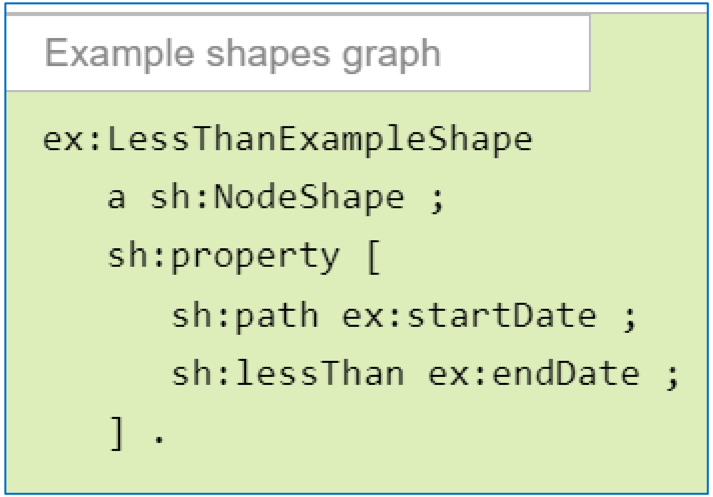


Example data graph

```
ex:Bob
	ex:firstName "Bob" ;
	ex:givenName "Bob" .
```

https://www.w3.org/TR/shacl/

sh:lessThan & sh:lessThanOrEquals



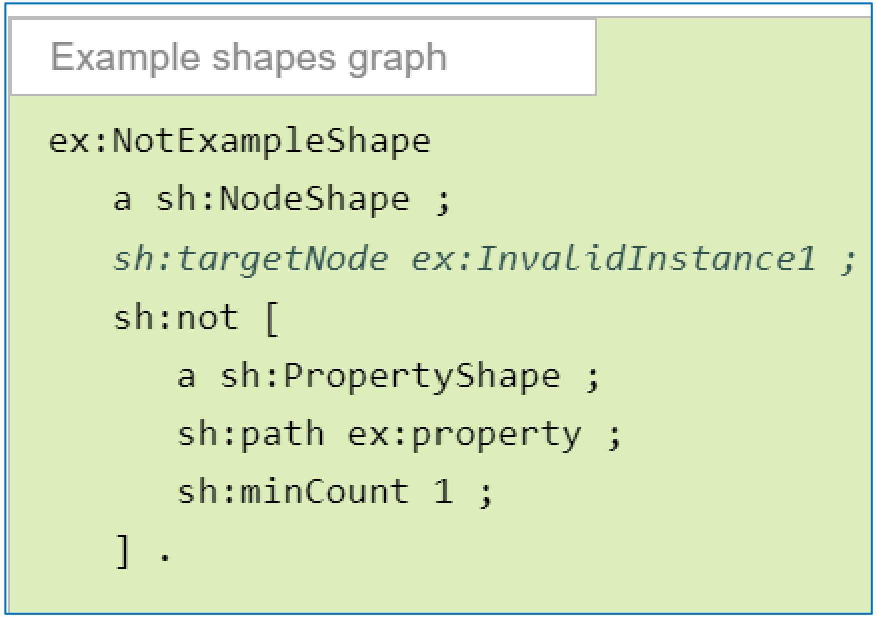
Example shapes graph

```
ex:LessThanExampleShape
   a sh:NodeShape ;
   sh:property [
      sh:path ex:startDate ;
      sh:lessThan ex:endDate ;
   ] .
```

```
:r001 a :Example ;                    #passes
   ex:startDate "2017-04-20T20:00:00"^^xsd:dateTime ;
   ex:endDate "2017-04-20T21:30:00"^^xsd:dateTime ;.
```

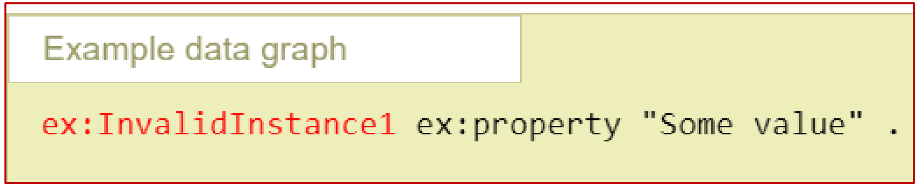
https://www.w3.org/TR/shacl/

sh:not & sh:and & sh:or & sh:xone



Example shapes graph

```
ex:NotExampleShape
    a sh:NodeShape ;
    sh:targetNode ex:InvalidInstance1 ;
    sh:not [
        a sh:PropertyShape ;
        sh:path ex:property ;
        sh:minCount 1 ;
    ] .
```

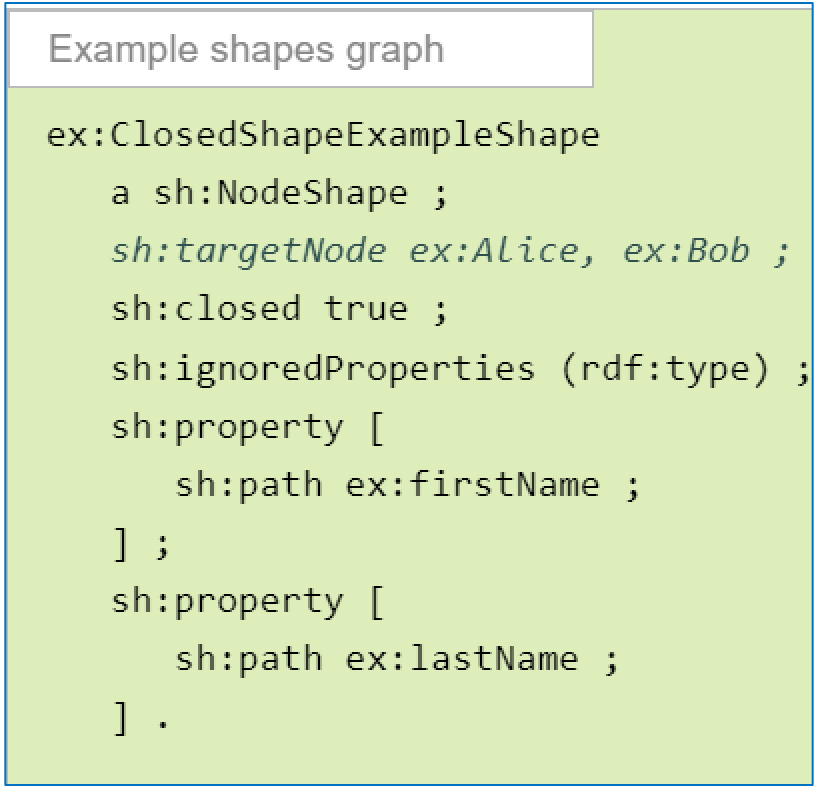


Example data graph

```
ex:InvalidInstance1 ex:property "Some value" .
```

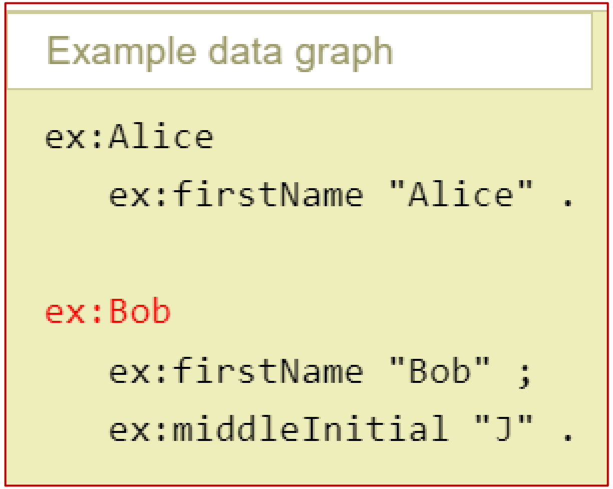
https://www.w3.org/TR/shacl/

sh:close & sh:ignoredProperties



Example shapes graph

```
ex:ClosedShapeExampleShape
    a sh:NodeShape ;
    sh:targetNode ex:Alice, ex:Bob ;
    sh:closed true ;
    sh:ignoredProperties (rdf:type) ;
    sh:property [
        sh:path ex:firstName ;
    ] ;
    sh:property [
        sh:path ex:lastName ;
    ] .
```

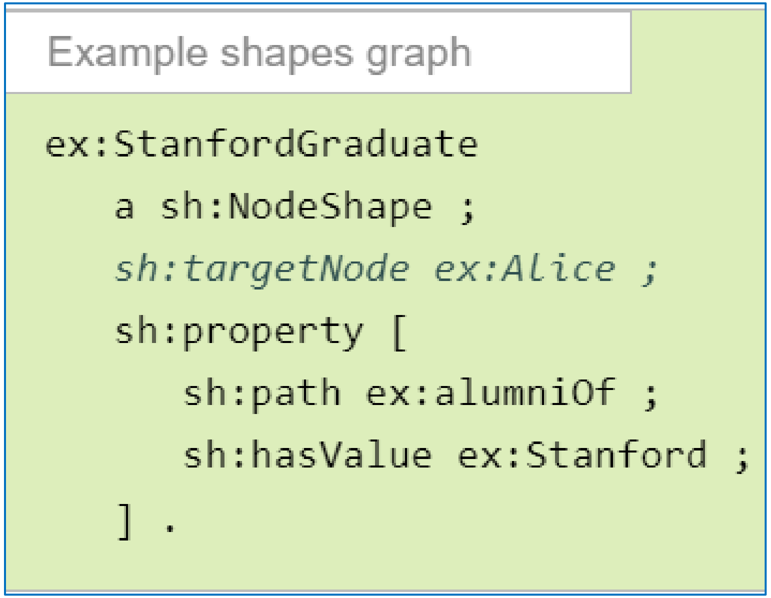


Example data graph

```
ex:Alice
    ex:firstName "Alice" .

ex:Bob
    ex:firstName "Bob" ;
    ex:middleInitial "J" .
```

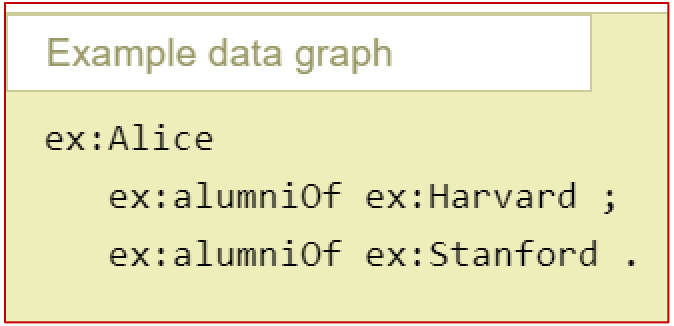
https://www.w3.org/TR/shacl/

sh:hasValue & sh:in



Example shapes graph

```
ex:StanfordGraduate
    a sh:NodeShape ;
    sh:targetNode ex:Alice ;
    sh:property [
        sh:path ex:alumniOf ;
        sh:hasValue ex:Stanford ;
    ] .
```



Example data graph

```
ex:Alice
    ex:alumniOf ex:Harvard ;
    ex:alumniOf ex:Stanford .
```

SHACL syntax (SHACL-SHACL)

- Target declarations are optional for node shape and property shape

- Property shape must have exactly one property path (i.e. sh:path)

- Each shape has at most one sh:datatype, sh:nodeKind , …

- Each shape can have multiple sh:class, sh:and, …

Validation Report

- identified by sh:ValidationReport

- has exactly one sh:conforms (true/false)

- optional has sh:result if False

```
:report a sh:ValidationReport ;
  sh:conforms true .
```

```
:report a sh:ValidationReport ;
  sh:conforms false ;
  sh:result
    [a sh:ValidationResult;
     sh:resultSeverity sh:Violation ;
     sh:sourceConstraintComponent
sh:DatatypeConstraintComponent ;
     sh:sourceShape ... ;
     sh:focusNode :r001 ;
     sh:value 2000;
     sh:resultPath :name ;
     sh:resultMessage "Value does not
have datatype xsd:string" ],
```

Hands-on exercise

Write SHACL shapes by hand

SHACL validation.ipynb
https://colab.research.google.com/drive/1n8EAjS7Yq022JF8z067hIA6CCXHIni7T?usp=drive_link

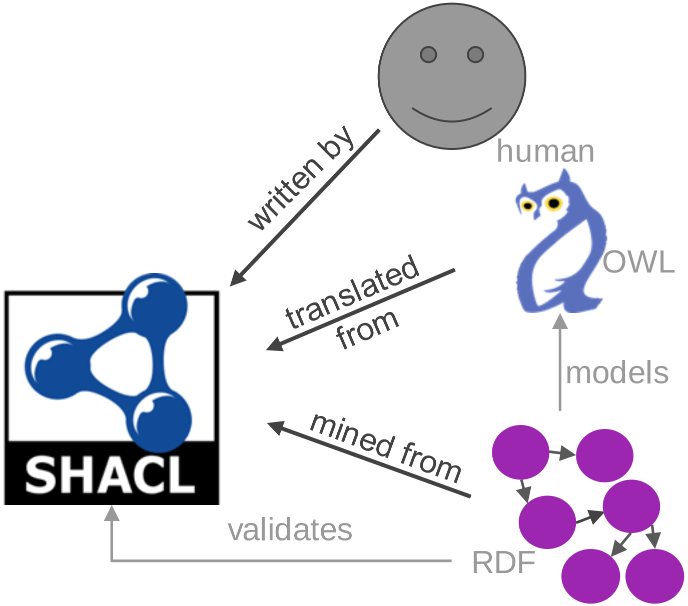
- Play with several simple tasks

- Learn to use SHACL-SHACL to validate whether the created SHACL shapes are well-formed

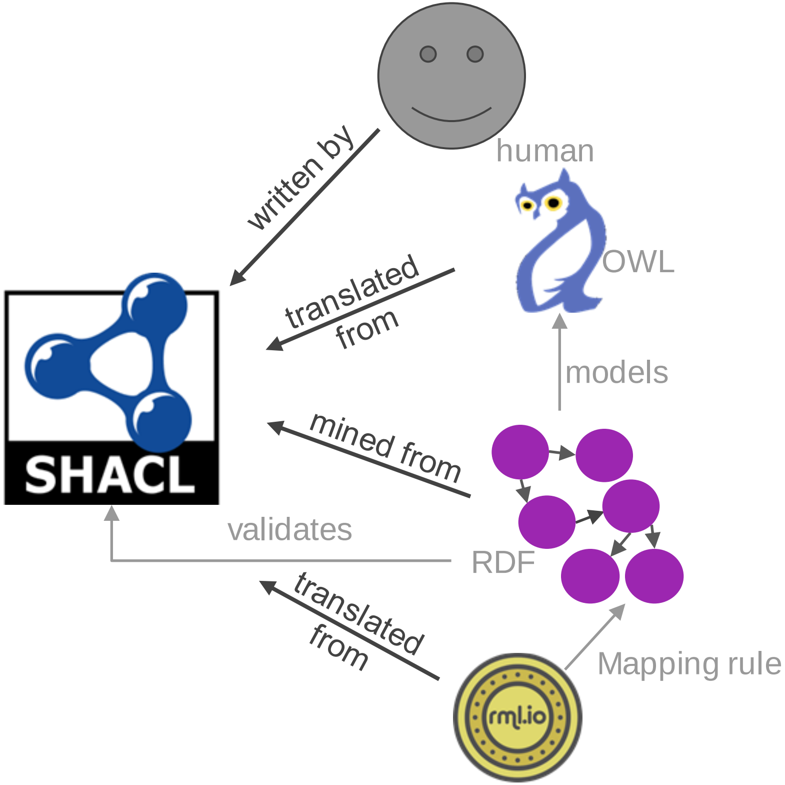
- Validate RDF graphs using created SHACL shapes

RML2SHACL

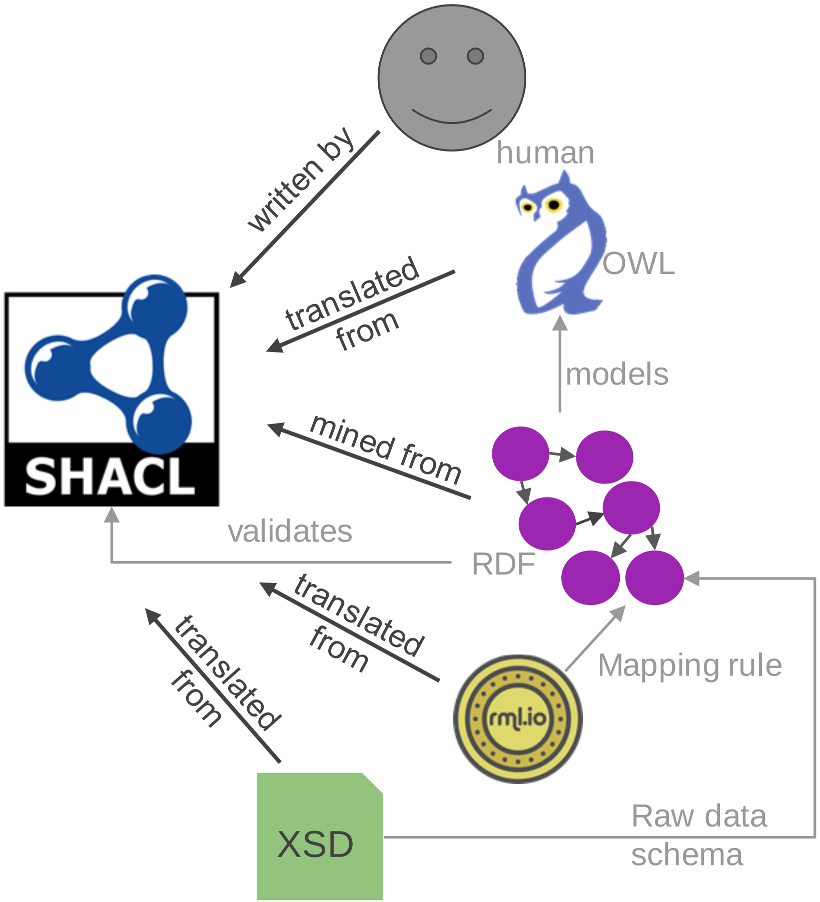
Automatic SHACL shapes extraction



Automatic SHACL shapes extraction

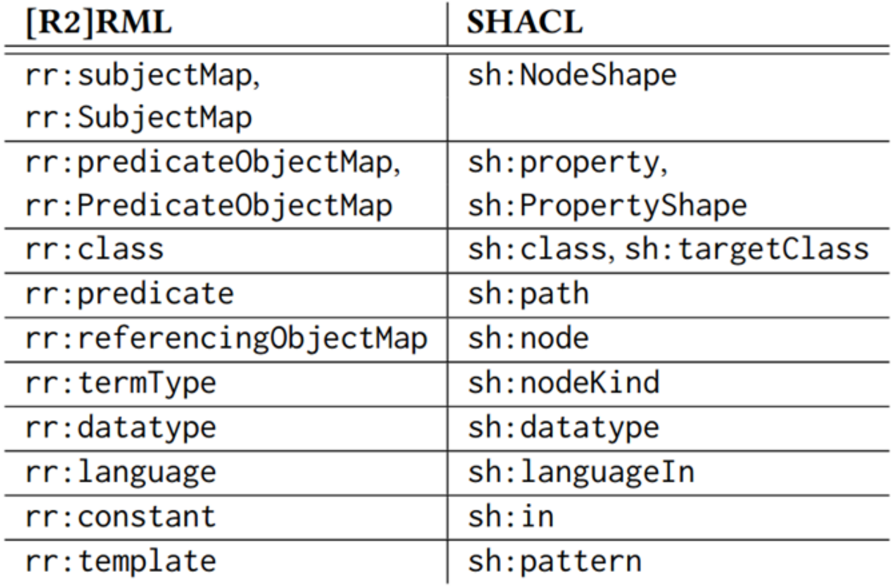


Automatic SHACL shapes extraction



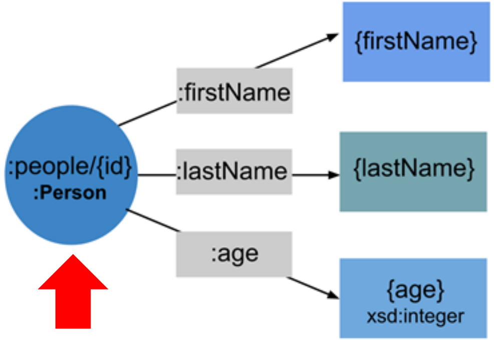
RML2SHACL

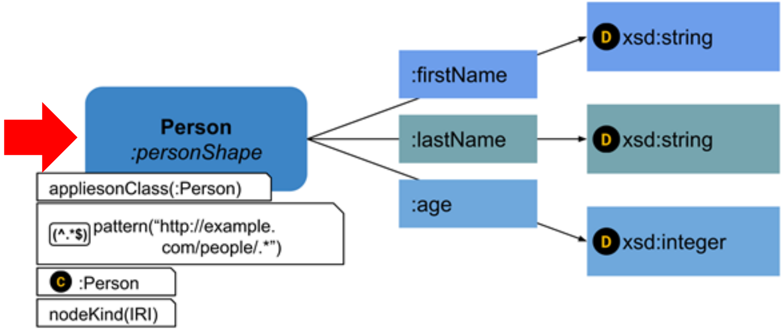
- Translate RML mapping rules to SHACL shapes
- Generate shapes that validate the output of RML mapping rules



[R2]RML	SHACL
rr:subjectMap, rr:SubjectMap	sh:NodeShape
rr:predicateObjectMap, rr:PredicateObjectMap	sh:property, sh:PropertyShape
rr:class	sh:class, sh:targetClass
rr:predicate	sh:path
rr:referencingObjectMap	sh:node
rr:termType	sh:nodeKind
rr:datatype	sh:datatype
rr:language	sh:languageIn
rr:constant	sh:in
rr:template	sh:pattern

Correspondence (1/7)





```
rr:subjectMap,
rr:SubjectMap
```

```
sh:NodeShape
```

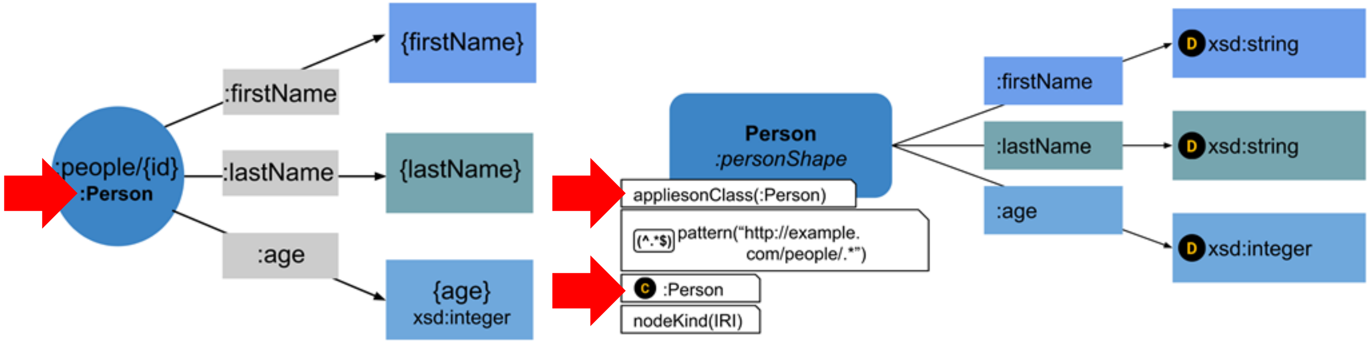
Correspondence (1/7)

```
:StudentMapping a rr:TriplesMap;

rr:subjectMap [
    rr:template "http://example.com/{id}";
    rr:class :Student
    rr:termType rr:IRI ];

  rr:predicateObjectMap [
    rr:predicate :name;
    rr:objectMap [
      rml:reference “name";
      rr:datatype xsd:string ] ].
```

```
:StudentShape a sh:NodeShape .
```

rr:subjectMap, rr:SubjectMap	sh:NodeShape

Correspondence (2/7)

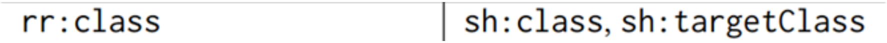


rr:class	sh:class, sh:targetClass

Correspondence (2/7)

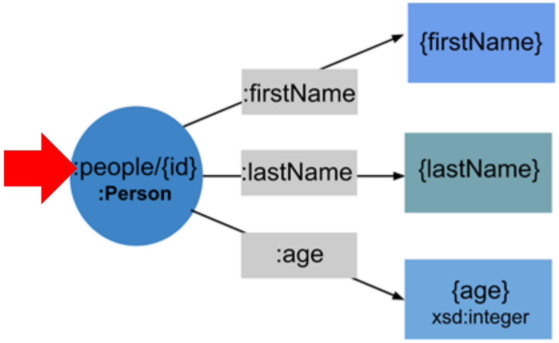
```
:StudentMapping a rr:TriplesMap;

rr:subjectMap [
    rr:template "http://example.com/{id}";
    rr:class :Student ;
    rr:termType rr:IRI ];

  rr:predicateObjectMap [
    rr:predicate :name;
    rr:objectMap [
      rml:reference “name";
      rr:datatype xsd:string ] ].
```

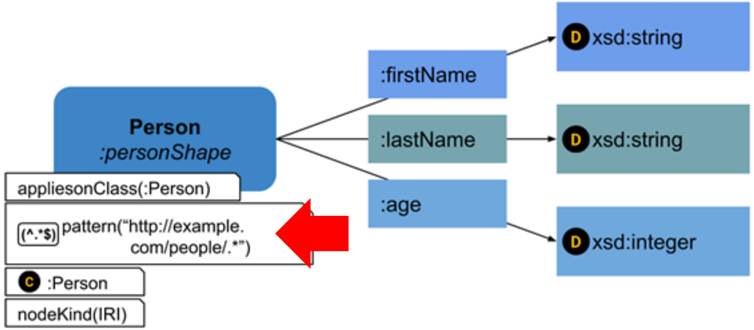
```
:StudentShape a sh:NodeShape ;
  sh:class :Student ;
  sh:targetClass :Student ;
```



rr:class	sh:class, sh:targetClass

Correspondence (3/7)



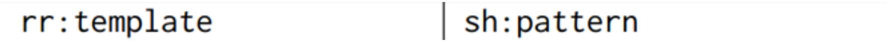


`rr:template` | `sh:pattern`

Correspondence (3/7)

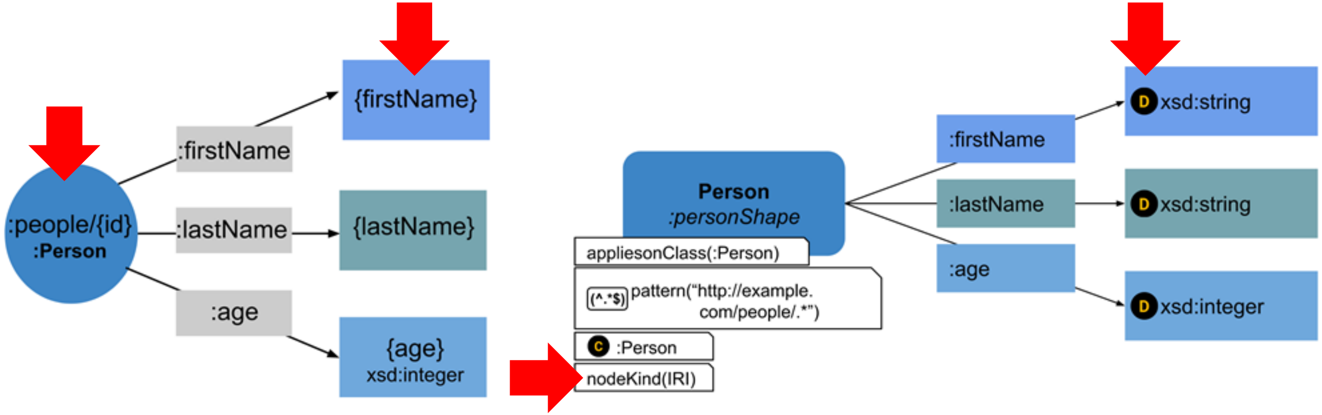
```
:StudentMapping a rr:TriplesMap;

rr:subjectMap [
    rr:template "http://example.com/{id}";
    rr:class :Student ;
    rr:termType rr:IRI ];

  rr:predicateObjectMap [
    rr:predicate :name;
    rr:objectMap [
      rml:reference “name";
      rr:datatype xsd:string ] ].
```

```
:StudentShape a sh:NodeShape ;
  sh:class :Student ;
  sh:targetClass :Student ;
  sh:pattern
"http://example.com/.*" .
```



rr:template	sh:pattern

Correspondence (4/7)



`rr:termType` | `sh:nodeKind`

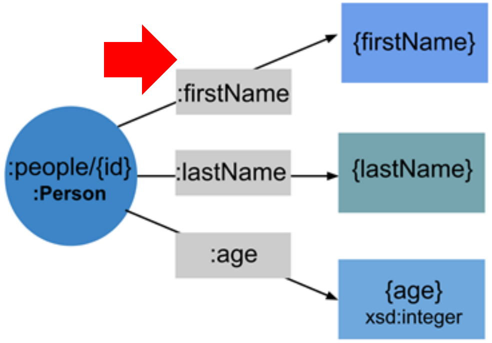
Correspondence (4/7)

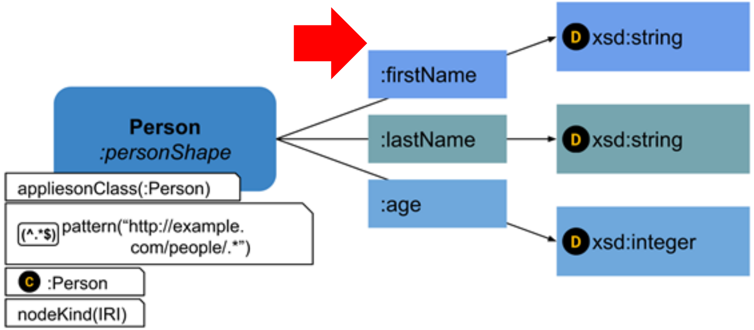
```
:StudentMapping a rr:TriplesMap;

rr:subjectMap [
    rr:template "http://example.com/{id}";
    rr:class :Student ;
    rr:termType rr:IRI];

  rr:predicateObjectMap [
    rr:predicate :name;
    rr:objectMap [
      rml:reference “name";
      rr:datatype xsd:string ] ].
```

```
:StudentShape a sh:NodeShape ;
  sh:class :Student ;
  sh:targetClass :Student ;
  sh:pattern +++ ;
  sh:nodeKind sh:IRI .
```

rr:termType | sh:nodeKind

Correspondence (5/7)



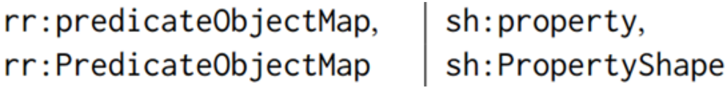


rr:predicateObjectMap,	sh:property,
rr:PredicateObjectMap	sh:PropertyShape

Correspondence (5/7)

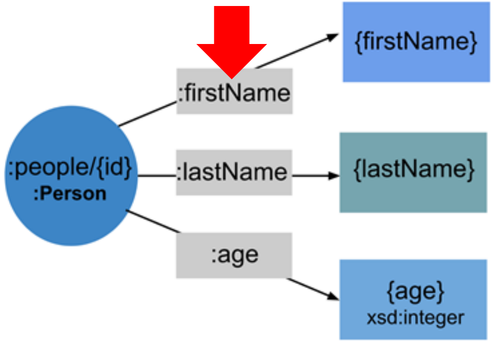
```
:StudentMapping a rr:TriplesMap;

rr:subjectMap [
    rr:template "http://example.com/{id}";
    rr:class :Student
    rr:termType rr:IRI];

  rr:predicateObjectMap [
    rr:predicate :name;
    rr:objectMap [
      rml:reference “name";
      rr:datatype xsd:string ] ].
```

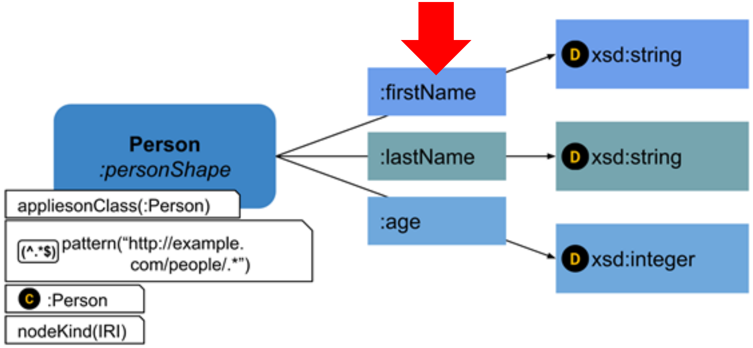
```
:StudentShape a sh:NodeShape ;
  sh:class :Student ;
  sh:targetClass :Student ;
  sh:nodeKind sh:IRI ;
  sh:pattern
"http://example.com/.*" ;
  sh:property :NameShape .

:NameShape a sh:PropertyShape ;
  sh:path :name ;
  sh:datatype xsd:string .
```



rr:predicateObjectMap, rr:PredicateObjectMap	sh:property, sh:PropertyShape

Correspondence (6/7)





rr:predicate | sh:path

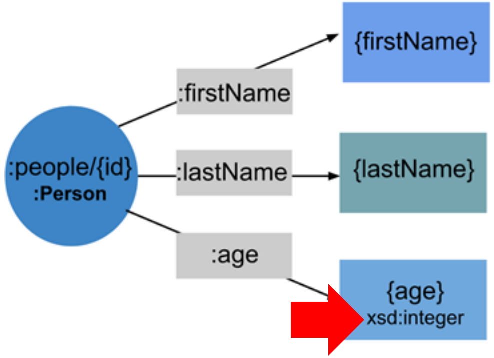
Correspondence (6/7)

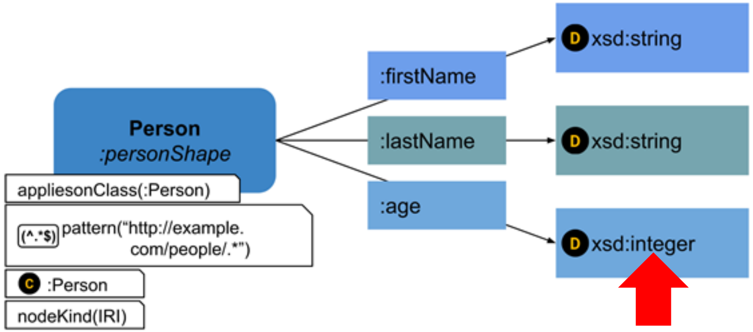
```
:StudentMapping a rr:TriplesMap;

rr:subjectMap [
    rr:template "http://example.com/{id}";
    rr:class :Student ];

  rr:predicateObjectMap [
    rr:predicate :name;
    rr:objectMap [
      rml:reference “name";
      rr:datatype xsd:string ] ].
```

```
:StudentShape a sh:NodeShape ;
  sh:targetClass :Student ;
  sh:nodeKind sh:IRI ;
  sh:pattern
"http://example.com/.*" ;
  sh:property :NameShape .

:NameShape a sh:PropertyShape ;
  sh:path :name ;
  sh:datatype xsd:string .
```

rr:predicate	sh:path

Correspondence (7/7)





rr:datatype | sh:datatype

Correspondence (7/7)

```
:StudentMapping a rr:TriplesMap;

rr:subjectMap [
    rr:template "http://example.com/{id}";
    rr:class :Student ;
    rr:termType rr:IRI];

  rr:predicateObjectMap [
    rr:predicate :name;
    rr:objectMap [
      rml:reference “name";
      rr:datatype xsd:string ] ].
```

```
:StudentShape a sh:NodeShape ;
  sh:class :Student ;
  sh:targetClass :Student ;
  sh:nodeKind sh:IRI ;
  sh:pattern
"http://example.com/.*" ;
  sh:property :NameShape .

:NameShape a sh:PropertyShape ;
  sh:path :name ;
  sh:datatype xsd:string .
```

rr:datatype | sh:datatype

Try RML2SHACL

Try RML2SHACL

SHACL validation.ipynb

https://colab.research.google.com/drive/1n8EAjS7Yq022JF8z067hIA6CCXHIni7T?usp=drive_link

- generate SHACL shapes from RML mapping rules

- validate RDF graphs using RML-driven shapes

Wrapping up...

Conclusions

- Declarative pipeline for KG creation and validation
  - From heterogeneous data sources
  - Relying on standards (or in process to be)
  - Maintenance, reproducibility, understandability
  - Mature ecosystem of compliant systems
- Struggle in adoption (but progressing)
  - Some approaches are too verbose
  - Requires learning curve despite user-friendly developments
  - Manual effort to create mappings and shapes
    - How can LLMs help?

As for K-CAP…

Session 6
Thursday 7th at 14:10

Re-Construction Impact on Metadata Representation Models

Ana Iglesias-Molina, Jhon Toledo, Oscar Corcho and David Chaves-Fraga

Session 6
Thursday 7th at 14:50

XSD2SHACL: Capturing RDF Constraints from XML Schema

Xuemin Duan, David Chaves-Fraga and Anastasia Dimou

Declarative Construction and Validation of Knowledge Graphs

Ana Iglesias-Molina and Xuemin Duan

Thanks for attending!