SHACL

W3C's Shapes Constraint Language

Bernd Neumayr

Johannes Kepler University Linz
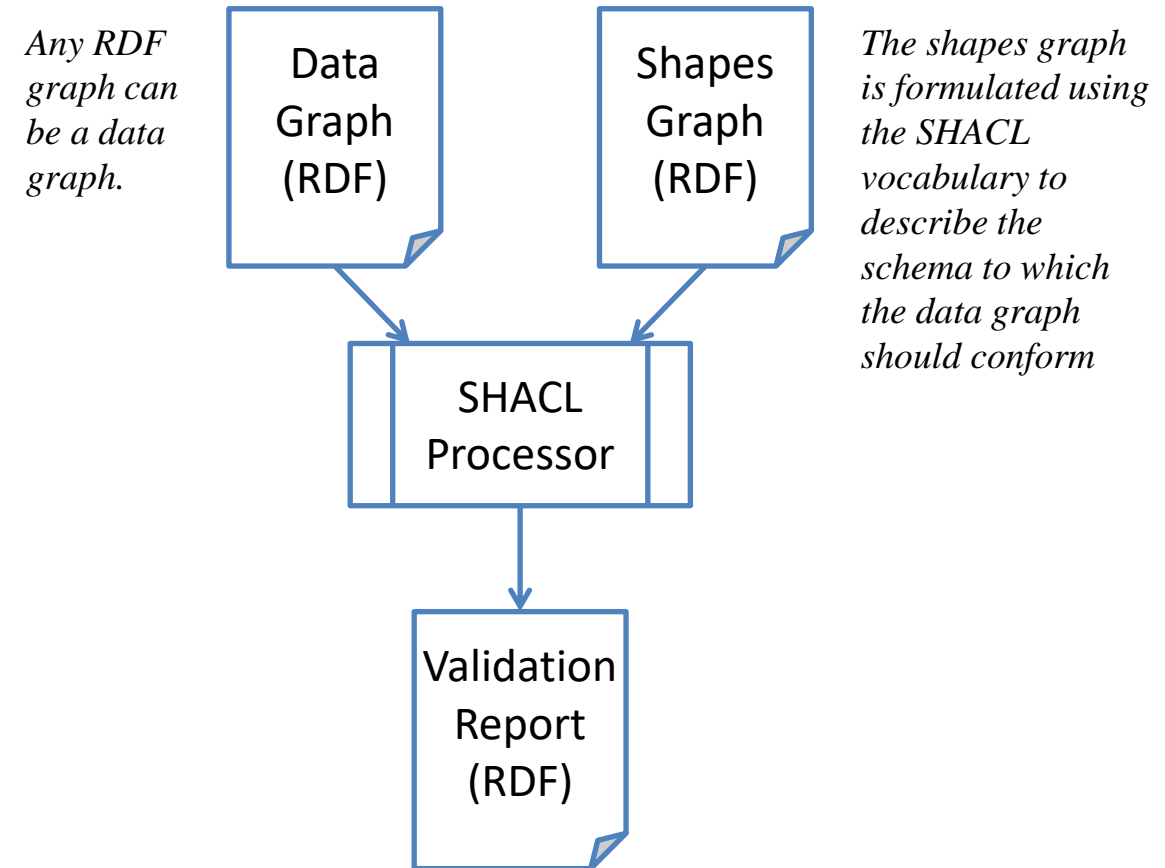
based on

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

How to use these slides?

- These slides are a first introduction to SHACL
- The goal is to thoroughly understand the basics of SHACL and not to cover the whole language
  - try to fully understand everything in this slides! *(that will be enough for the exam)*
- For full coverage of the language please consult the W3C recommendation at https://www.w3.org/TR/shacl/
- With these slides as introduction it should be rather easy to understand the W3C recommendation.

Why SHACL?

- With RDF Schema (RDFS) one can define vocabularies (sets of classes and properties) to be used in RDF data

- With OWL one can define the exact meaning of classes and properties in such vocabularies.

- What was missing in the semantic web stack was a way to specify structural constraints over RDF data similar to a database schema.

- Using RDFS we can specify a vocabulary consisting of classes Man, Woman, Person, hasFather, and hasMother and hasParent. We can specify:
  - every woman/man is also a person
  - a hasParent statement relates a person to a person
  - a hasMother/hasFather statement also expresses a hasParent statement and relates a person to a woman/man

- In OWL one can specify, using this vocabulary, that every person has exactly one mother and has exactly one father (without making a statement about whether these relationships are also recorded in the Knowledge Graph)

- In SHACL one can specify that every **description of a person** in some knowledge graph must come with exactly one statement with predicate hasMother and exactly one statement with hasFather.

SHACL

- The SHACL Shapes Constraint Language (SHACL) is a language for validating RDF graphs against a set of conditions.

- These conditions are provided as shapes and other constructs expressed in the form of an RDF graph.

- RDF graphs that are used in this manner are called "shapes graphs" in SHACL and the RDF graphs that are validated against a shapes graph are called "data graphs".

- As SHACL shape graphs are used to validate that data graphs satisfy a set of conditions they can also be viewed as a description of the data graphs that do satisfy these conditions.

- Such descriptions may be used for a variety of purposes beside validation, including user interface building, code generation and data integration.

Any RDF graph can be a data graph.

Data Graph (RDF)

Shapes Graph (RDF)

The shapes graph is formulated using the SHACL vocabulary to describe the schema to which the data graph should conform

SHACL Processor
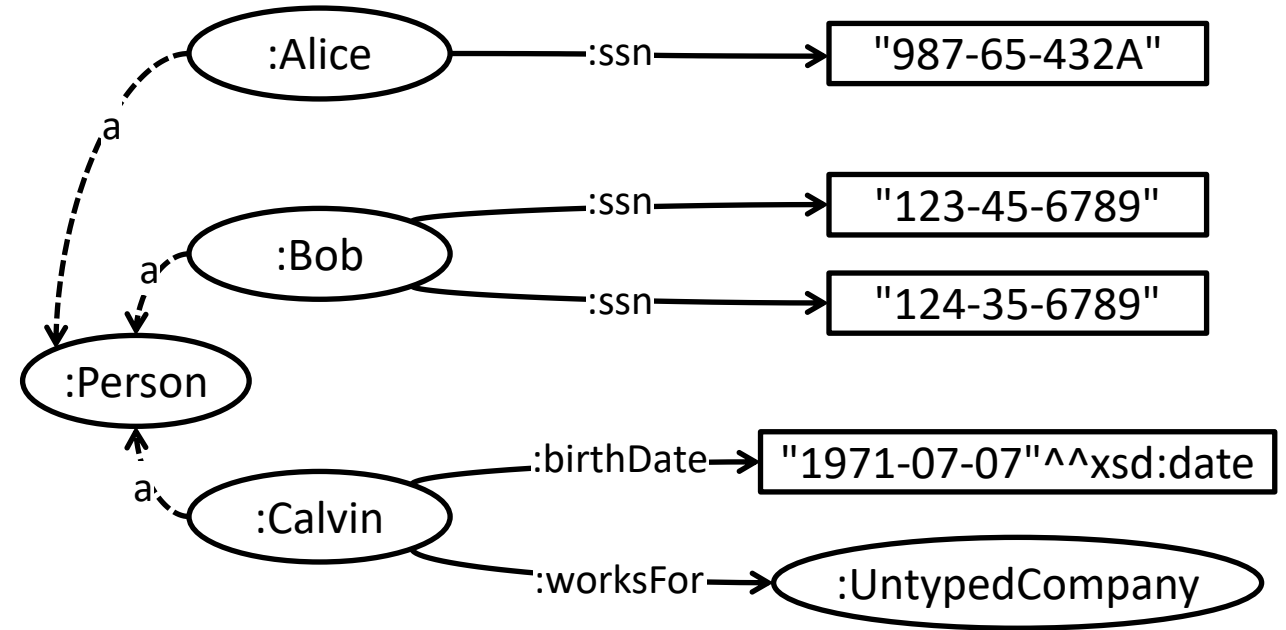
Validation Report (RDF)

The validation report is the result of the validation process that reports the conformance and the set of all validation results. The validation report is described with the SHACL Validation Report Vocabulary. It provides guidance on how to identify or fix violations in the data graph.

SHACL - First Example

Data Graph

```
:Alice   a :Person ;
         :ssn "987-65-432A" .
:Bob     a :Person ;
         :ssn "123-45-6789" ;
         :ssn "124-35-6789" .
:Calvin a :Person ;
         :birthDate "1971-07-07"^^xsd:date ;
         :worksFor :UntypedCompany .
```
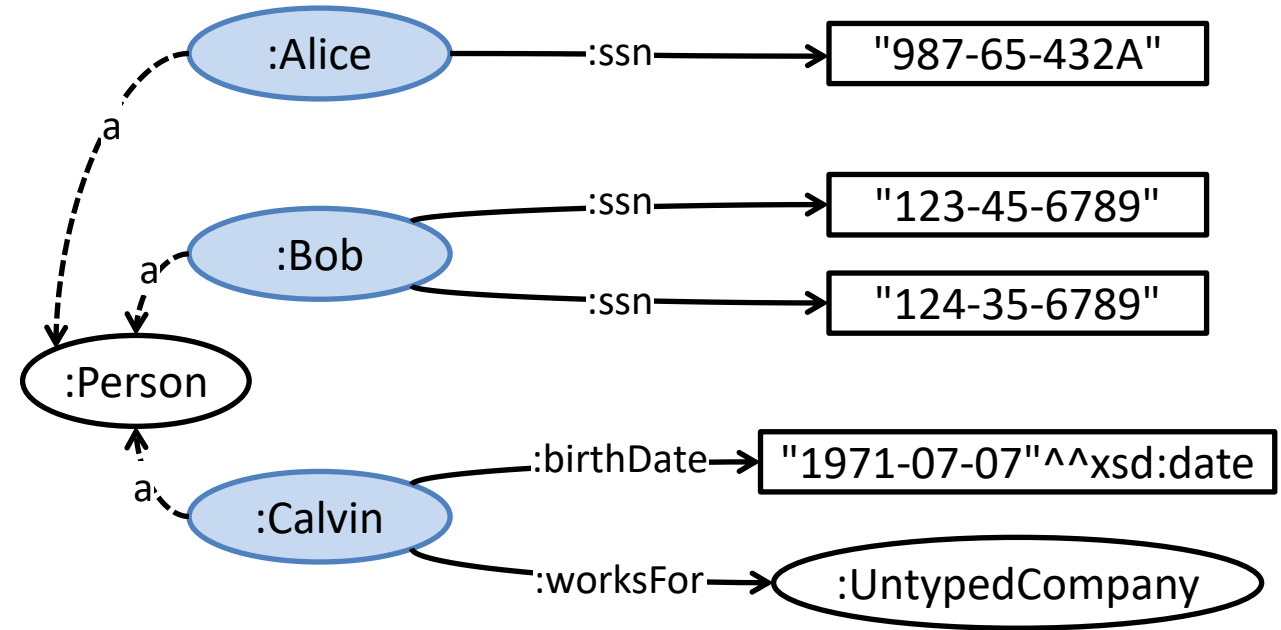
Data Graph

```
:Alice    a :Person ;
          :ssn "987-65-432A" .
:Bob      a :Person ;
          :ssn "123-45-6789" ;
          :ssn "124-35-6789" .
:Calvin a :Person ;
          :birthDate "1971-07-07"^^xsd:date ;
          :worksFor :UntypedCompany .
```

Let us define some structural constraints for descriptions of persons ...
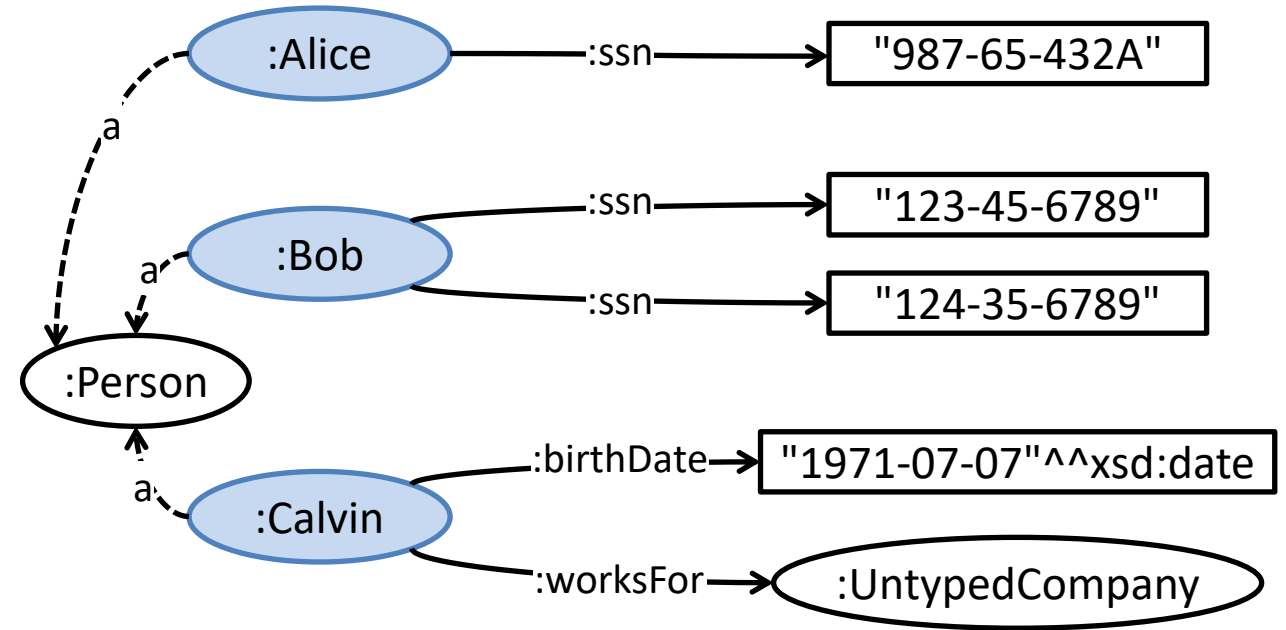
Data Graph

```
:Alice    a :Person ;
          :ssn "987-65-432A" .
:Bob      a :Person ;
          :ssn "123-45-6789" ;
          :ssn "124-35-6789" .
:Calvin a :Person ;
          :birthDate "1971-07-07"^^xsd:date ;
          :worksFor :UntypedCompany .
```

Shapes Graph

```
:PersonShape  a sh:NodeShape;
    sh:targetClass :Person ;
    ...  .
```

A SHACL instance of `:Person`
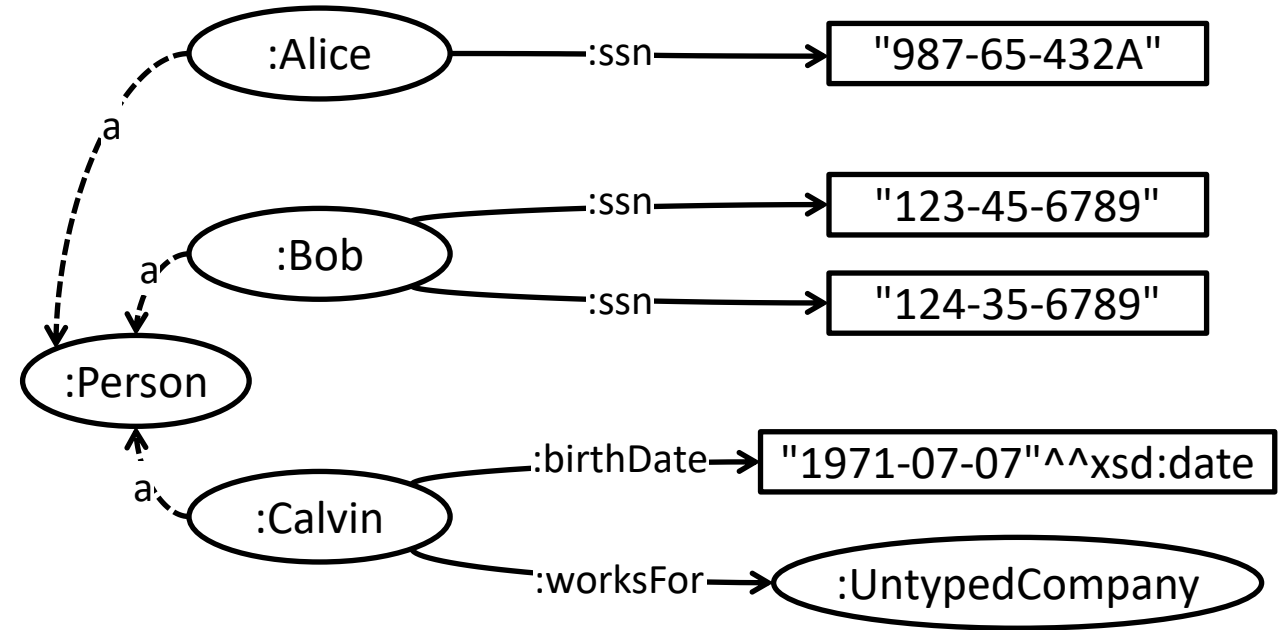* ...

SHACL - First Example

Data Graph

```
:Alice    a :Person ;
          :ssn "987-65-432A" .
:Bob      a :Person ;
          :ssn "123-45-6789" ;
          :ssn "124-35-6789" .
:Calvin a :Person ;
          :birthDate "1971-07-07"^^xsd:date ;
          :worksFor :UntypedCompany .
```

Shapes Graph

```
:PersonShape  a sh:NodeShape;
   sh:targetClass :Person ;
   sh:property :SsnShape .

:SsnShape   a sh:PropertyShape ;
   sh:path :ssn ;
   sh:maxCount 1 .
```


A SHACL instance of `:Person`
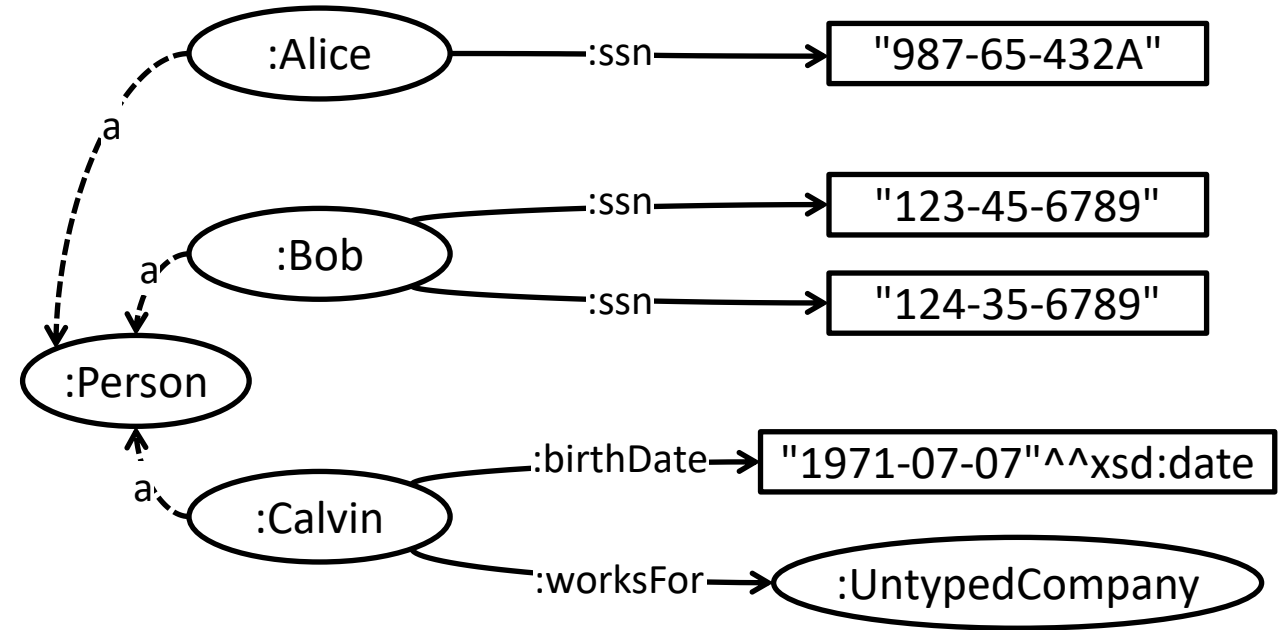- can have at most one value for the property `:ssn`.

SHACL - First Example

Data Graph

```
:Alice    a :Person ;
          :ssn "987-65-432A" .
:Bob      a :Person ;
          :ssn "123-45-6789" ;
          :ssn "124-35-6789" .
:Calvin a :Person ;
          :birthDate "1971-07-07"^^xsd:date ;
          :worksFor :UntypedCompany .
```

Shapes Graph

```
:PersonShape  a sh:NodeShape;
   sh:targetClass :Person ;
   sh:property :SsnShape .

:SsnShape  a sh:PropertyShape ;
   sh:path :ssn ;
   sh:maxCount 1 ;
   sh:datatype xsd:string ;
   sh:pattern "^\\d{3}-\\d{2}-\\d{4}$" .
```


A SHACL instance of :Person
- can have at most one value for the property :ssn, and this value is a literal with the datatype xsd:string that matches a specified regular expression.

SHACL - First Example

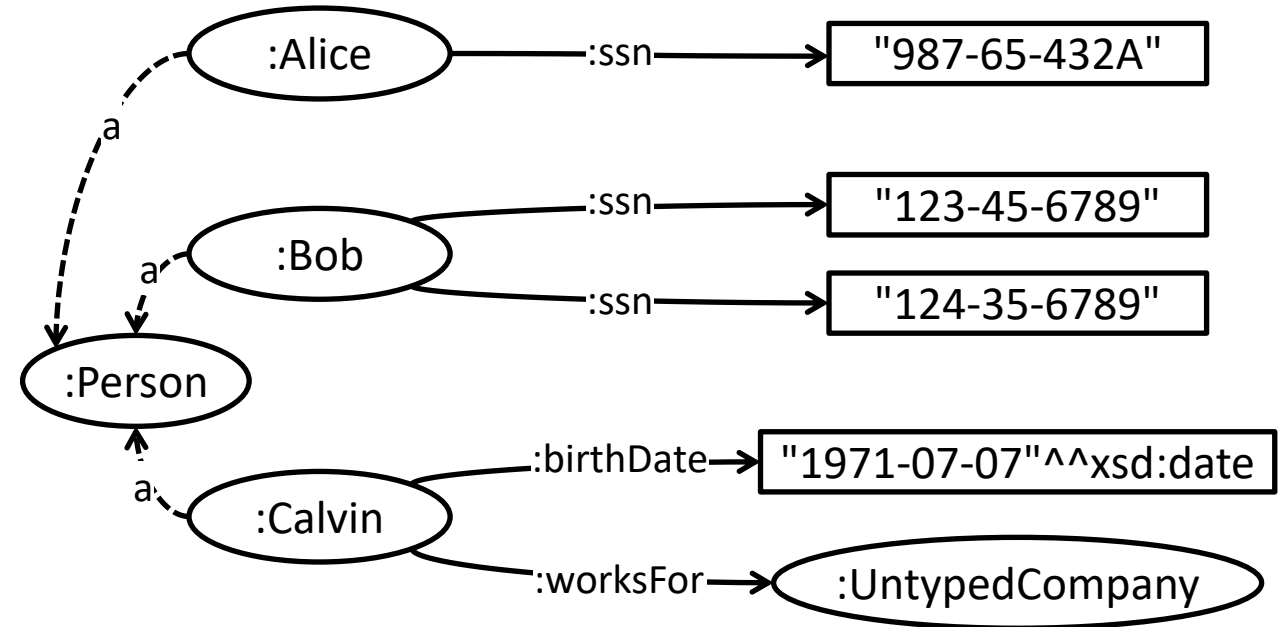## Data Graph

```
:Alice    a :Person ;
          :ssn "987-65-432A" .
:Bob      a :Person ;
          :ssn "123-45-6789" ;
          :ssn "124-35-6789" .
:Calvin a :Person ;
          :birthDate "1971-07-07"^^xsd:date ;
          :worksFor :UntypedCompany .
```

Shapes Graph

```
:PersonShape  a sh:NodeShape;
    sh:targetClass :Person ;
    sh:property :SsnShape, :WorksForShape .

:SsnShape  a sh:PropertyShape ;
    sh:path :ssn ;
    sh:maxCount 1 ;
    sh:datatype xsd:string ;
    sh:pattern "^\\d{3}-\\d{2}-\\d{4}$" .

:WorksForShape  a sh:PropertyShape ;
    sh:path :worksFor ;
    sh:class :Company ;
    sh:nodeKind sh:IRI .
```


A SHACL instance of :Person
- can have at most one value for the property :ssn, and this value is a literal with the datatype xsd:string that matches a specified regular expression.
- can have unlimited values for the property :worksFor, and these values are IRIs and SHACL instances of :Company.

SHACL - First Example
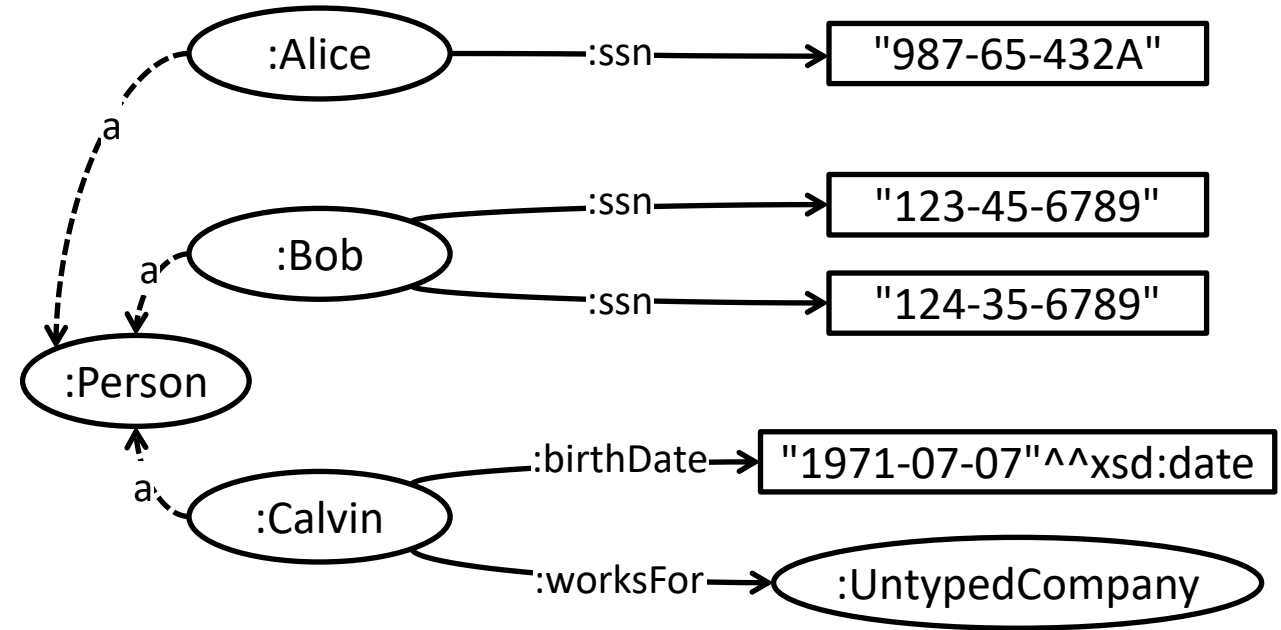
Data Graph

```
:Alice   a :Person ;
         :ssn "987-65-432A" .
:Bob     a :Person ;
         :ssn "123-45-6789" ;
         :ssn "124-35-6789" .
:Calvin a :Person ;
         :birthDate "1971-07-07"^^xsd:date ;
         :worksFor :UntypedCompany .
```

Shapes Graph

```
:PersonShape  a sh:NodeShape;
    sh:targetClass :Person ;
    sh:property :SsnShape, :WorksForShape ;
    sh:closed true ;
    sh:ignoredProperties ( rdf:type ) .

:SsnShape  a sh:PropertyShape ;
    sh:path :ssn ;
    sh:maxCount 1 ;
    sh:datatype xsd:string ;
    sh:pattern "^\\d{3}-\\d{2}-\\d{4}$" .

:WorksForShape  a sh:PropertyShape ;
    sh:path :worksFor ;
    sh:class :Company ;
    sh:nodeKind sh:IRI .
```

A SHACL instance of `:Person`
- can have at most one value for the property `:ssn`, and this value is a literal with the datatype `xsd:string` that matches a specified regular expression.
- can have unlimited values for the property `:worksFor`, and these values are IRIs and SHACL instances of `:Company`.
- cannot have values for any other property apart from `:ssn`, `:worksFor` and `rdf:type`.
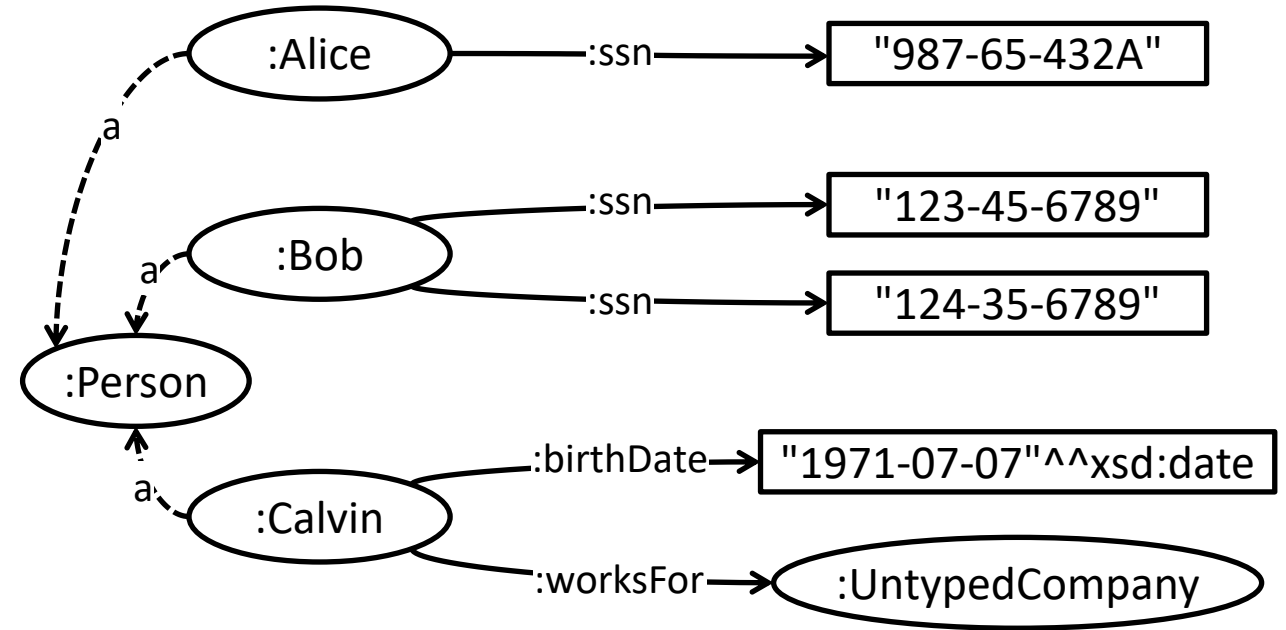
Data Graph

```
:Alice    a :Person ;
          :ssn "987-65-432A" .
:Bob      a :Person ;
          :ssn "123-45-6789" ;
          :ssn "124-35-6789" .
:Calvin a :Person ;
          :birthDate "1971-07-07"^^xsd:date ;
          :worksFor :UntypedCompany .
```

Shapes Graph

```
:PersonShape  a sh:NodeShape;
    sh:targetClass :Person ;
    sh:property :SsnShape, :WorksForShape;
    sh:closed true ;
    sh:ignoredProperties ( rdf:type ) .

:SsnShape  a sh:PropertyShape ;
    sh:path :ssn ;
    sh:maxCount 1 ;
    sh:datatype xsd:string ;
    sh:pattern "^\\d{3}-\\d{2}-\\d{4}$" .

:WorksForShape  a sh:PropertyShape ;
    sh:path :worksFor ;
    sh:class :Company ;
    sh:nodeKind sh:IRI .
```


Let the SHACL processor validate the data graph against the shapes graph ...

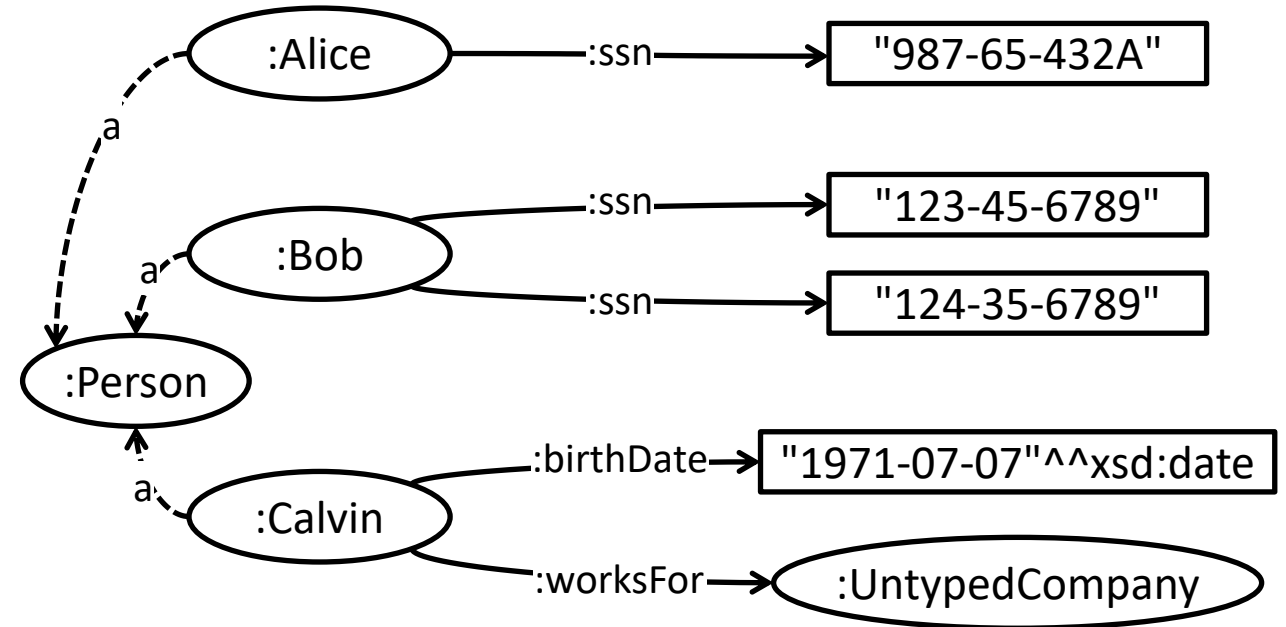## Data Graph

```
:Alice    a :Person ;
          :ssn "987-65-432A" .
:Bob      a :Person ;
          :ssn "123-45-6789" ;
          :ssn "124-35-6789" .
:Calvin a :Person ;
          :birthDate "1971-07-07"^^xsd:date ;
          :worksFor :UntypedCompany .
```

Shapes Graph

```
:PersonShape  a sh:NodeShape;
    sh:targetClass :Person ;
    sh:property :SsnShape, :WorksForShape;
    sh:closed true ;
    sh:ignoredProperties ( rdf:type ) .

:SsnShape  a sh:PropertyShape ;
    sh:path :ssn ;
    sh:maxCount 1 ;
    sh:datatype xsd:string ;
    sh:pattern "^\\d{3}-\\d{2}-\\d{4}$" .

:WorksForShape  a sh:PropertyShape ;
    sh:path :worksFor ;
    sh:class :Company ;
    sh:nodeKind sh:IRI .
```

Validation Report

```
[
 a sh:ValidationReport ;
 sh:conforms false ;
 ...
```

SHACL - First Example

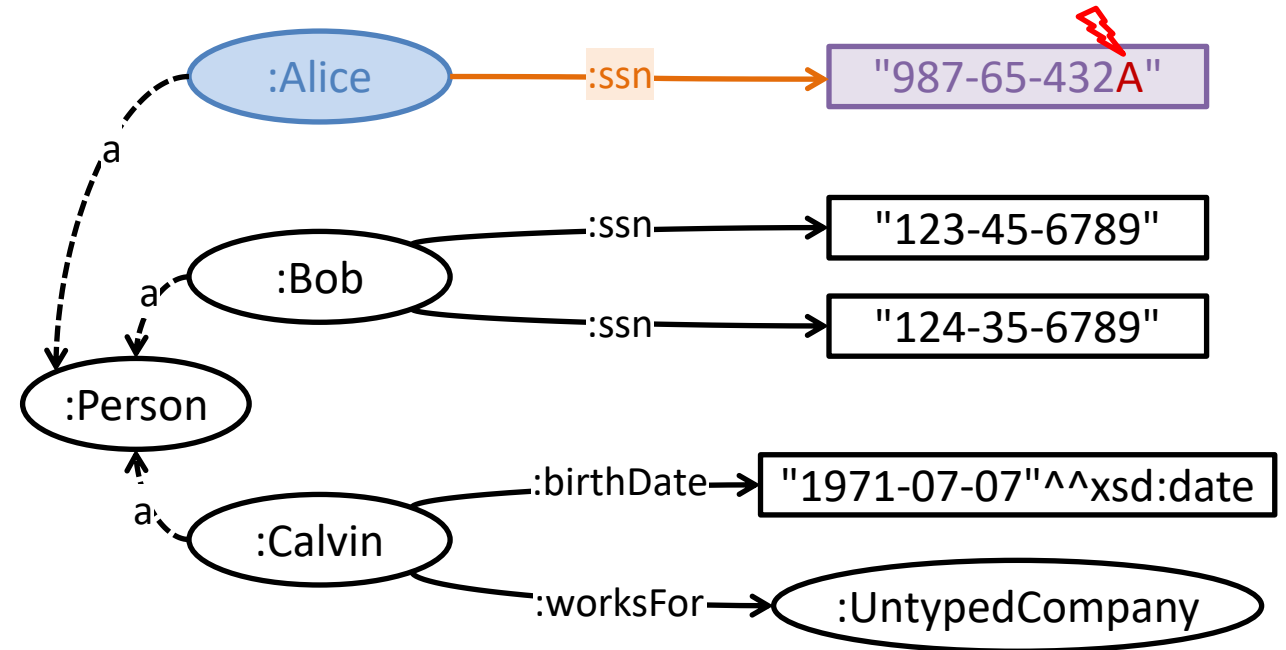## Data Graph

```
:Alice    a :Person ;
          :ssn "987-65-432A" .
:Bob      a :Person ;
          :ssn "123-45-6789" ;
          :ssn "124-35-6789" .
:Calvin a :Person ;
          :birthDate "1971-07-07"^^xsd:date ;
          :worksFor :UntypedCompany .
```

Shapes Graph

```
:PersonShape  a sh:NodeShape;
    sh:targetClass :Person ;
    sh:property :SsnShape, :WorksForShape;
    sh:closed true ;
    sh:ignoredProperties ( rdf:type ) .

:SsnShape   a sh:PropertyShape ;
    sh:path :ssn ;
    sh:maxCount 1 ;
    sh:datatype xsd:string ;
    sh:pattern "^\\d{3}-\\d{2}-\\d{4}$" .

:WorksForShape  a sh:PropertyShape ;
    sh:path :worksFor ;
    sh:class :Company ;
    sh:nodeKind sh:IRI .
```


Validation Report

```
...
sh:result
[ a sh:ValidationResult ;
  sh:resultSeverity sh:Violation ;
  sh:focusNode :Alice ;
  sh:resultPath :ssn ;
  sh:value "987-65-432A" ;
  sh:sourceConstraintComponent sh:RegexConstraintComp...;
  sh:sourceShape :SsnShape ] ;
...
```

SHACL - First Example
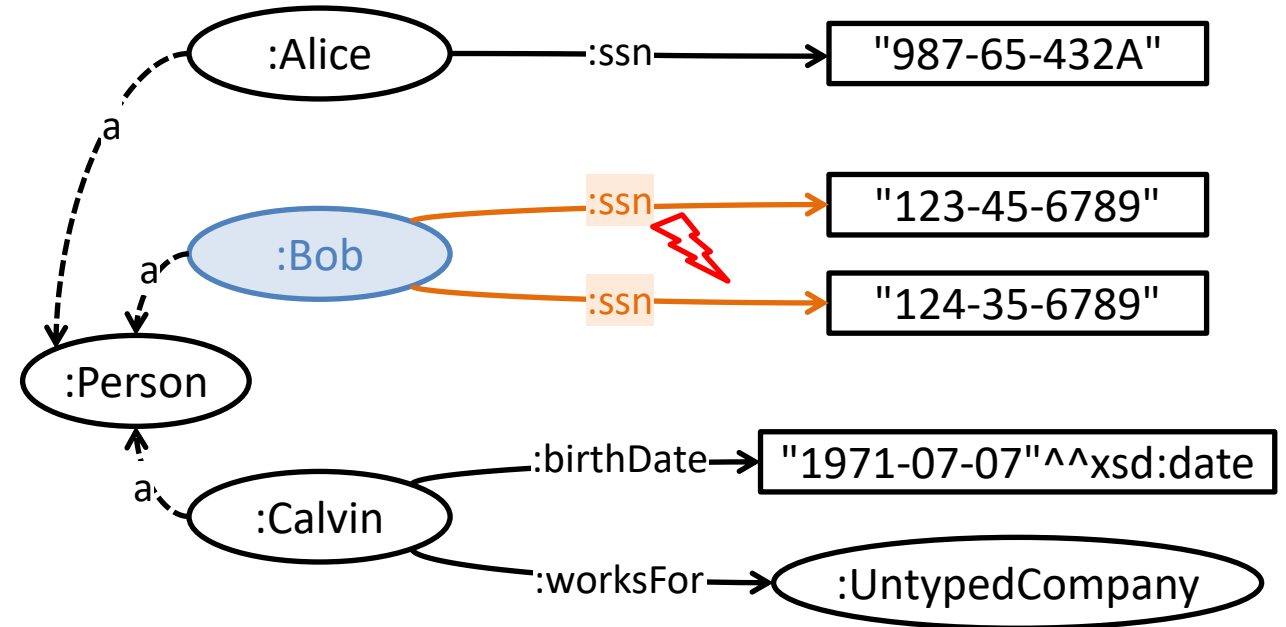
Data Graph

```
:Alice   a :Person ;
         :ssn "987-65-432A" .
:Bob     a :Person ;
         :ssn "123-45-6789" ;
         :ssn "124-35-6789" .
:Calvin a :Person ;
         :birthDate "1971-07-07"^^xsd:date ;
         :worksFor :UntypedCompany .
```

Shapes Graph

```
:PersonShape  a sh:NodeShape;
    sh:targetClass :Person ;
    sh:property :SsnShape, :WorksForShape;
    sh:closed true ;
    sh:ignoredProperties ( rdf:type ) .

:SsnShape  a sh:PropertyShape ;
    sh:path :ssn ;
    sh:maxCount 1 ;
    sh:datatype xsd:string ;
    sh:pattern "^\\d{3}-\\d{2}-\\d{4}$" .

:WorksForShape  a sh:PropertyShape ;
    sh:path :worksFor ;
    sh:class :Company ;
    sh:nodeKind sh:IRI .
```


Validation Report

```
...
sh:result
[ a sh:ValidationResult ;
  sh:resultSeverity sh:Violation ;
  sh:focusNode :Bob ;
  sh:resultPath :ssn ;
  sh:sourceConstraintComponent sh:MaxCountConstraintC...;
  sh:sourceShape :SsnShape ] ;
...
```

SHACL - First Example
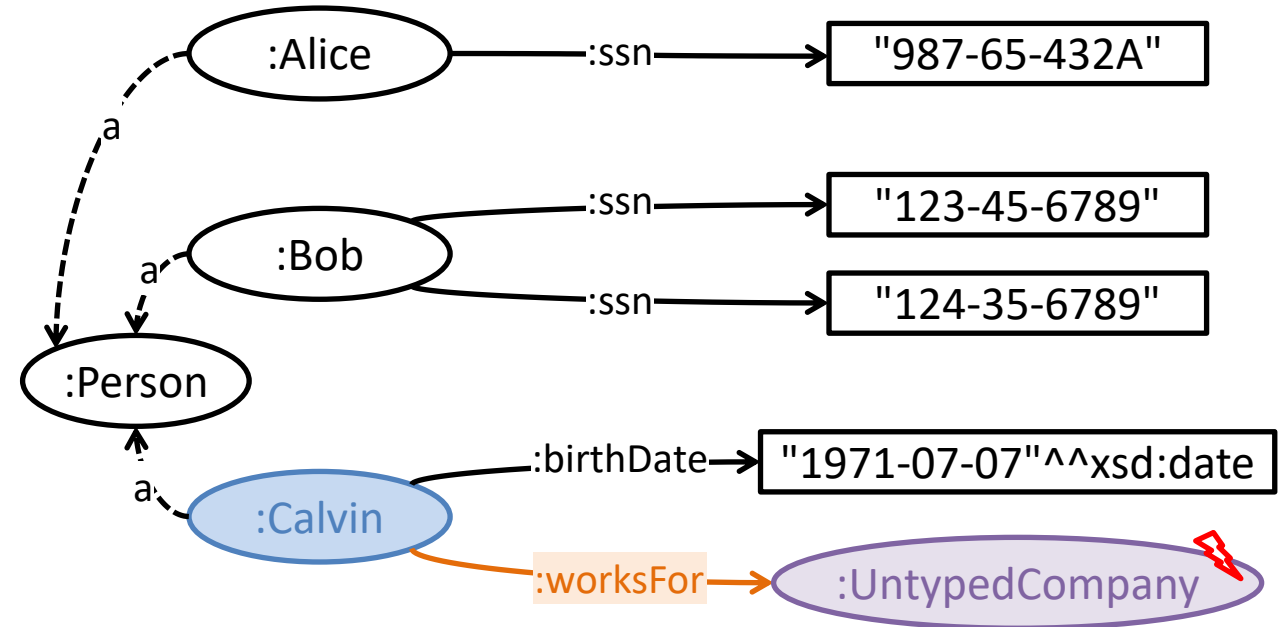
Data Graph

```
:Alice   a :Person ;
         :ssn "987-65-432A" .
:Bob     a :Person ;
         :ssn "123-45-6789" ;
         :ssn "124-35-6789" .
:Calvin a :Person ;
         :birthDate "1971-07-07"^^xsd:date ;
         :worksFor :UntypedCompany .
```

Shapes Graph

```
:PersonShape  a sh:NodeShape;
    sh:targetClass :Person ;
    sh:property :SsnShape, :WorksForShape;
    sh:closed true ;
    sh:ignoredProperties ( rdf:type ) .

:SsnShape  a sh:PropertyShape ;
    sh:path :ssn ;
    sh:maxCount 1 ;
    sh:datatype xsd:string ;
    sh:pattern "^\\d{3}-\\d{2}-\\d{4}$" .

:WorksForShape  a sh:PropertyShape ;
    sh:path :worksFor ;
    sh:class :Company ;
    sh:nodeKind sh:IRI .
```


Validation Report

```
...
sh:result
[ a sh:ValidationResult ;
  sh:resultSeverity sh:Violation ;
  sh:focusNode :Calvin ;
  sh:resultPath :worksFor ;
  sh:value :UntypedCompany ;
  sh:sourceConstraintComponent sh:ClassConstraintComp...;
  sh:sourceShape :WorksForShape ] ;
...
```

SHACL - First Example
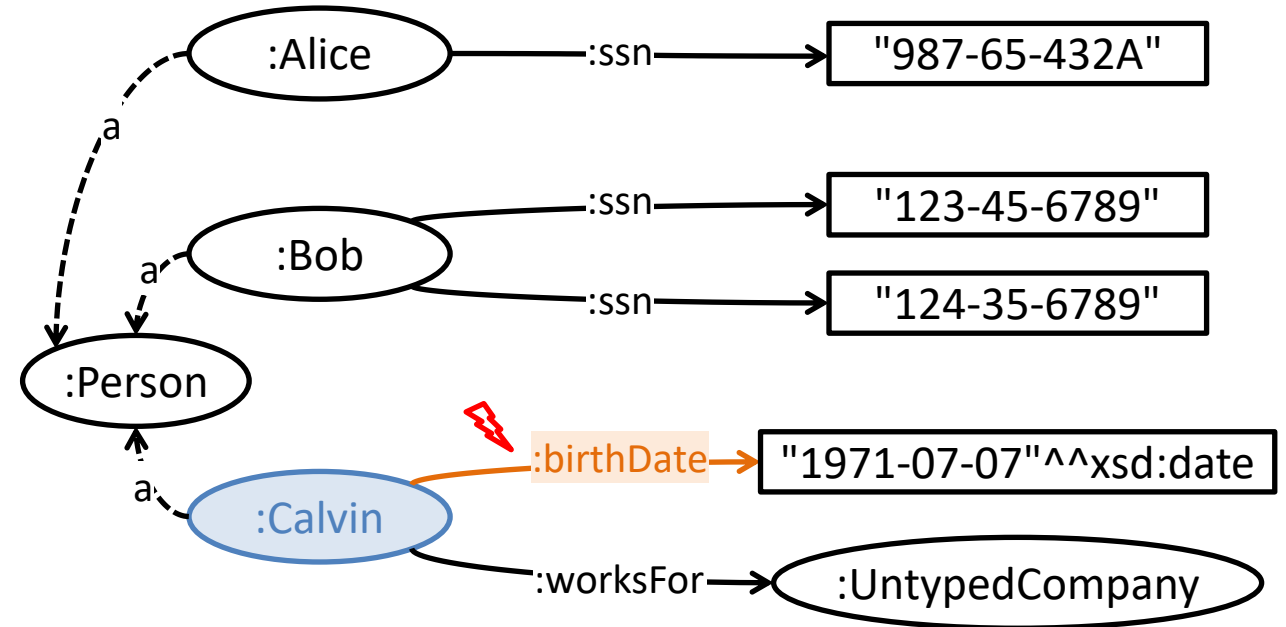
Data Graph

```
:Alice   a :Person ;
         :ssn "987-65-432A" .
:Bob     a :Person ;
         :ssn "123-45-6789" ;
         :ssn "124-35-6789" .
:Calvin a :Person ;
         :birthDate "1971-07-07"^^xsd:date ;
         :worksFor :UntypedCompany .
```

Shapes Graph

```
:PersonShape  a sh:NodeShape;
    sh:targetClass :Person ;
    sh:property :SsnShape, :WorksForShape;
    sh:closed true ;
    sh:ignoredProperties ( rdf:type ) .

:SsnShape  a sh:PropertyShape ;
    sh:path :ssn ;
    sh:maxCount 1 ;
    sh:datatype xsd:string ;
    sh:pattern "^\\d{3}-\\d{2}-\\d{4}$" .

:WorksForShape  a sh:PropertyShape ;
    sh:path :worksFor ;
    sh:class :Company ;
    sh:nodeKind sh:IRI .
```

Validation Report

```
...
sh:result
[ a sh:ValidationResult ;
  sh:resultSeverity sh:Violation ;
  sh:focusNode ex:Calvin ;
  sh:resultPath ex:birthDate ;
  sh:value "1971-07-07"^^xsd:date ;
  sh:sourceConstraintComponent sh:ClosedConstraintComp...;
  sh:sourceShape :PersonShape ]
].
```

Visualization of Shapes Graphs (no standard)

Data Graph

```
:Alice    a :Person ;
          :ssn "987-65-432A" .
:Bob      a :Person ;
          :ssn "123-45-6789" ;
          :ssn "124-35-6789" .
:Calvin a :Person ;
          :birthDate "1971-07-07"^^xsd:date ;
          :worksFor :UntypedCompany .
```
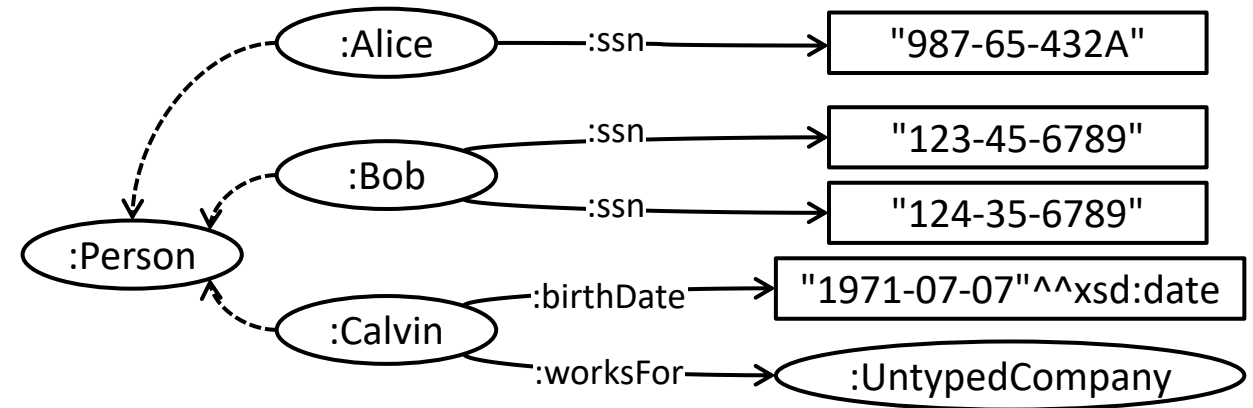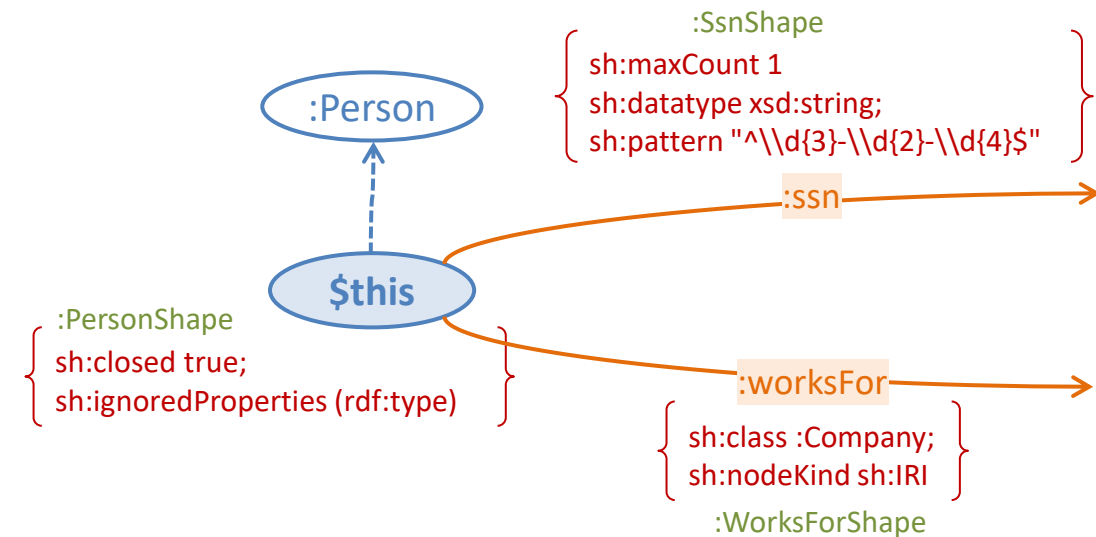

Shapes Graph

```
:PersonShape  a sh:NodeShape;
    sh:targetClass :Person ;
    sh:property :SsnShape, :WorksForShape;
    sh:closed true ;
    sh:ignoredProperties ( rdf:type ) .

:SsnShape  a sh:PropertyShape ;
    sh:path :ssn ;
    sh:maxCount 1 ;
    sh:datatype xsd:string ;
    sh:pattern "^\\d{3}-\\d{2}-\\d{4}$" .

:WorksForShape  a sh:PropertyShape ;
    sh:path :worksFor ;
    sh:class :Company ;
    sh:nodeKind sh:IRI .
```

Node Shape

- ≡ a shape that is not the subject of a sh:path statement.
- Node shapes specify constraints that need to be met with respect to focus nodes. In contrast to property shapes they primarily apply to the focus node itself, not to its property values.

Shapes Graph

```
:PersonShape  a sh:NodeShape;
    sh:targetClass :Person ;
    sh:property :SsnShape, :WorksForShape;
    sh:closed true ;
    sh:ignoredProperties ( rdf:type ) .

:SsnShape  a sh:PropertyShape ;
    sh:path :ssn ;
    sh:maxCount 1 ;
    sh:datatype xsd:string ;
    sh:pattern "^\\d{3}-\\d{2}-\\d{4}$" .

:WorksForShape  a sh:PropertyShape ;
    sh:path :worksFor ;
    sh:class :Company ;
    sh:nodeKind sh:IRI .
```

Target Declaration

- can be understood as a query returning the set of focus nodes for a shape
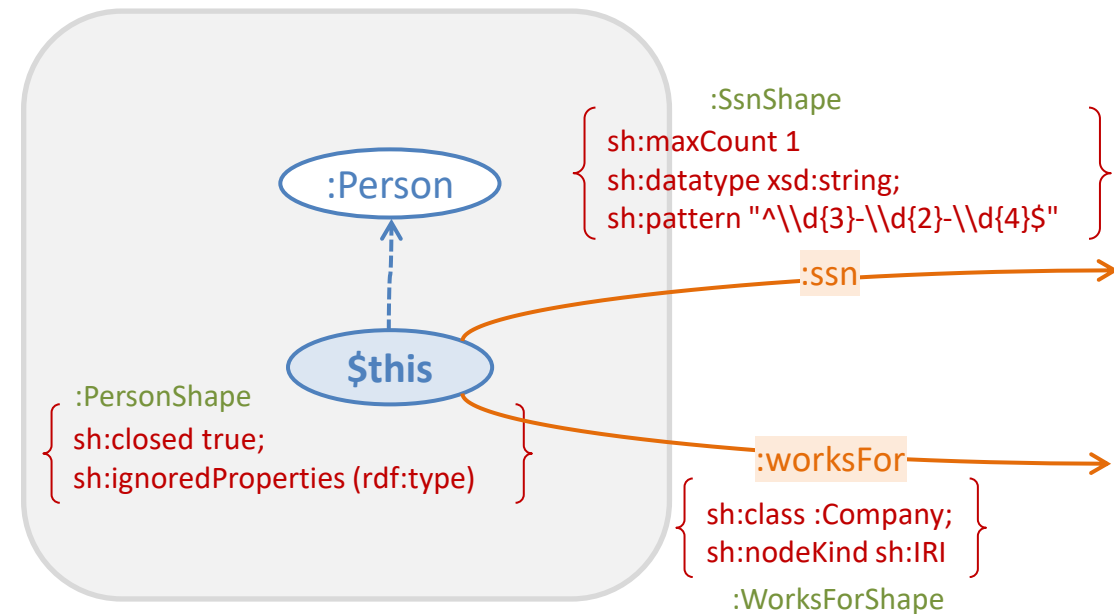
Shapes Graph

```
:PersonShape  a sh:NodeShape;
    sh:targetClass :Person ;
    sh:property :SsnShape, :WorksForShape;
    sh:closed true ;
    sh:ignoredProperties ( rdf:type ) .

:SsnShape  a sh:PropertyShape ;
    sh:path :ssn ;
    sh:maxCount 1 ;
    sh:datatype xsd:string ;
    sh:pattern "^\\d{3}-\\d{2}-\\d{4}$" .

:WorksForShape  a sh:PropertyShape ;
    sh:path :worksFor ;
    sh:class :Company ;
    sh:nodeKind sh:IRI .
```
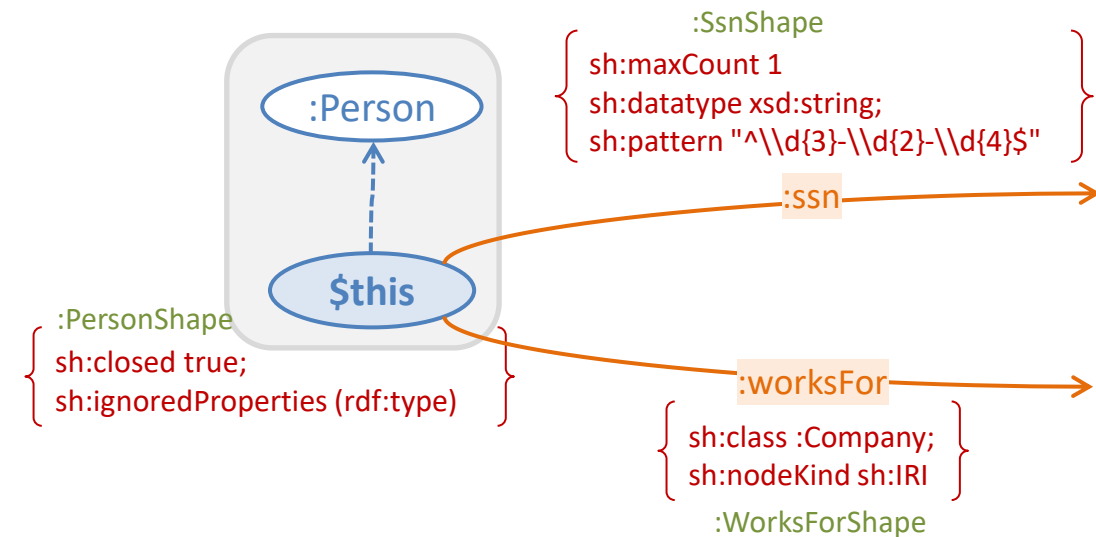
:SsnShape
sh:maxCount 1
sh:datatype xsd:string;
sh:pattern "^\\d{3}-\\d{2}-\\d{4}$"

:ssn

:Person

$this

:PersonShape
sh:closed true;
sh:ignoredProperties (rdf:type)

:worksFor

sh:class :Company;
sh:nodeKind sh:IRI

:WorksForShape

Property Shape

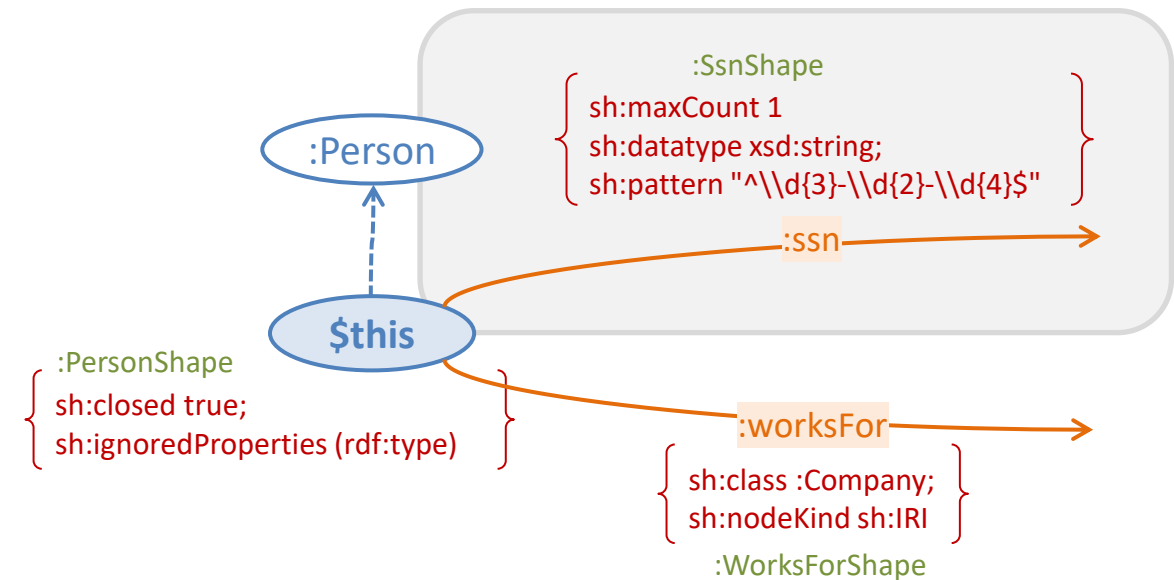- ≡ a shape that is the subject of a sh:path statement.
- A property shape has exactly one sh:path statement.

Shapes Graph

```
:PersonShape  a sh:NodeShape;
    sh:targetClass :Person ;
    sh:property :SsnShape, :WorksForShape;
    sh:closed true ;
    sh:ignoredProperties ( rdf:type ) .

:SsnShape  a sh:PropertyShape ;
    sh:path :ssn ;
    sh:maxCount 1 ;
    sh:datatype xsd:string ;
    sh:pattern "^\\d{3}-\\d{2}-\\d{4}$" .

:WorksForShape  a sh:PropertyShape ;
    sh:path :worksFor ;
    sh:class :Company ;
    sh:nodeKind sh:IRI .
```

:Person

:SsnShape
sh:maxCount 1
sh:datatype xsd:string;
sh:pattern "^\\d{3}-\\d{2}-\\d{4}$"

:ssn

$this

:PersonShape
sh:closed true;
sh:ignoredProperties (rdf:type)

:worksFor

sh:class :Company;
sh:nodeKind sh:IRI

:WorksForShape

Property Shape

- ≡ a shape that is the subject of a sh:path statement.
- A property shape has exactly one sh:path statement
- and may have its own target declaration but is typically referenced from (one or more) node shape(s) by sh:property statement(s) and then has as focus nodes the focus nodes of that node shape.

Shapes Graph

```
:PersonShape   a sh:NodeShape;
    sh:targetClass :Person ;
    sh:property :SsnShape, :WorksForShape;
    sh:closed true ;
    sh:ignoredProperties ( rdf:type ) .

:SsnShape   a sh:PropertyShape ;
    sh:path :ssn ;
    sh:maxCount 1 ;
    sh:datatype xsd:string ;
    sh:pattern "^\\d{3}-\\d{2}-\\d{4}$" .

:WorksForShape   a sh:PropertyShape ;
    sh:path :worksFor ;
    sh:class :Company ;
    sh:nodeKind sh:IRI .
```

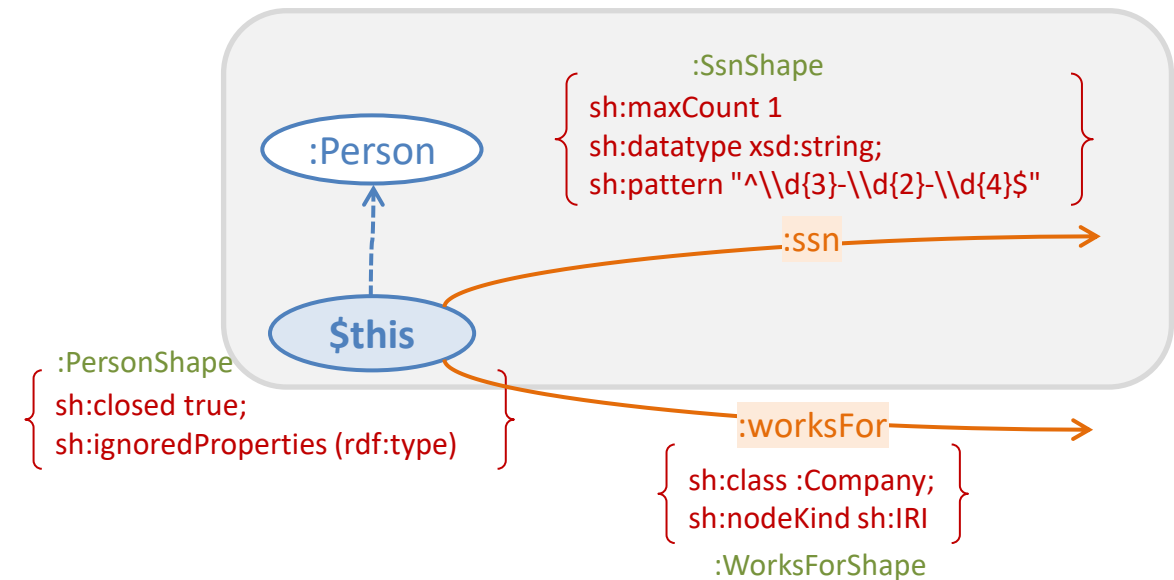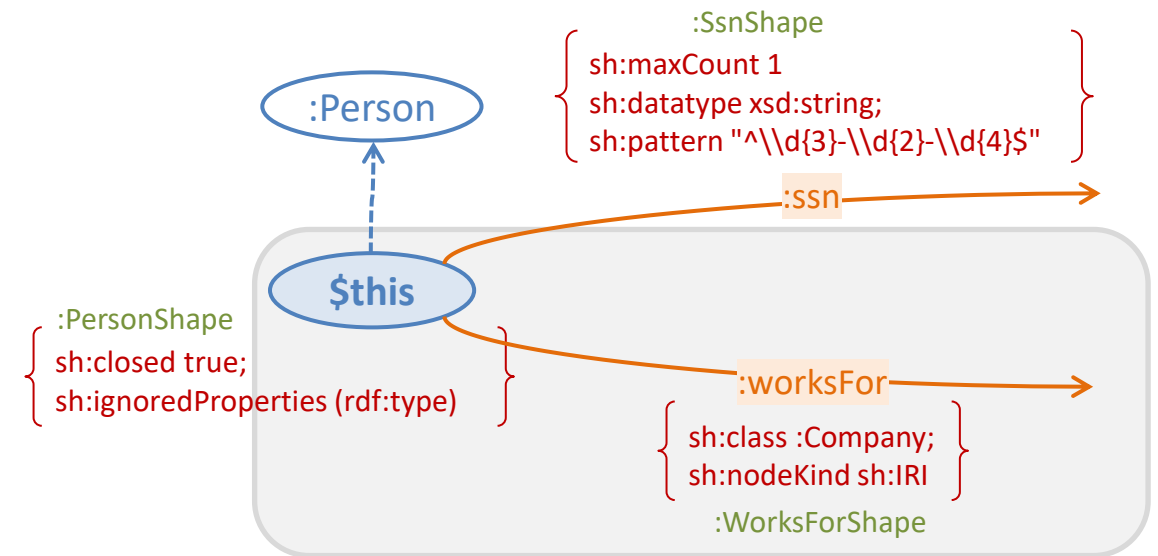**Property Shape**
- ≡ a shape that is the subject of a sh:path statement.
- A property shape has exactly one sh:path statement
- and may have its own target declaration but is typically referenced from (one or more) node shape(s) by sh:property statement(s) and then has as focus nodes the focus nodes of that node shape

Shapes Graph

```
:PersonShape   a sh:NodeShape;
    sh:targetClass :Person ;
    sh:property :SsnShape,  :WorksForShape;
    sh:closed true ;
    sh:ignoredProperties ( rdf:type ) .

:SsnShape   a sh:PropertyShape ;
    sh:path :ssn ;
    sh:maxCount 1 ;
    sh:datatype xsd:string ;
    sh:pattern "^\\d{3}-\\d{2}-\\d{4}$" .

:WorksForShape   a sh:PropertyShape ;
    sh:path :worksFor ;
    sh:class :Company ;
    sh:nodeKind sh:IRI .
```
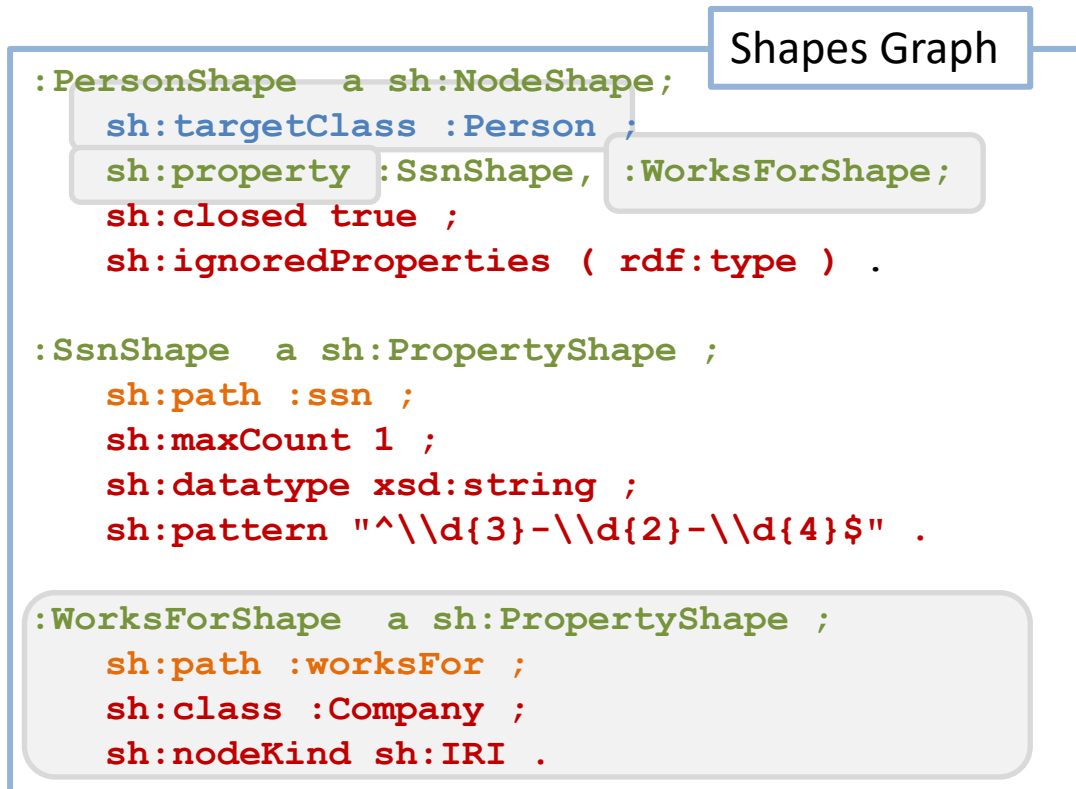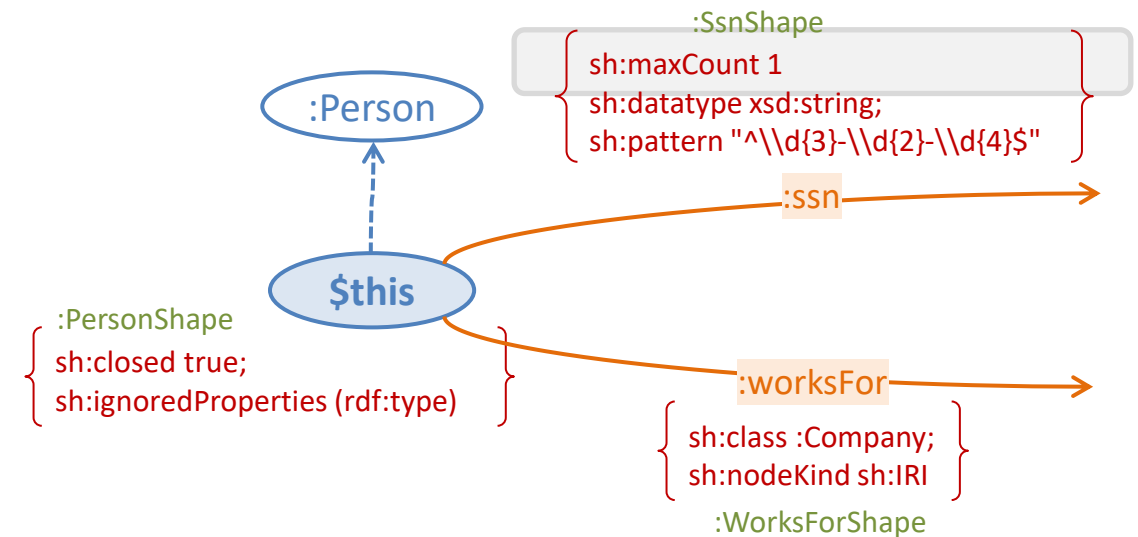
:SsnShape
sh:maxCount 1
sh:datatype xsd:string;
sh:pattern "^\\d{3}-\\d{2}-\\d{4}$"

:Person

:ssn

$this

:PersonShape
sh:closed true;
sh:ignoredProperties (rdf:type)

:worksFor

sh:class :Company;
sh:nodeKind sh:IRI

:WorksForShape

Constraint

- ≡ a condition that is true for every focus node of the shape
  (in a data graph that conforms to the shapes graph).

For example: Every focus node of SsnShape has at most one :ssn statement.

Shapes Graph

```
:PersonShape  a sh:NodeShape;
    sh:targetClass :Person ;
    sh:property :SsnShape, :WorksForShape;
    sh:closed true ;
    sh:ignoredProperties ( rdf:type ) .

:SsnShape  a sh:PropertyShape ;
    sh:path :ssn ;
    sh:maxCount 1 ;
    sh:datatype xsd:string ;
    sh:pattern "^\\d{3}-\\d{2}-\\d{4}$" .

:WorksForShape  a sh:PropertyShape ;
    sh:path :worksFor ;
    sh:class :Company ;
    sh:nodeKind sh:IRI .
```
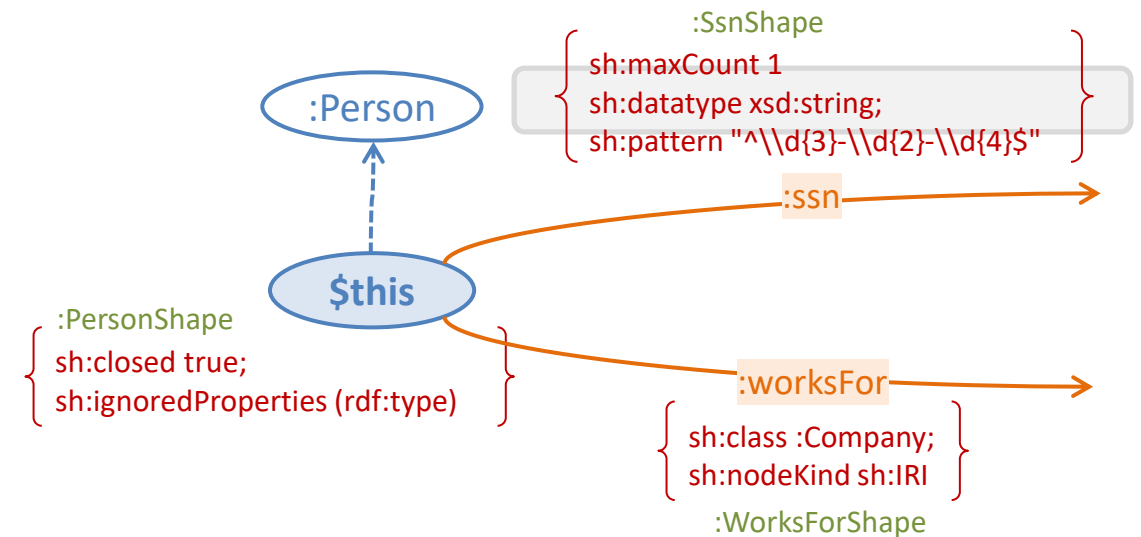
Constraint

- ≡ a condition that is true for every focus node of the shape
  (in a data graph that conforms to the shapes graph).

For example: For every focus node of SsnShape the value
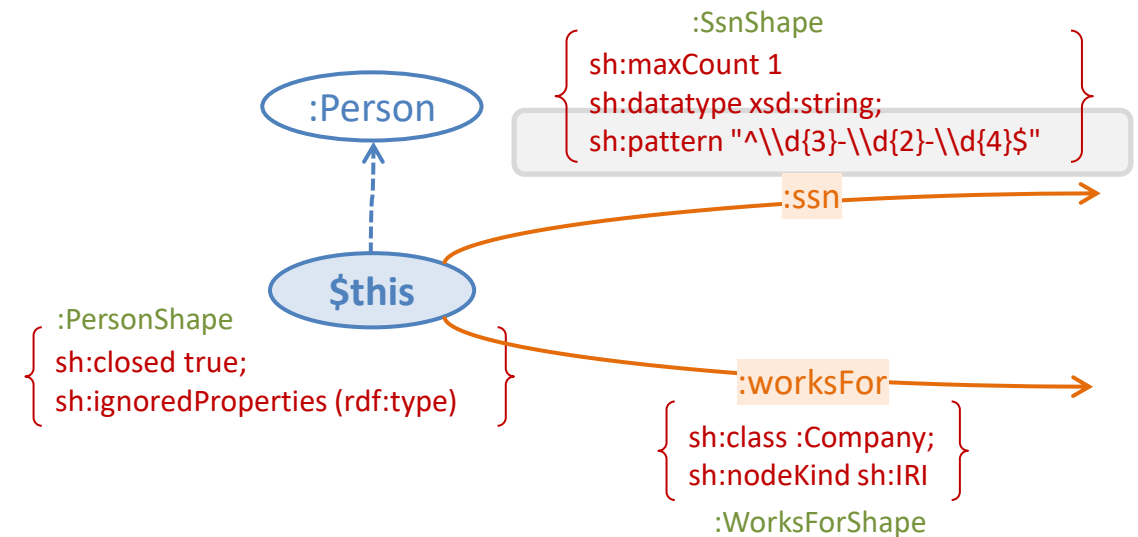(object) of each of its :ssn statement
must be of datatype xsd:string.

Shapes Graph

```
:PersonShape  a sh:NodeShape;
    sh:targetClass :Person ;
    sh:property :SsnShape, :WorksForShape;
    sh:closed true ;
    sh:ignoredProperties ( rdf:type ) .

:SsnShape  a sh:PropertyShape ;
    sh:path :ssn ;
    sh:maxCount 1 ;
    sh:datatype xsd:string ;
    sh:pattern "^\\d{3}-\\d{2}-\\d{4}$" .

:WorksForShape  a sh:PropertyShape ;
    sh:path :worksFor ;
    sh:class :Company ;
    sh:nodeKind sh:IRI .
```


:SsnShape
sh:maxCount 1
sh:datatype xsd:string;
sh:pattern "^\\d{3}-\\d{2}-\\d{4}$"

:ssn

:Person

$this

:PersonShape
sh:closed true;
sh:ignoredProperties (rdf:type)

:worksFor

sh:class :Company;
sh:nodeKind sh:IRI

:WorksForShape

Constraint

- ≡ a condition that is true for every focus node of the shape
  (in a data graph that conforms to the shapes graph).

For example: For every focus node of SsnShape the value
(object) of every :ssn statement
must correspond to a given regular expression.

Shapes Graph
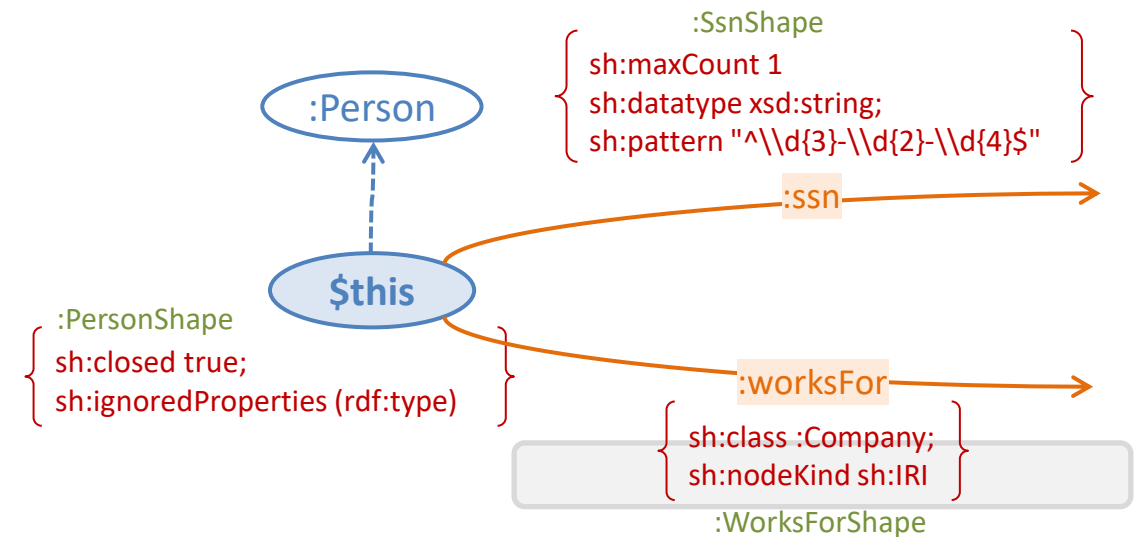
```
:PersonShape  a sh:NodeShape;
    sh:targetClass :Person ;
    sh:property :SsnShape, :WorksForShape;
    sh:closed true ;
    sh:ignoredProperties ( rdf:type ) .

:SsnShape  a sh:PropertyShape ;
    sh:path :ssn ;
    sh:maxCount 1 ;
    sh:datatype xsd:string ;
    sh:pattern "^\\d{3}-\\d{2}-\\d{4}$" .

:WorksForShape  a sh:PropertyShape ;
    sh:path :worksFor ;
    sh:class :Company ;
    sh:nodeKind sh:IRI .
```


:SsnShape
sh:maxCount 1
sh:datatype xsd:string;
sh:pattern "^\\d{3}-\\d{2}-\\d{4}$"

:ssn

:Person

$this

:PersonShape
sh:closed true;
sh:ignoredProperties (rdf:type)

:worksFor

sh:class :Company;
sh:nodeKind sh:IRI

:WorksForShape

Constraint

- ≡ a condition that is true for every focus node of the shape
  (in a data graph that conforms to the shapes graph).

For example: For every focus node of WorksForShape the value (object) of each of its worksFor statements must be of class Company.
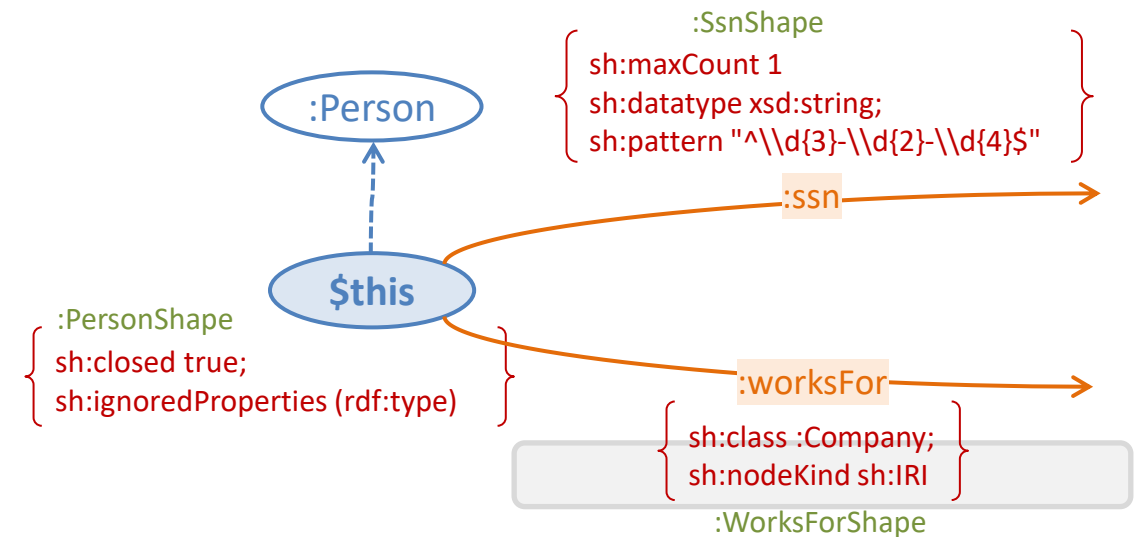
Shapes Graph

```
:PersonShape  a sh:NodeShape;
    sh:targetClass :Person ;
    sh:property :SsnShape, :WorksForShape;
    sh:closed true ;
    sh:ignoredProperties ( rdf:type ) .

:SsnShape  a sh:PropertyShape ;
    sh:path :ssn ;
    sh:maxCount 1 ;
    sh:datatype xsd:string ;
    sh:pattern "^\\d{3}-\\d{2}-\\d{4}$" .

:WorksForShape  a sh:PropertyShape ;
    sh:path :worksFor ;
    sh:class :Company ;
    sh:nodeKind sh:IRI .
```


:SsnShape
sh:maxCount 1
sh:datatype xsd:string;
sh:pattern "^\\d{3}-\\d{2}-\\d{4}$"

:ssn

:Person

$this

:PersonShape
sh:closed true;
sh:ignoredProperties (rdf:type)

:worksFor

sh:class :Company;
sh:nodeKind sh:IRI

:WorksForShape

Constraint

- ≡ a condition that is true for every focus node of the shape
  (in a data graph that conforms to the shapes graph).

For example: For every focus node of WorksForShape the value (object) of each of its :worksFor statements must be a IRI.

Shapes Graph

```
:PersonShape  a sh:NodeShape;
    sh:targetClass :Person ;
    sh:property :SsnShape, :WorksForShape;
    sh:closed true ;
    sh:ignoredProperties ( rdf:type ) .

:SsnShape  a sh:PropertyShape ;
    sh:path :ssn ;
    sh:maxCount 1 ;
    sh:datatype xsd:string ;
    sh:pattern "^\\d{3}-\\d{2}-\\d{4}$" .

:WorksForShape  a sh:PropertyShape ;
    sh:path :worksFor ;
    sh:class :Company ;
    sh:nodeKind sh:IRI .
```

:Person

:SsnShape
sh:maxCount 1
sh:datatype xsd:string;
sh:pattern "^\\d{3}-\\d{2}-\\d{4}$"

:ssn

$this

:PersonShape
sh:closed true;
sh:ignoredProperties (rdf:type)

:worksFor

sh:class :Company;
sh:nodeKind sh:IRI

:WorksForShape

Constraint

- ≡ a condition that is true for every focus node of the shape
  (in a data graph that conforms to the shapes graph).
- one *constraint component*, e.g., the sh:ClosedConstraintComponent, can have multiple *parameters.*

For example: A focus node of PersonShape may only be subject of statements with properties :ssn, :worksFor and rdf:type as

Shapes Graph

```
:PersonShape  a sh:NodeShape;
    sh:targetClass :Person ;
    sh:property :SsnShape, :WorksForShape;
    sh:closed true ;
    sh:ignoredProperties ( rdf:type ) .

:SsnShape  a sh:PropertyShape ;
    sh:path :ssn ;
    sh:maxCount 1 ;
    sh:datatype xsd:string ;
    sh:pattern "^\\d{3}-\\d{2}-\\d{4}$" .

:WorksForShape  a sh:PropertyShape ;
    sh:path :worksFor ;
    sh:class :Company ;
    sh:nodeKind sh:IRI .
```


:SsnShape
sh:maxCount 1
sh:datatype xsd:string;
sh:pattern "^\\d{3}-\\d{2}-\\d{4}$"

:ssn

:Person

$this

:PersonShape
sh:closed true;
sh:ignoredProperties (rdf:type)

:worksFor

sh:class :Company;
sh:nodeKind sh:IRI

:WorksForShape
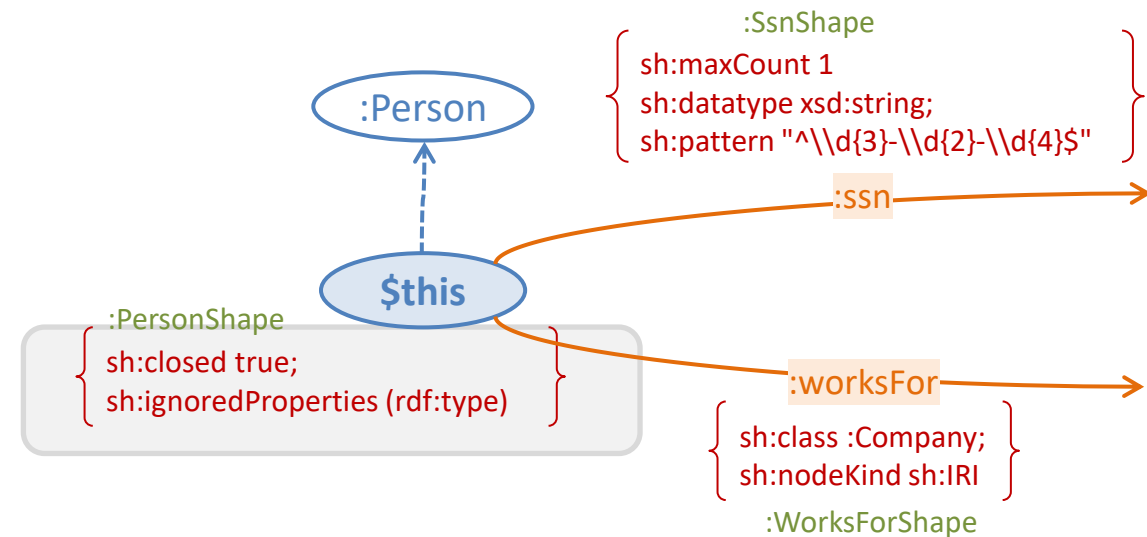
SHACL Property Paths
- In addition to predicate paths (see example), SHACL includes RDF terms to represent a subset of SPARQL property paths: InversePath, SequencePath, AlternativePath, ZeroOrMorePath, OneOrMorePath and ZeroOrOnePath.

Shapes Graph

```
:PersonShape  a sh:NodeShape;
    sh:targetClass :Person ;
    sh:property :SsnShape, :WorksForShape;
    sh:closed true ;
    sh:ignoredProperties ( rdf:type ) .

:SsnShape  a sh:PropertyShape ;
    sh:path :ssn ;
    sh:maxCount 1 ;
    sh:datatype xsd:string ;
    sh:pattern "^\\d{3}-\\d{2}-\\d{4}$" .

:WorksForShape  a sh:PropertyShape ;
    sh:path :worksFor ;
    sh:class :Company ;
    sh:nodeKind sh:IRI .
```


:SsnShape
sh:maxCount 1
sh:datatype xsd:string;
sh:pattern "^\\d{3}-\\d{2}-\\d{4}$"

:Person

:ssn

$this

:PersonShape
sh:closed true;
sh:ignoredProperties (rdf:type)

:worksFor

sh:class :Company;
sh:nodeKind sh:IRI

:WorksForShape
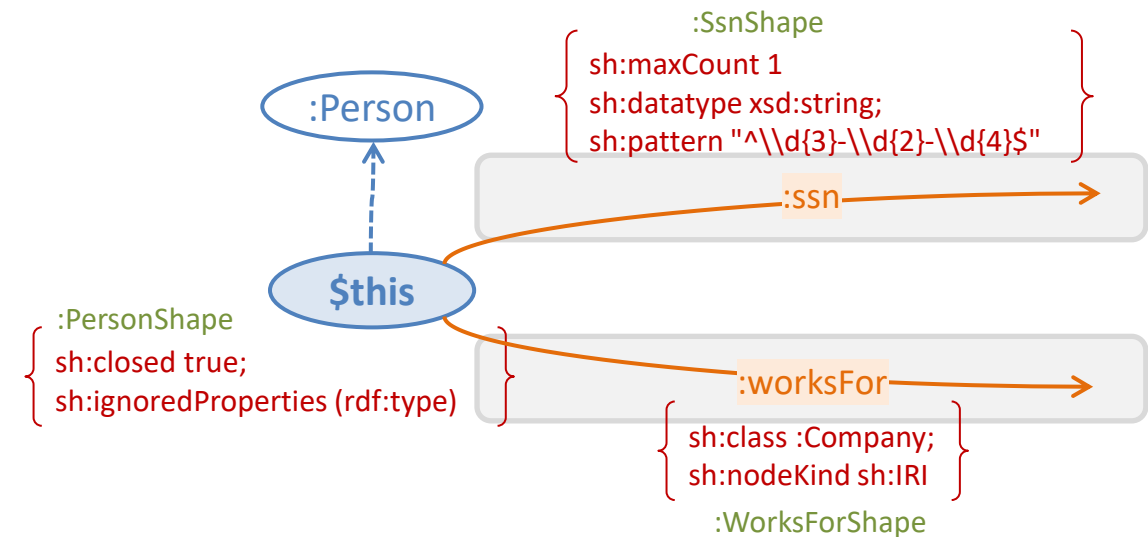
- An RDF term that is validated against a shape using the triples from a data graph is called a **focus node**.

- The set of focus nodes for a shape may be identified as follows:
  - specified in a shape using **target declarations (see Example 2)**
  - specified in any constraint that references a shape in parameters of **shape-expecting constraint parameters**, e.g. sh:property (see Example 1) or sh:node, **(see Example 3)**
  - specified as explicit input to the SHACL processor for validating a specific RDF term against a shape (not exemplified)
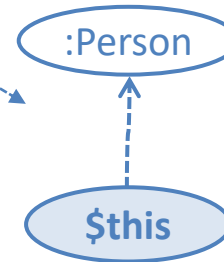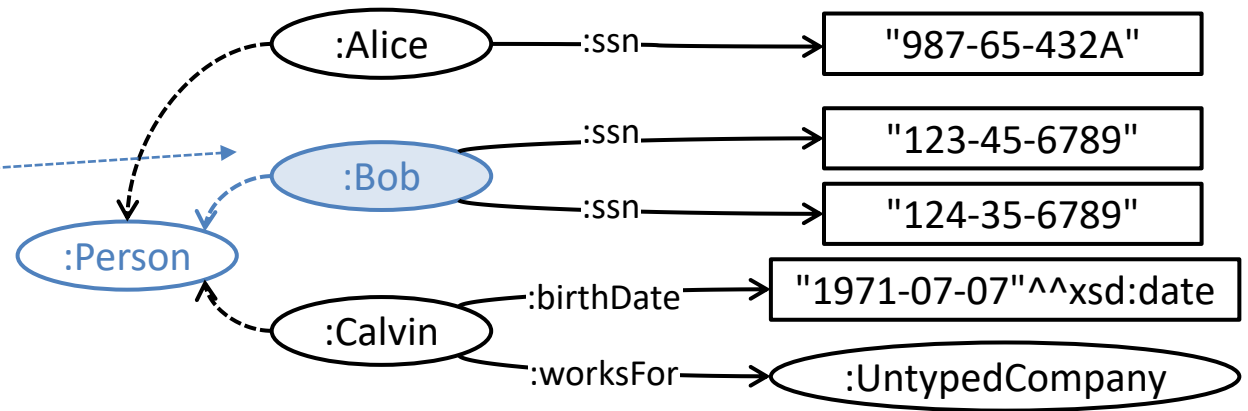
A depiction of a target declaration (a target declaration is a graph pattern to match focus nodes)

Data Graph

```
:Mary    a :Woman ;
         :knows :John, :Bob, :Bello.
:John    a :Man ;
         :knows :Ben .
:Ben     a :Man .
:Man     rdfs:subClassOf :Person.
:Woman   rdfs:subclassOf :Person.
```

Shapes Graph

```
:ManShape  a sh:PropertyShape ;
    sh:targetClass :Man ;
    sh:path :name; sh:minCount 1.
:BenShape a sh:PropertyShape ;
    sh:targetNode :Ben ;
    sh:path :name; sh:minCount 1.
:PersonShape a sh:PropertyShape ;
    sh:targetClass :Person ;
    sh:path :name; sh:minCount 1.
:KnowsSubjectShape  a sh:PropertyShape ;
    sh:targetSubjectsOf :knows ;
    sh:path :name; sh:minCount 1.
:KnowsObjectsShape  a sh:PropertyShape ;
    sh:targetObjectsOf :knows ;
    sh:path :name; sh:minCount 1.
```
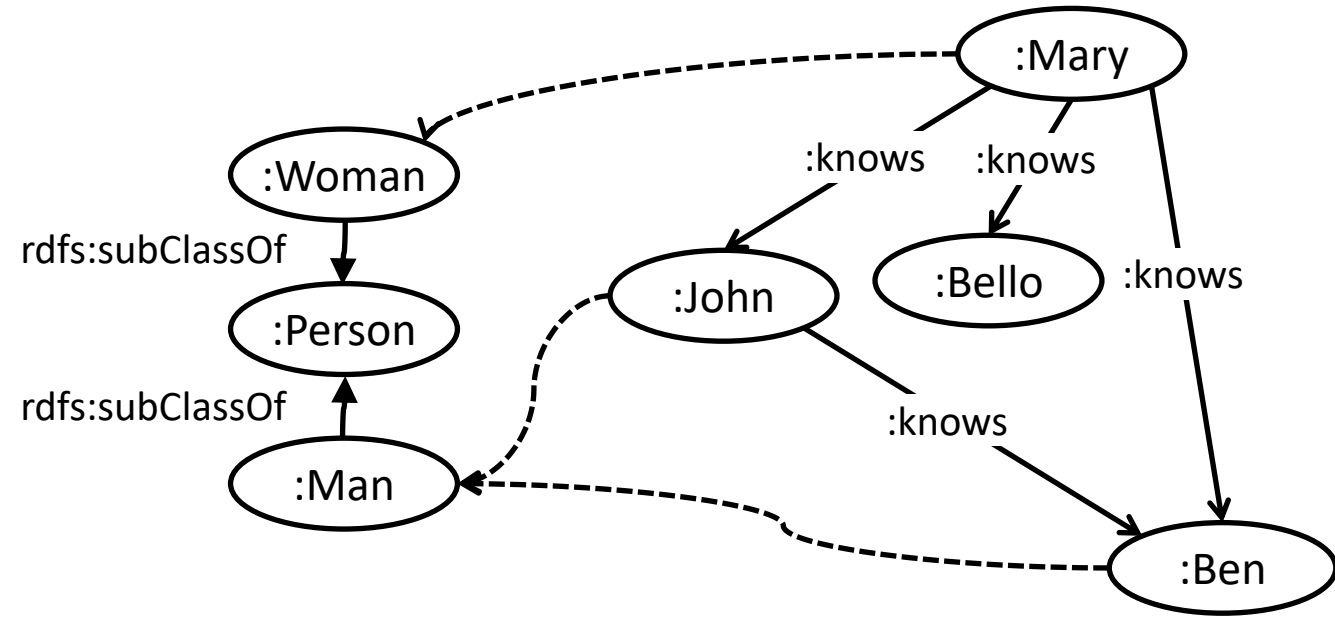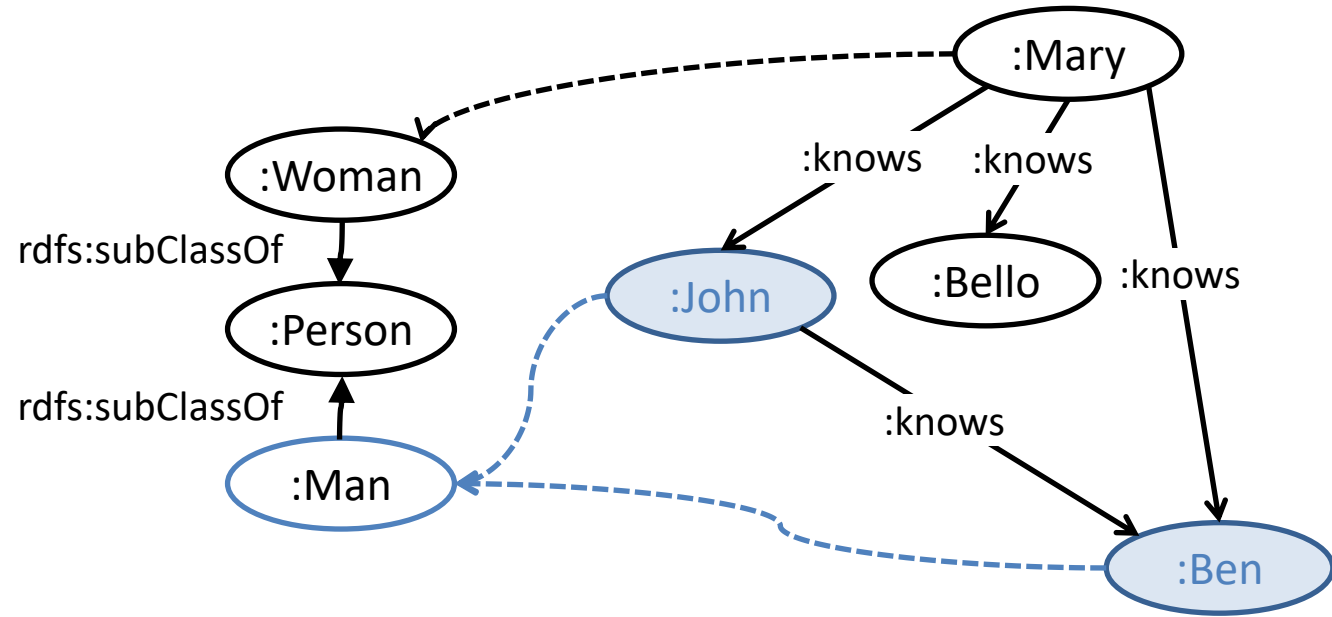
Data Graph

```
:Mary    a :Woman ;
         :knows :John, :Bob, :Bello.
:John    a :Man ;
         :knows :Ben .
:Ben     a :Man .
:Man     rdfs:subClassOf :Person.
:Woman   rdfs:subclassOf :Person.
```

Shapes Graph

```
:ManShape  a sh:PropertyShape ;
    sh:targetClass :Man ;
    sh:path :name; sh:minCount 1.
:BenShape a sh:PropertyShape ;
    sh:targetNode :Ben ;
    sh:path :name; sh:minCount 1.
:PersonShape a sh:PropertyShape ;
    sh:targetClass :Person ;
    sh:path :name; sh:minCount 1.
:KnowsSubjectShape  a sh:PropertyShape ;
    sh:targetSubjectsOf :knows ;
    sh:path :name; sh:minCount 1.
:KnowsObjectsShape  a sh:PropertyShape ;
    sh:targetObjectsOf :knows ;
    sh:path :name; sh:minCount 1.
```


Validation Report

```
[
 a sh:ValidationReport ;
 sh:conforms false ;
 ...
```

Example 2: Target Declarations
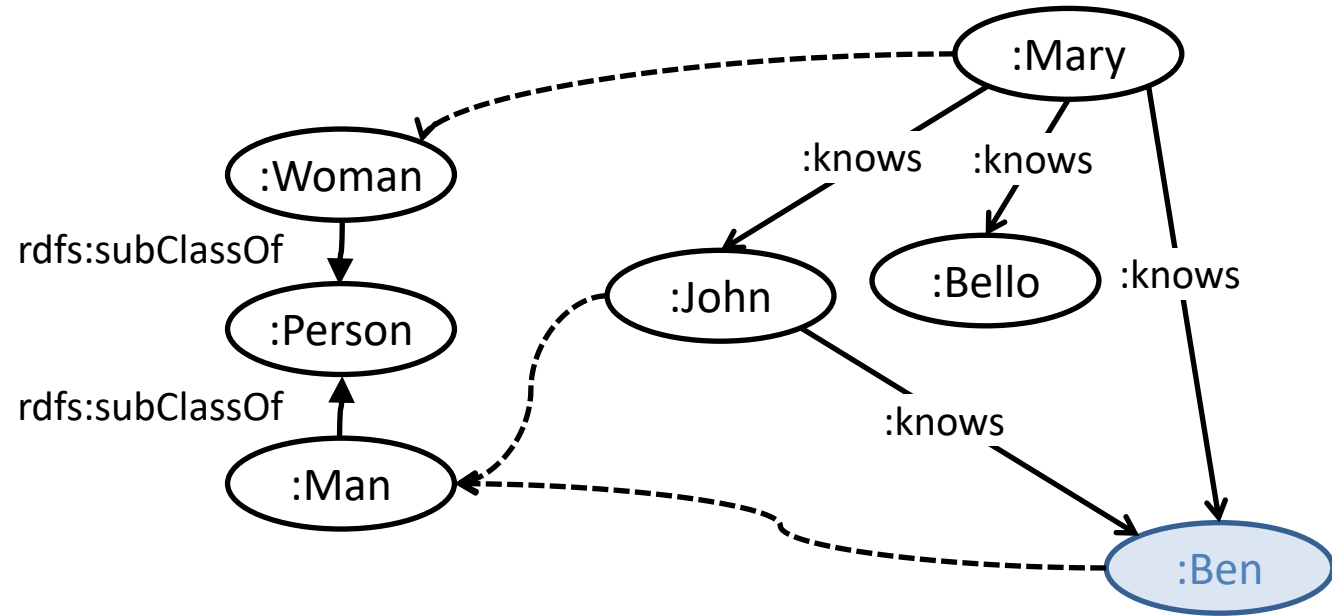
Data Graph

```
:Mary      a :Woman ;
           :knows :John, :Bob, :Bello.
:John      a :Man ;
           :knows :Ben .
:Ben       a :Man .
:Man       rdfs:subClassOf :Person.
:Woman     rdfs:subClassOf :Person.
```

Shapes Graph

```
:ManShape  a sh:PropertyShape ;
    sh:targetClass :Man ;
    sh:path :name; sh:minCount 1.
:BenShape a sh:PropertyShape ;
    sh:targetNode :Ben ;
    sh:path :name; sh:minCount 1.
:PersonShape a sh:PropertyShape ;
    sh:targetClass :Person ;
    sh:path :name; sh:minCount 1.
:KnowsSubjectShape  a sh:PropertyShape ;
    sh:targetSubjectsOf :knows ;
    sh:path :name; sh:minCount 1.
:KnowsObjectsShape  a sh:PropertyShape ;
    sh:targetObjectsOf :knows ;
    sh:path :name; sh:minCount 1.
```

Validation Report

```
...
sh:result [ ...
  sh:focusNode    :Ben ;
  sh:sourceShape  :ManShape ] ;
sh:result [ ...
  sh:focusNode    :John ;
  sh:sourceShape  :ManShape ] ;
...
```

Data Graph

```
:Mary      a :Woman ;
           :knows :John, :Bob, :Bello.
:John      a :Man ;
           :knows :Ben .
:Ben       a :Man .
:Man       rdfs:subClassOf :Person.
:Woman     rdfs:subclassOf :Person.
```

Shapes Graph

```
:ManShape  a sh:PropertyShape ;
    sh:targetClass :Man ;
    sh:path :name; sh:minCount 1.
:BenShape a sh:PropertyShape ;
    sh:targetNode :Ben ;
    sh:path :name; sh:minCount 1.
:PersonShape a sh:PropertyShape ;
    sh:targetClass :Person ;
    sh:path :name; sh:minCount 1.
:KnowsSubjectShape  a sh:PropertyShape ;
    sh:targetSubjectsOf :knows ;
    sh:path :name; sh:minCount 1.
:KnowsObjectsShape  a sh:PropertyShape ;
    sh:targetObjectsOf :knows ;
    sh:path :name; sh:minCount 1.
```

$this
= :Ben

Validation Report

```
 ...
sh:result [ ...
   sh:focusNode    :Ben ;
   sh:sourceShape  :BenShape ] ;
 ...
```
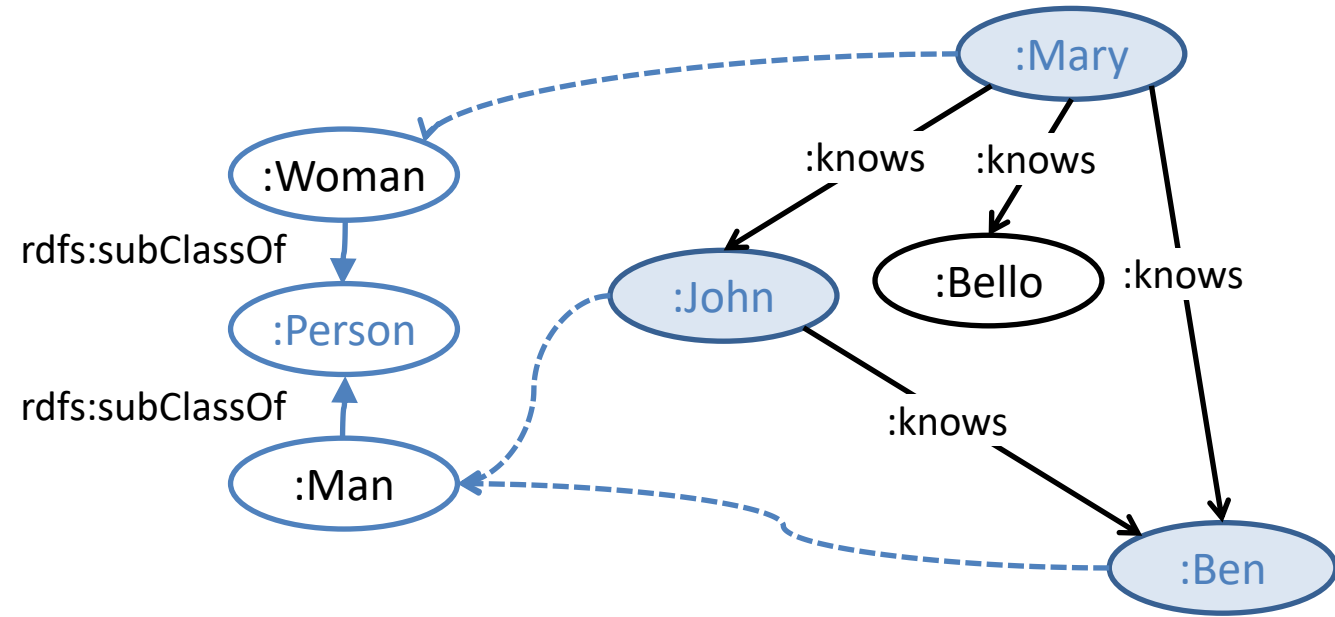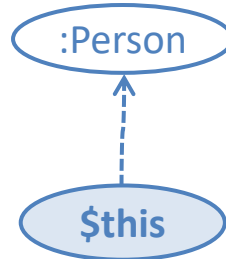
Example 2: Target Declarations

Data Graph

```
:Mary     a :Woman ;
          :knows :John, :Bob, :Bello.
:John     a :Man ;
          :knows :Ben .
:Ben      a :Man .
:Man      rdfs:subClassOf :Person.
:Woman    rdfs:subClassOf :Person.
```

Shapes Graph

```
:ManShape  a sh:PropertyShape ;
    sh:targetClass :Man ;
    sh:path :name; sh:minCount 1.
:BenShape a sh:PropertyShape ;
    sh:targetNode :Ben ;
    sh:path :name; sh:minCount 1.
:PersonShape a sh:PropertyShape ;
    sh:targetClass :Person ;
    sh:path :name; sh:minCount 1.
:KnowsSubjectShape  a sh:PropertyShape ;
    sh:targetSubjectsOf :knows ;
    sh:path :name; sh:minCount 1.
:KnowsObjectsShape  a sh:PropertyShape ;
    sh:targetObjectsOf :knows ;
    sh:path :name; sh:minCount 1.
```

Validation Report

```
 ...
 sh:result [ ...
   sh:focusNode    :Mary ;
   sh:sourceShape  :PersonShape ] ;
 sh:result [ ...
   sh:focusNode    :John ;
   sh:sourceShape  :PersonShape ] ;
 sh:result [ ...
   sh:focusNode    :Ben ;
   sh:sourceShape  :PersonShape ] ;
...
```

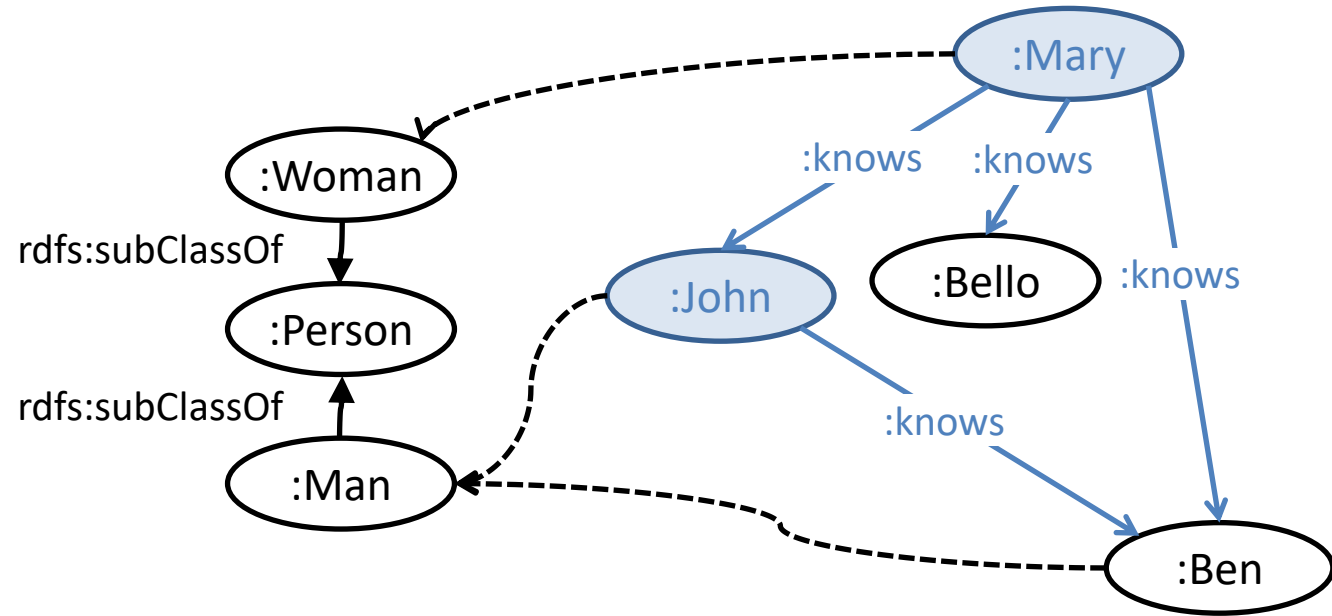## Data Graph

```
:Mary      a :Woman ;
           :knows :John, :Bob, :Bello.
:John      a :Man ;
           :knows :Ben .
:Ben       a :Man .
:Man       rdfs:subClassOf :Person.
:Woman     rdfs:subclassOf :Person.
```

Shapes Graph

```
:ManShape  a sh:PropertyShape ;
    sh:targetClass :Man ;
    sh:path :name; sh:minCount 1.
:BenShape a sh:PropertyShape ;
    sh:targetNode :Ben ;
    sh:path :name; sh:minCount 1.
:PersonShape a sh:PropertyShape ;
    sh:targetClass :Person ;
    sh:path :name; sh:minCount 1.
:KnowsSubjectShape  a sh:PropertyShape ;
    sh:targetSubjectsOf :knows ;
    sh:path :name; sh:minCount 1.
:KnowsObjectsShape  a sh:PropertyShape ;
    sh:targetObjectsOf :knows ;
    sh:path :name; sh:minCount 1.
```

$this

:knows

Validation Report

```
...
sh:result [ ...
  sh:focusNode    :Mary ;
  sh:sourceShape  :KnowsSubjectShape ] ;
sh:result [ ...
  sh:focusNode    :John ;
  sh:sourceShape  :KnowsSubjectShape ] ;
...
```
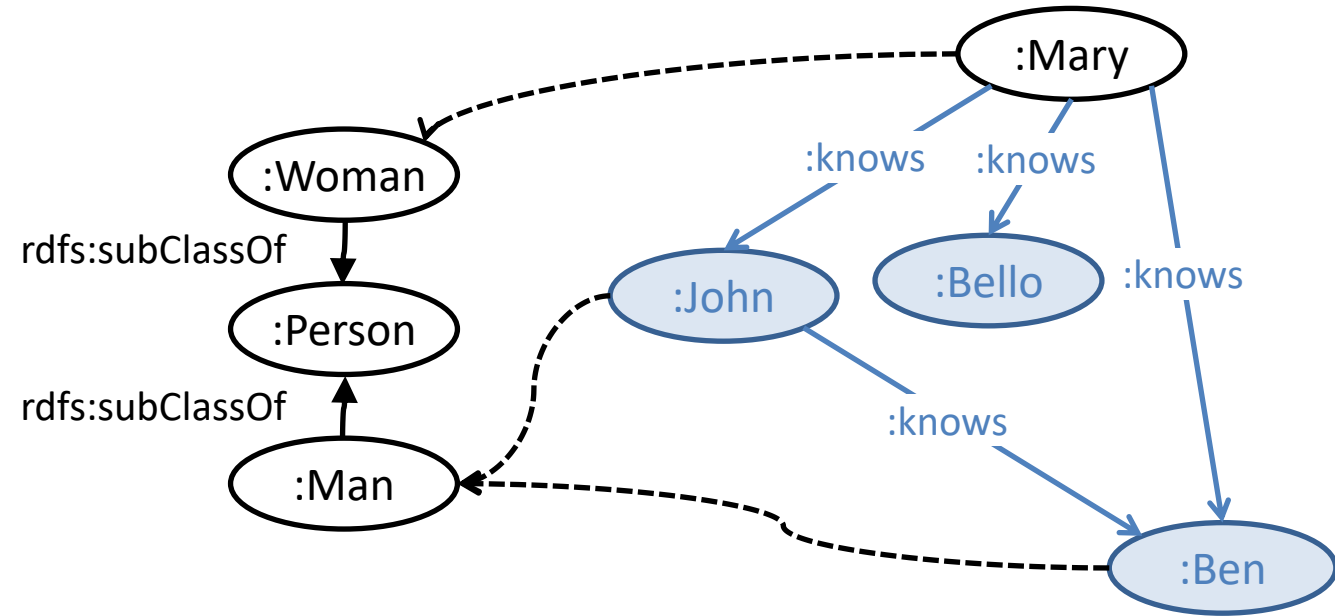
Data Graph

```
:Mary     a :Woman ;
          :knows :John, :Bob, :Bello.
:John     a :Man ;
          :knows :Ben .
:Ben      a :Man .
:Man      rdfs:subClassOf :Person.
:Woman    rdfs:subclassOf :Person.
```

Shapes Graph

```
:ManShape  a sh:PropertyShape ;
    sh:targetClass :Man ;
    sh:path :name; sh:minCount 1.
:BenShape a sh:PropertyShape ;
    sh:targetNode :Ben ;
    sh:path :name; sh:minCount 1.
:PersonShape a sh:PropertyShape ;
    sh:targetClass :Person ;
    sh:path :name; sh:minCount 1.
:KnowsSubjectShape  a sh:PropertyShape ;
    sh:targetSubjectsOf :knows ;
    sh:path :name; sh:minCount 1.
:KnowsObjectsShape  a sh:PropertyShape ;
    sh:targetObjectsOf :knows ;
    sh:path :name; sh:minCount 1.
```

:knows → $this

rdfs:subClassOf

rdfs:subClassOf

:Woman → :Person → :Man

:Mary :knows :John, :Bello, :Ben

Validation Report

```
 ...
 sh:result [ ...
   sh:focusNode    :Bello ;
   sh:sourceShape  :KnowsObjectShape ] ;
 sh:result [ ...
   sh:focusNode    :John ;
   sh:sourceShape  : KnowsObjectShape ] ;
 sh:result [ ...
   sh:focusNode    :Ben ;
   sh:sourceShape  : KnowsObjectShape ]
].
```
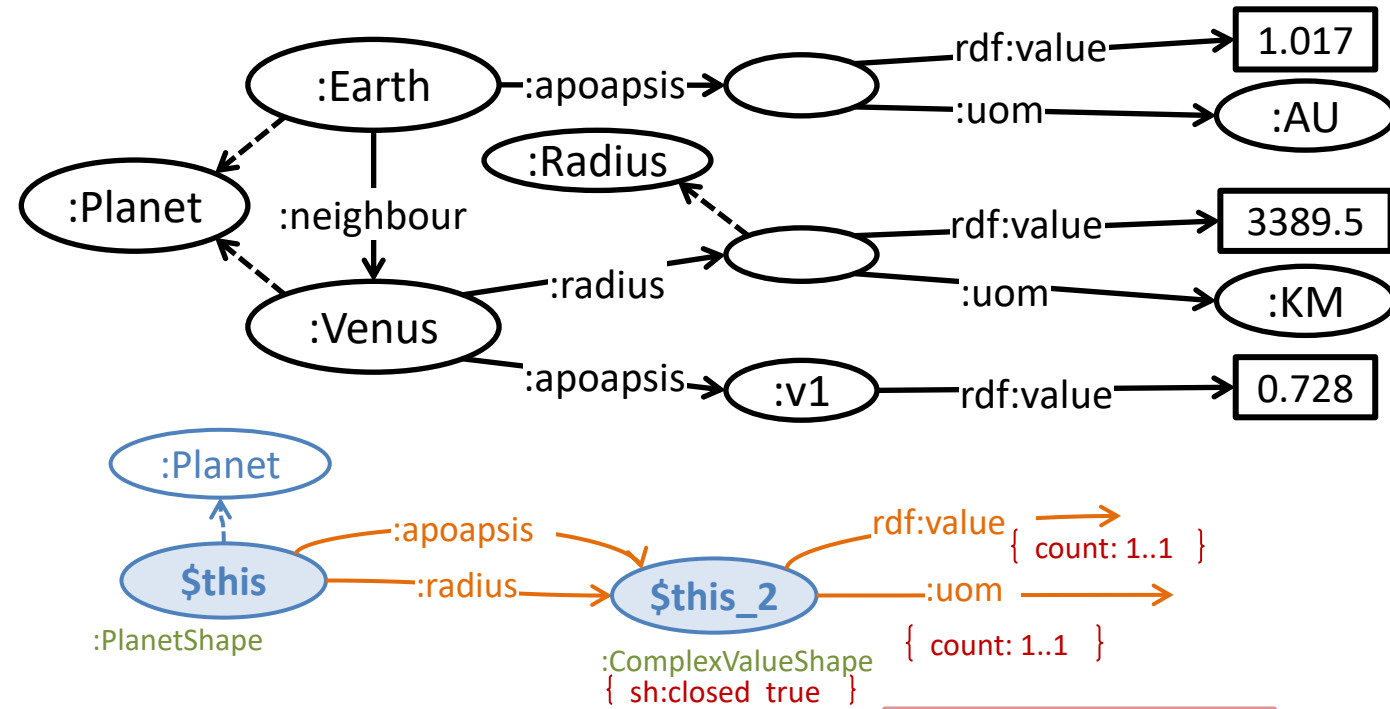
Example 3: References to Node Shapes

Data Graph

```
:Earth a :Planet ;
  :apoapsis [rdf:value 1.017; :uom :AU];
  :neighbour :Venus .
:Venus a :Planet;
  :apoapsis :v1;
  :radius [rdf:value 3389.5; :uom :KM ;
                    a :Radius] .
:v1 rdf:value 0.728.
```

Shapes Graph

```
:PlanetShape  a sh:NodeShape ;
  sh:targetClass :Planet ;
  sh:property [
    sh:path :radius;
    sh:node :ComplexValueShape ];
  sh:property [
    sh:path :apoapsis;
    sh:node :ComplexValueShape ].

:ComplexValueShape a sh:NodeShape ;
  sh:closed true;
  sh:property [
    sh:path rdf:value;
    sh:minCount 1; sh:maxCount 1 ];
  sh:property [
    sh:path :uom;
    sh:minCount 1; sh:maxCount 1 ].
```

Validation Report

```
[
 a sh:ValidationReport ;
 sh:conforms  false ;
 ...
```
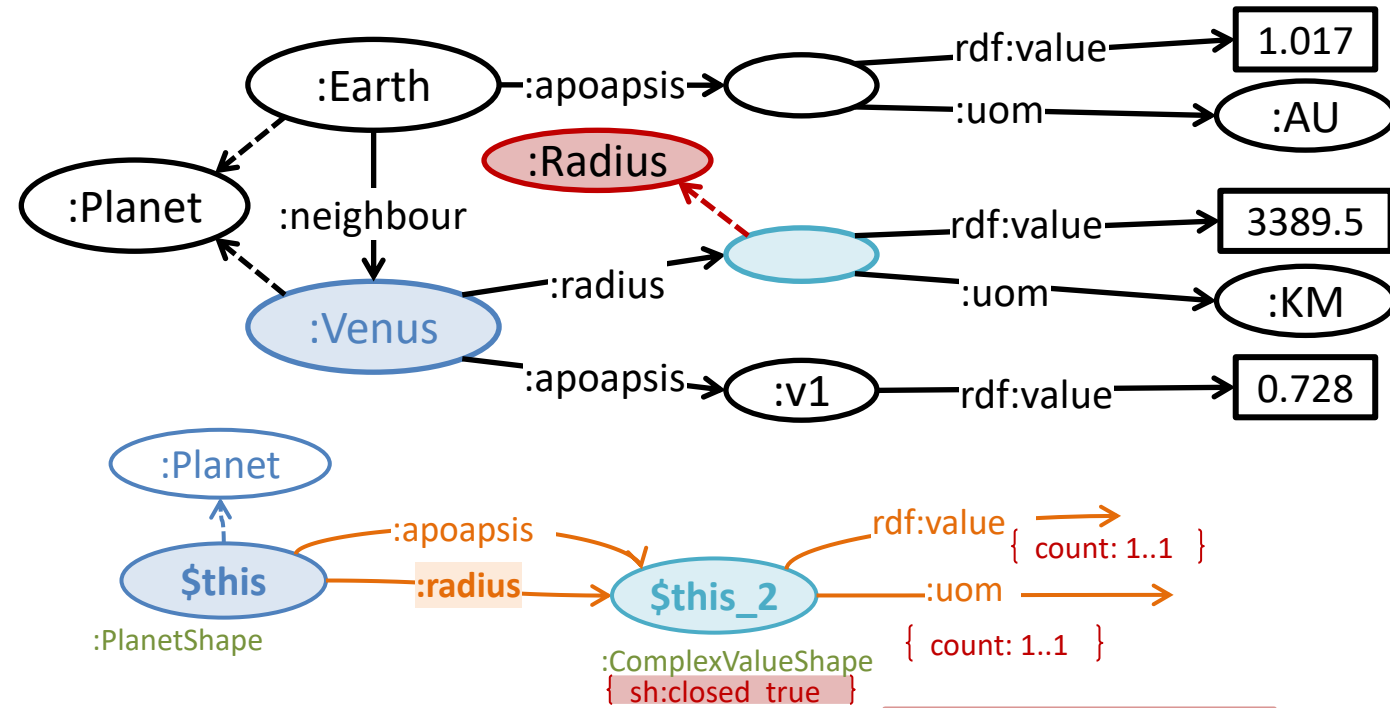
Data Graph

```
:Earth a :Planet ;
  :apoapsis [rdf:value 1.017; :uom :AU];
  :neighbour :Venus .
:Venus a :Planet;
  :apoapsis :v1;
  :radius [rdf:value 3389.5; :uom :KM,
                    a :Radius] .
:v1 rdf:value 0.728.
```

Shapes Graph

```
:PlanetShape  a sh:NodeShape ;
  sh:targetClass :Planet ;
  sh:property [
    sh:path :radius;
    sh:node :ComplexValueShape ];
  sh:property [
    sh:path :apoapsis;
    sh:node :ComplexValueShape ].

:ComplexValueShape a sh:NodeShape ;
  sh:closed true;
  sh:property [
    sh:path rdf:value;
    sh:minCount 1; sh:maxCount 1 ];
  sh:property [
    sh:path :uom;
    sh:minCount 1; sh:maxCount 1 ].
```

Validation Report

```
...
sh:result [ a sh:ValidationResult ;
  sh:focusNode      :Venus ;
  sh:resultMessage "Node[<http://example.com/ns#Complex-
    ValueShape>] at focusNode _:B12..." ;
  sh:resultPath      :radius ;
  sh:resultSeverity sh:Violation ;
  sh:sourceConstraintComponent sh:NodeConstraintComponent;
  sh:sourceShape      []  ;
  sh:value            []
]; ...
```
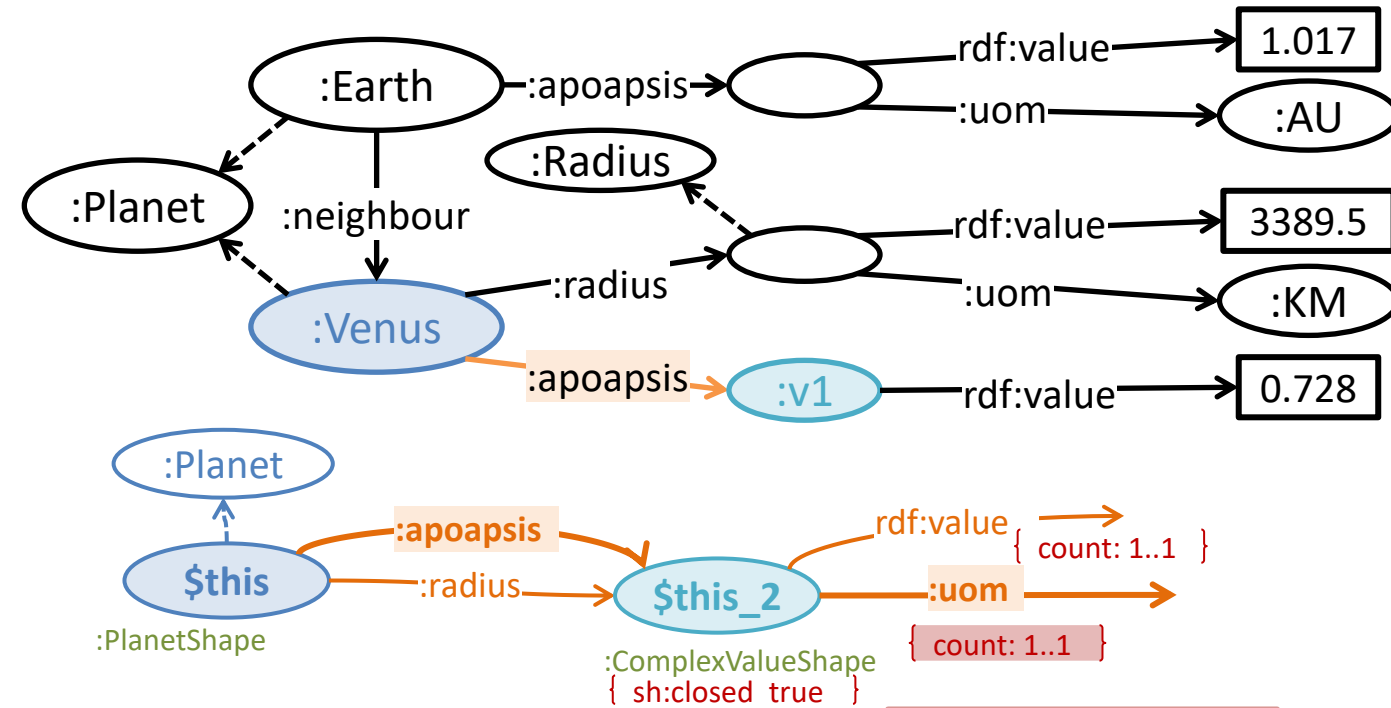
Data Graph

```
:Earth a :Planet ;
  :apoapsis [rdf:value 1.017; :uom :AU];
  :neighbour :Venus .
:Venus a :Planet;
  :apoapsis :v1;
  :radius [rdf:value 3389.5; :uom :KM,
                    a :Radius] .
:v1 rdf:value 0.728.
```


Shapes Graph

```
:PlanetShape  a sh:NodeShape ;
  sh:targetClass :Planet ;
  sh:property [
    sh:path :radius;
    sh:node :ComplexValueShape ];
  sh:property [
    sh:path :apoapsis;
    sh:node :ComplexValueShape ].

:ComplexValueShape a sh:NodeShape ;
  sh:closed true;
  sh:property [
    sh:path rdf:value;
    sh:minCount 1; sh:maxCount 1 ];
  sh:property [
    sh:path :uom;
    sh:minCount 1; sh:maxCount 1 ].
```


Validation Report

```
  ...
  sh:result  [ a sh:ValidationResult ;
    sh:focusNode          :Venus ;
    sh:resultMessage "Node[<http://example.com/ns#Complex-
      ValueShape>] at focusNode <http://example.com/ns#v1>";
    sh:resultPath         :apoapsis ;
    sh:resultSeverity  sh:Violation ;
    sh:sourceConstraintComponent sh:NodeConstraintComponent;
    sh:sourceShape        []  ;
    sh:value              :v1
]] .
```

Data Graph

```
:Earth a :Planet;
  :radius [rdf:value 3389.5; :uom :KM].
:Venus a :Planet;
  :radius [:nilReason "unknown"].
:Mars a :Planet;
  :radius [rdf:value 234.4].
:Mercury a :Planet;
  :radius [rdf:value 3389.5; :uom :KM; :nilReason "other"].
```

The value of radius statements of instances of
Planet is either a complex value or a node with a
nil-reason.

Shapes Graph

```
:PlanetShape  a sh:NodeShape ;
  sh:targetClass :Planet ;
  sh:property [
    sh:path :radius;
    sh:xone (:ComplexValueShape :NilShape)
  ].

:ComplexValueShape a sh:NodeShape ;
  sh:closed true;
  sh:property [
    sh:path rdf:value;
    sh:minCount 1; sh:maxCount 1 ];
  sh:property [
    sh:path :uom;
    sh:minCount 1; sh:maxCount 1 ].

:NilShape a sh:NodeShape ;
  sh:closed true;
  sh:property [
    sh:path :nilReason;
    sh:in ( "unknown" "inapplicable" "other" );
    sh:minCount 1; sh:maxCount 1 ].
```

Example 4: Logical Constraint Components

Data Graph

```
:Earth a :Planet;
  :radius [rdf:value 3389.5; :uom :KM].  👍
:Venus a :Planet;
  :radius [:nilReason "unknown"].  👍
:Mars a :Planet;
  :radius [rdf:value 234.4].  ⚡
:Mercury a :Planet;
  :radius [rdf:value 3389.5; :uom :KM; :nilReason "other"].  ⚡
```

Validation Report

```
[ a sh:ValidationReport ;
  sh:conforms  false ;
  sh:result  [
    a                 sh:ValidationResult ;
    sh:focusNode      :Mars ;
    sh:resultMessage  "Xone has 0 conforming shapes ..." ;
    sh:resultPath     :radius ;
    ... ] ;
  sh:result [
    a  sh:ValidationResult ;
    sh:focusNode      :Mercury ;
    sh:resultMessage  "Xone has 0 conforming shapes ..." ;
    sh:resultPath     :radius ;
    ... ]
] .
```

Shapes Graph

```
:PlanetShape  a sh:NodeShape ;
  sh:targetClass :Planet ;
  sh:property [
    sh:path :radius;
    sh:xone (:ComplexValueShape :NilShape)
  ].

:ComplexValueShape a sh:NodeShape ;
  sh:closed true;
  sh:property [
    sh:path rdf:value;
    sh:minCount 1; sh:maxCount 1 ];
  sh:property [
    sh:path :uom;
    sh:minCount 1; sh:maxCount 1 ].

:NilShape a sh:NodeShape ;
  sh:closed true;
  sh:property [
    sh:path :nilReason;
    sh:in ( "unknown" "inapplicable" "other" );
    sh:minCount 1; sh:maxCount 1 ].
```

Basic vocabulary

- sh:Shape
  - sh:targetClass

- sh:NodeShape
  - sh:property

- sh:PropertyShape
  - sh:path
  - sh:maxCount
  - sh:minCount
  - sh:class
  - sh:datatype

Further relevant vocabulary

- sh:Shape
  - sh:targetNode
  - sh:targetSubjectOf
  - sh:targetObjectOf

- sh:NodeShape
  - sh:closed
  - sh:ignoredProperties

- sh:PropertyShape
  - sh:pattern
  - sh:nodeKind
  - sh:in

Advanced relevant vocabulary

- sh:PropertyShape
  - sh:node
  - sh:xone

- sh:Shape
  - sh:rule

- sh:SPARQLRule
  - sh:construct

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

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

Further Reading

- https://en.wikipedia.org/wiki/SHACL

- https://www.topquadrant.com/technology/shacl/tutorial/

- Shapes Constraint Language (SHACL), W3C Recommendation 20 July 2017, https://www.w3.org/TR/shacl/
    - SHACL Property Paths
    - Non-Validating Property Shape Characteristics
    - Core Constraint Components
   - SPARQL-based Constraints
    - ...

SHACL Rules

Bernd Neumayr
Johannes Kepler University Linz

based on
Shapes Constraint Language (SHACL) - W3C Recommendation 20 July 2017 - https://www.w3.org/TR/shacl/
SHACL Advanced Features - W3C Working Group Note 08 June 2017 - https://www.w3.org/TR/shacl-af/

Semantic Web Stack

KG Technology Stack
(based on Semantic Web Technology)

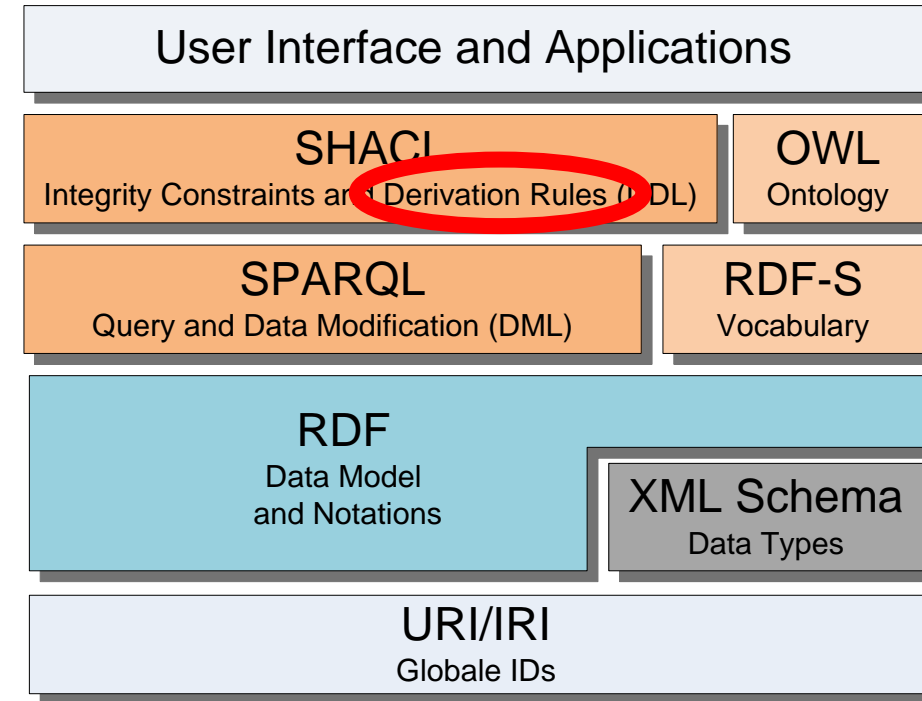# Knowledge Graph Technology Stack

Semantic Web Stack
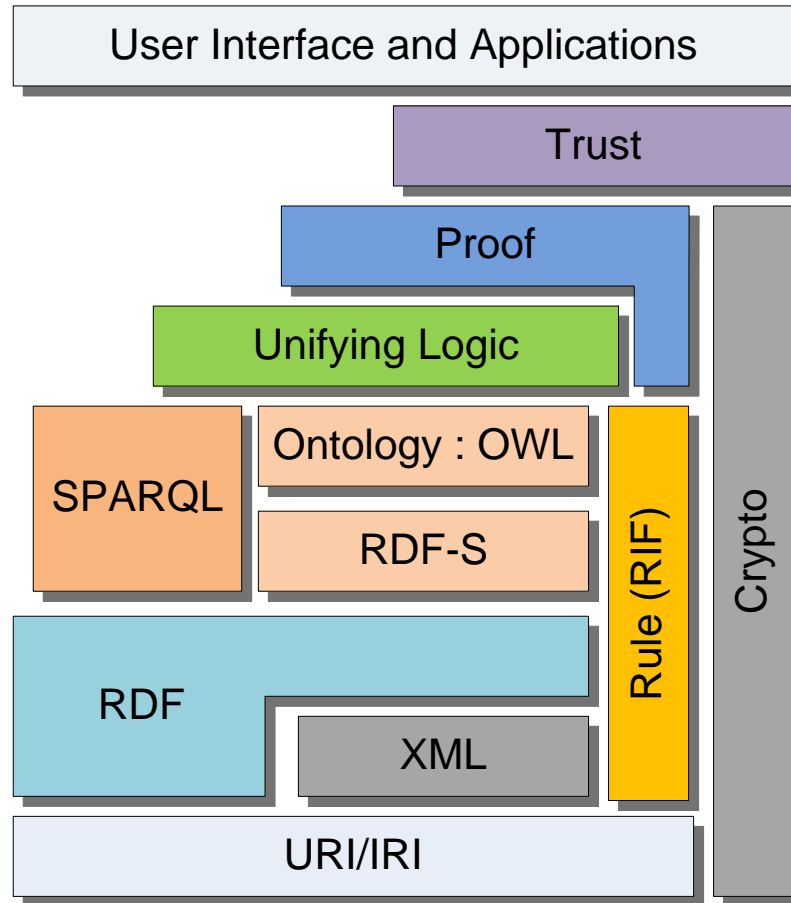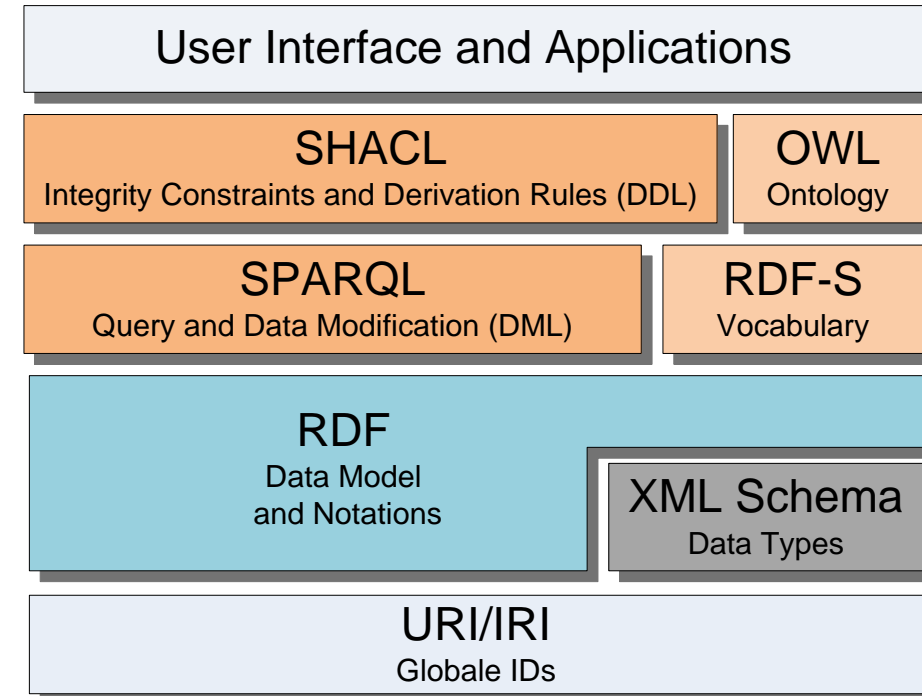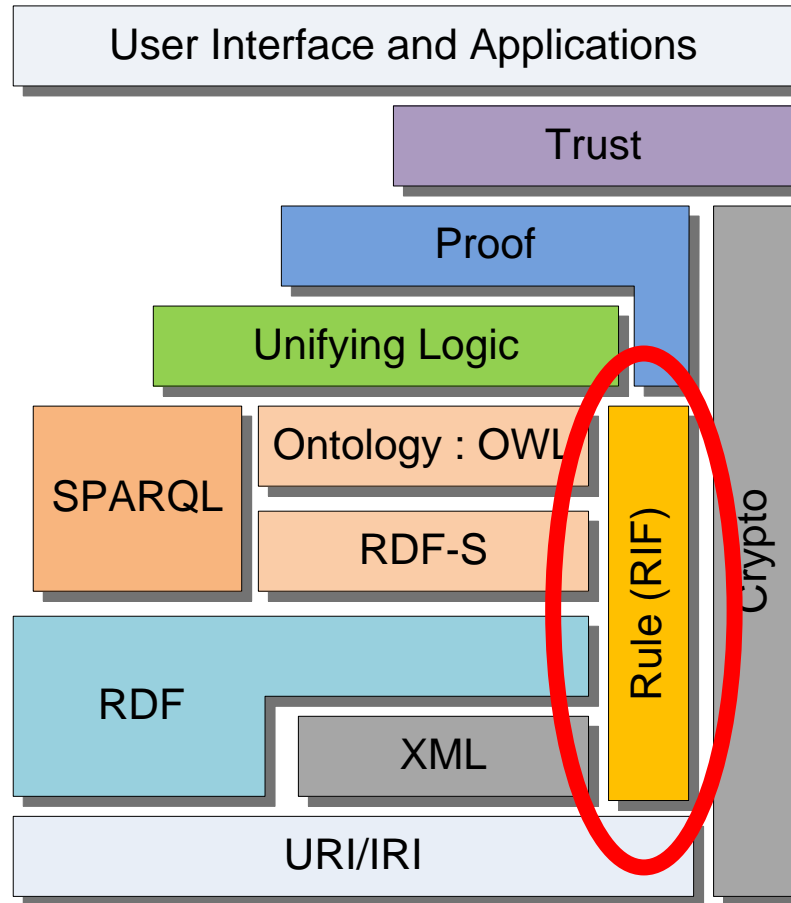
KG Technology Stack
(based on Semantic Web Technology)

Motivation for SHACL Rules:
- Full integration with RDF/SPARQL/SHACL
- Express rule sets as RDF graphs
- Reuse SPARQL and SHACL

- **SHACL property paths** allow to represent a subset of SPARQL property paths as RDF nodes
  https://www.w3.org/TR/shacl/#property-paths

- **SHACL functions** declare operations that produce an RDF term based on zero or more parameters and a data graph.
  https://www.w3.org/TR/shacl-af/#functions

- **Node expressions** are declared as RDF nodes in a shapes graph and instruct a SHACL engine how to compute a set of nodes for a given focus node.
  https://www.w3.org/TR/shacl-af/#node-expressions

- **SHACL rules** build on SHACL to form a light-weight RDF vocabulary for the exchange of rules that can be used to derive inferred RDF triples from existing asserted triples.
  https://www.w3.org/TR/shacl-af/#rules

SHACL property paths allow to represent a subset of SPARQL property paths as RDF nodes
https://www.w3.org/TR/shacl/#property-paths , e.g.,

SPARQL Property Path	**SHACL Property Path**
`^ex:parent`	`[ sh:inversePath ex:parent ]`
`ex:parent/ex:firstName`	`( ex:parent ex:firstName )`
`rdf:type/rdfs:subClassOf*`	`( rdf:type [ sh:zeroOrMorePath rdfs:subClassOf ] )`
`ex:father|ex:mother`	`[ sh:alternativePath ( ex:father ex:mother ) ]`
`rdfs:subClassOf+`	`[ sh:oneOrMorePath rdfs:subClassOf ]`