Informatiemodellering met SHACL

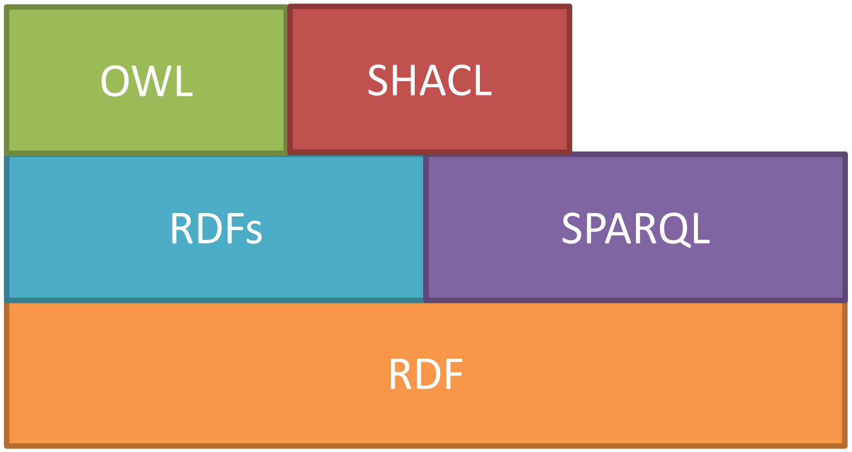
Jan Voskuil
Jesse Bakker

Vandaag:

- Introductie
- SHACL Overview
- De rol van SHACL
- SHACL SPARQL
- SHACL Rules

© Copyright 2018 TopQuadrant Inc.

TopQuadrant™



TAXONIC

© Copyright 2018 TopQuadrant Inc.

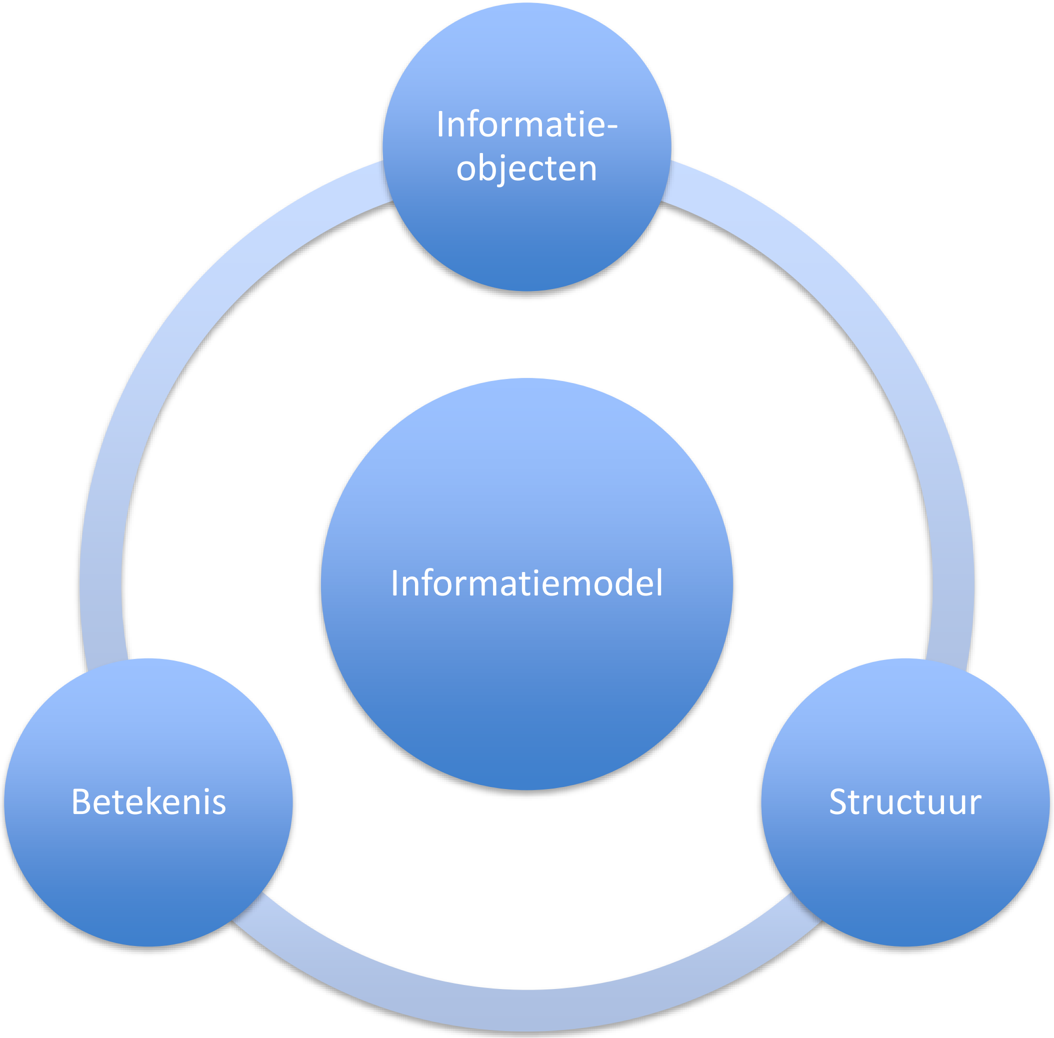
Informatiemodellering

- Een informatiemodel is een beschrijving van welke informatieobjecten er zijn en wat hun structuur en betekenis is.

> https://wiki.nationaalarchief.nl/pagina/DUTO:Informatiemodel

© Copyright 2018 TopQuadrant Inc.

Triniteit


Informatie-
objecten
Informatiemodel
Betekenis
Structuur

SHACL Introduction

If you know OWL: Familiar things you can do using with SHACL

- Specify cardinalities for a property when used with a member of a class
  - Also can do qualified cardinalities (owl:someValuesFrom = min 1 QCR)
  - Closed world meaning
- Specify a range of values for a property when used with a member of a class
  - Similar to owl:allValuesFrom, but closed world
- Combine restrictions (shapes) using logical operators
  - "and" is assumed, by default
  - or, not and xone are available

© Copyright 2018 TopQuadrant Inc.

If you know OWL: Some new things you can do using with SHACL

- Larger pre-built vocabulary for restricting property values
  - min/max, regex, node-kind
- Restricting property value based on the value of another property
- Not limited to a direct property values – can use paths just like in SPARQL
- Restricting resource itself
  - Node-kind, URI, closed shape (with ignore list)

© Copyright 2018 TopQuadrant Inc.

If you know OWL: More new things you can do using with SHACL

- De-activating – useful for re-use and testing
- Defining such restrictions (constraints) not just for a member of a class - for a specific resource/some other grouping of resources
- Extending – declaratively define your own constraint types (components)
- Error messages, some UI generation support, etc.

© Copyright 2018 TopQuadrant Inc.

TopQuadrant™

SHACL Terminology

- Targets (of a shape)
  - determine what resources (or, more generally, RDF graph nodes) are to be validated against a shape
  - during the validation, targets are referred to as ”focus nodes”
- Node Shapes
  - specify conditions a target node itself must comply with
  - used to group property shapes

© Copyright 2018 TopQuadrant Inc.

SHACL Terminology – 2

- Property Shapes
  - specify conditions that related nodes (property values) must comply with
  - for example:
    - target nodes are all resources with type td:Person
    - property shape says that the values of the td:birthDate property for these resources must be dates that are less than 1/1/2018 and there can be only one birth date per person

© Copyright 2018 TopQuadrant Inc.

SHACL Terminology – 3

- Constraint Components
  - predefined CCs in SHACL Core form “SHACL Core vocabulary” e.g., sh:minCount, sh:datatype, sh:pattern, etc.
  - users can create new CCs – domain specific data validation languages
- Shapes Graph, Data Graph
  - These are “roles” – any graph can be declared to be a shapes graph or a data graph

© Copyright 2018 TopQuadrant Inc.

SHACL Terminology – 4

- Validation Report
  - RDF graph with validation results
  - SHACL includes a vocabulary for describing results
- SHACL Core
  - Predefined constraint components
- SHACL SPARQL
  - SPARQL constraints and SPARQL-based constraint components
- SHACL Advanced Features
  - Functions, rules, extended targets

© Copyright 2018 TopQuadrant Inc.

TopQuadrant™

SHACL Targets, Nodes Shapes and Property Shapes

TopQuadrant™

Example Data Graph

```
@prefix example: <http://example.org/> .
@prefix td: <http://www.sandbox.com/training-data#> .
@prefix schema: <http://schema.org/> .

td:Alice a schema:Person .
td:Bob a schema:Student .
td:Jack a schema:Person .
td:Jill a schema:Teacher .
example:Bob a schema:Person .
schema:Student rdfs:subClassOf schema:Person .
td:Alice schema:givenName “Alice”;
          schema:familyName “Jones”;
          schema:knows example:Bob;
          schema:birthDate 1942-05-03;
          schema:worksFor td:TopQuadrant .
example:Bob schema:givenName “Bob”;
               schema:familyName “Brown”.
td:Jack schema:givenName “Jack”;
          schema:familyName “Smith” ;
          schema:familyName “Jones”.
td:Jill schema:givenName 1 .
```

© Copyright 2018 TopQuadrant Inc.

Node Shapes and Property Shapes - 1

- Node shapes are used to:
  - Specify constraints on the ”target” nodes
  - Group property shapes
- Property shapes are used to specify constraints on nodes that are reached by following some path from the target nodes

© Copyright 2018 TopQuadrant Inc.

Node Shapes and Property Shapes - 2

TopQuadrant™

```
schema:PersonShape a sh:NodeShape ;
   sh:targetClass schema:Person ;
   sh:pattern "^http://www.sandbox.com/training-data";
   sh:property [
           sh:path schema:givenName ;
           sh:minCount 1 ;
           sh:datatype xsd:string ;
   ] ;
   sh:property [
           sh:path schema:familyName ;
           sh:minCount 1 ;
           sh:maxCount 1 ;
           sh:datatype xsd:string ;
           sh:maxLength 20 ;
   ] .
```

Node Shape

Must specify at least one schema:givenName and it must be a string

Must specify only 1 schema:familyName which is 20 characters or less in length

© Copyright 2018 TopQuadrant Inc.

Targets-1

- Define what nodes will be validated against a shape
- Target statement determines scope of applicability of a shape
  - For example, all instances of schema:Person class

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

- We could also limit the shape to just a specific resource (e.g., Alice):

```
schema:PersonShape a sh:NodeShape ;
sh:targetNode td:Alice .
```

© Copyright 2018 TopQuadrant Inc.

Targets – 2

- Pre-built vocabulary for targets:
  - sh:targetNode – targets are the specified resources
  - sh:targetClass – targets are all resources that are members of a specified class (or one of its sub classes)
  - sh:targetSubjectsOf – targets are all subjects of triples with a given predicate
  - sh:targetObjectsOf – targets are all objects of triples with a given predicate

© Copyright 2018 TopQuadrant Inc.

Targets – 3

- Implicit class targets
  - If a node shape is also a class, it doesn't need an explicit sh:targetClass statement – integration point for existing ontologies
- SPARQL-based targets
  - Advanced feature

© Copyright 2018 TopQuadrant Inc.

Implicit Targets

TopQuadrant™

- When a class is also a node shape, it means that targets of a shape are class members

```
schema:Person a sh:NodeShape ;
              a owl:Class;
sh:pattern "^http://www.sandbox.com/training-data";
sh:property [
        sh:path schema:givenName ;
        sh:minCount 1 ;
        sh:datatype xsd:string ; ] ;
  sh:property [
        sh:path schema:familyName ;
        sh:minCount 1 ;
        sh:maxCount 1 ;
        sh:datatype xsd:string ;
        sh:maxLength 20 ; ] .
```

Applies to any member of the schema:Person class

© Copyright 2018 TopQuadrant Inc.

Targeting Specific Subjects or Objects

- RDF triple: subject / predicate / object
- Shapes can target all resources that are subjects or objects in triples with a specific predicate or property
-

```
schema:WorksForShape a sh:NodeShape ;
   sh:targetSubjectsOf schema:worksFor;
   sh:pattern “^http://www.sandbox.com/training-data” ;
   sh:property [
            sh:path schema:worksFor ;
            sh:minCount 1 ;
            sh:maxCount 1 ; ] .
```

All resources which have a schema:worksFor must have exactly 1

© Copyright 2018 TopQuadrant Inc.

Closed Shapes

```
schema:ClosedPersonShape a sh:NodeShape ;
   sh:targetClass schema:Person ;
   sh:closed true;
   sh:ignoredProperties ( rdf:type ) ;
   sh:property [
            sh:path schema:givenName ;
            sh:minCount 1 ;
            sh:datatype xsd:string ; ] ;
   sh:property [
            sh:path schema:familyName ;
            sh:minCount 1 ;
            sh:maxCount 1 ;
            sh:datatype xsd:string ;
            sh:maxLength 20 ; ] .
```

- By default, if we do not say anything about a property, then it can have any value
- But, if there is sh:closed true, then properties that are not explicitly mentioned (except the "ignored properties") are not allowed

© Copyright 2018 TopQuadrant Inc.

TopQuadrant™

Property Shapes – 2

```
schema:PersonShape a sh:NodeShape ;
sh:targetClass schema:Person ;
sh:property [  #b1
        sh:path schema:givenName ;
        sh:minCount 1 ;
        sh:datatype xsd:string ; ] ;
sh:property [ #b2
        sh:path schema:familyName ;
        sh:minCount 1 ;
        sh:maxCount 1 ;
        sh:datatype xsd:string ;
        sh:maxLength 20 ;] .
```

Node Shape

Property Shape

© Copyright 2018 TopQuadrant Inc.

Property Shapes – 3

- Here we use URIs for the property shapes
- URIs can be addressed/extended from other graphs

Node Shape

Property Shape

schema:PersonShape2 a sh:NodeShape ;
sh:targetClass schema:Person ;
sh:property td:Person-givenName ;
sh:property td:Person-familyName .

schema:Person-givenName a sh:PropertyShape ;
sh:path schema:givenName ;
sh:minCount 1 ;
sh:datatype xsd:string .

schema:Person-familyName a sh:PropertyShape ;
sh:path schema:familyName ;
sh:minCount 1 ;
sh:maxCount 1 ;
sh:datatype xsd:string ;
sh:maxLength 20 .

© Copyright 2018 TopQuadrant Inc.

Summary of SHACL Core Constraint Components

Constraint Components

- Value Type
- Value Range
- Cardinality
- String Values
- Property Pairs
- Logical Expressions
- Shape
- Qualified Value Shapes
- Miscellaneous

© Copyright 2018 TopQuadrant Inc.

Validation Report

Validation Report Vocabulary

- sh:conforms – true if no validation results were produced
- sh:result/sh:ValidationResult
- sh:focusNode – identifies a node that produced the results i.e., a node that has problems
- sh:value – identifies what value is incorrect
- sh:resultPath – identifies how the incorrect value is connected to the focus node
- sh:sourceShape – what shape has been violated
- sh:sourceConstraintComponent – what constraint component has been violated
- sh:detail – further details
- sh:resultMessage – tools may use this to return helpful messages to the users
- sh:resultSeverity

© Copyright 2018 TopQuadrant Inc.

Path Expressions

© Copyright 2018 TopQuadrant Inc.

Path Expressions - 1

- The value of sh:path can be a single predicate - or it can be a property path
- SHACL supports a subset of SPARQL property paths. Specifically:
  - PredicatePath - simply the property
  - InversePath – using inverse. We created inverse path for “children” in exercise 2
  - SequencePath – a sequential list of properties that used as a path
  - AlternativePath – provides alternative paths. For example, rdfs:label or skos:prefLabel must exist
  - ZeroOrMorePath, OneOrMorePath and ZeroOrOnePath – using *, + and ? operators in SPARQL

© Copyright 2018 TopQuadrant Inc.

Different results demonstrate SHACL's use of rdf:type inferencing

- Two ways to state "anyone a person knows must be a person":
  - One uses a property path of two predicates and sh:hasValue constraint
  - Another. uses a single predicate path and sh:class constraint

```
schema:PersonKnows a sh:NodeShape ;
                    sh:targetClass schema:Person ;
sh:property [
          sh:path ( schema:knows rdf:type ) ;
          sh:hasValue schema:Person ; ] .
```

```
schema:PersonKnows a sh:NodeShape ;
                    sh:targetClass schema:Person ;
sh:property [
          sh:path schema:knows;
          sh:class schema:Person ; ] .
```

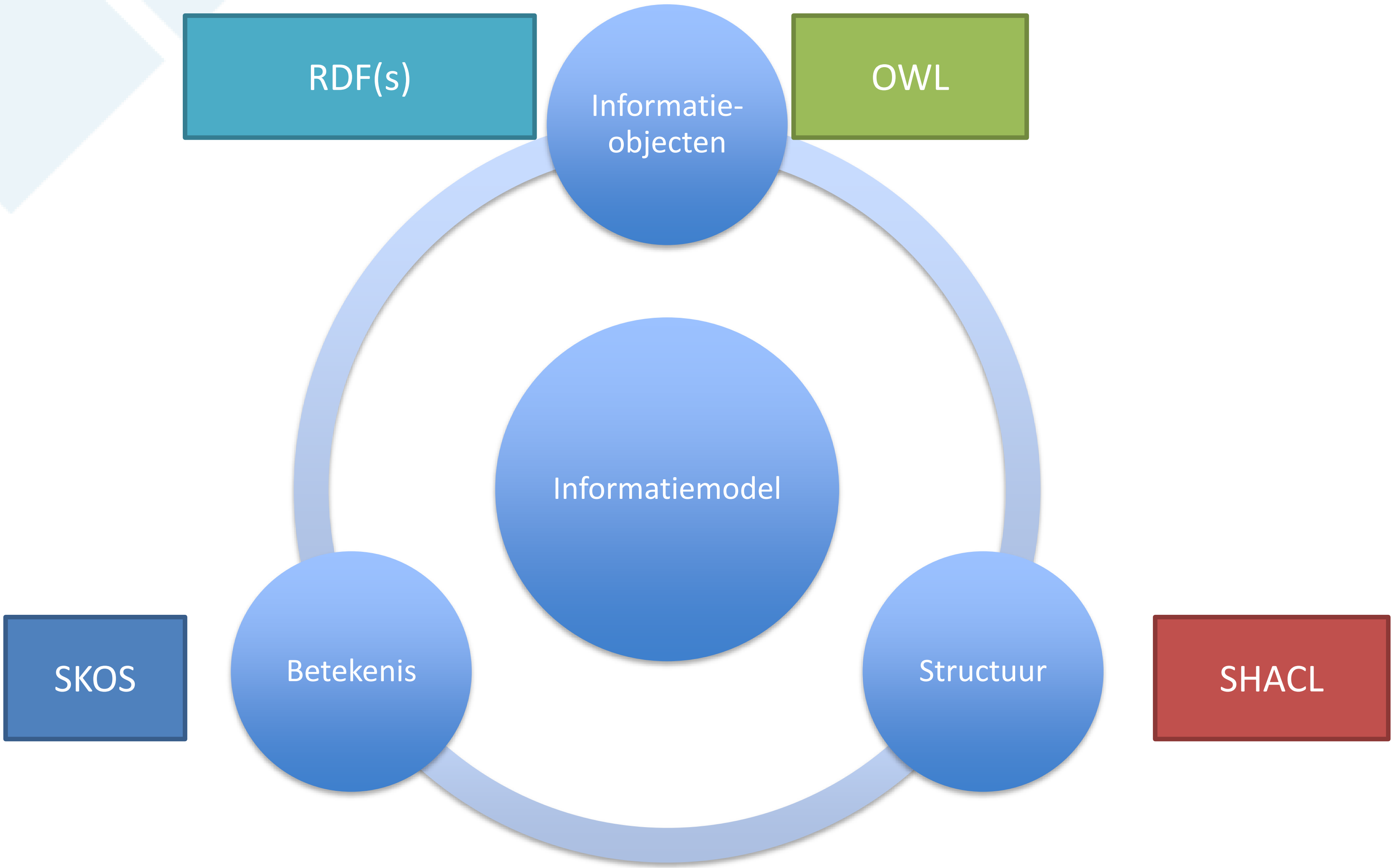
not valid

valid

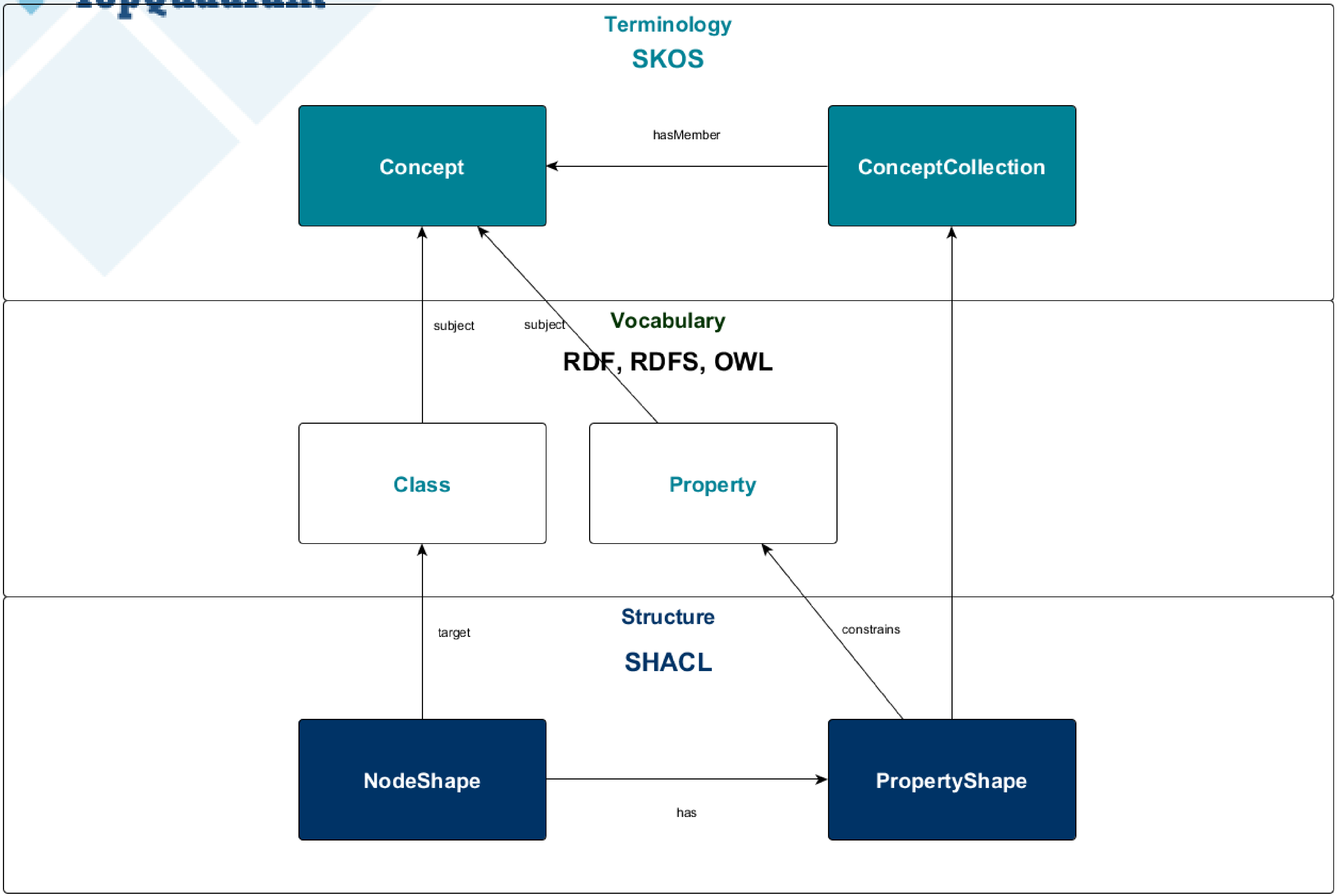
```
td:Alice a schema:Person;
         schema:knows example:Bob .
example:Bob a td:Student.
td:Student rdfs:subClassOf schema:Person .
```

© Copyright 2018 TopQuadrant Inc.

De rol van SHACL



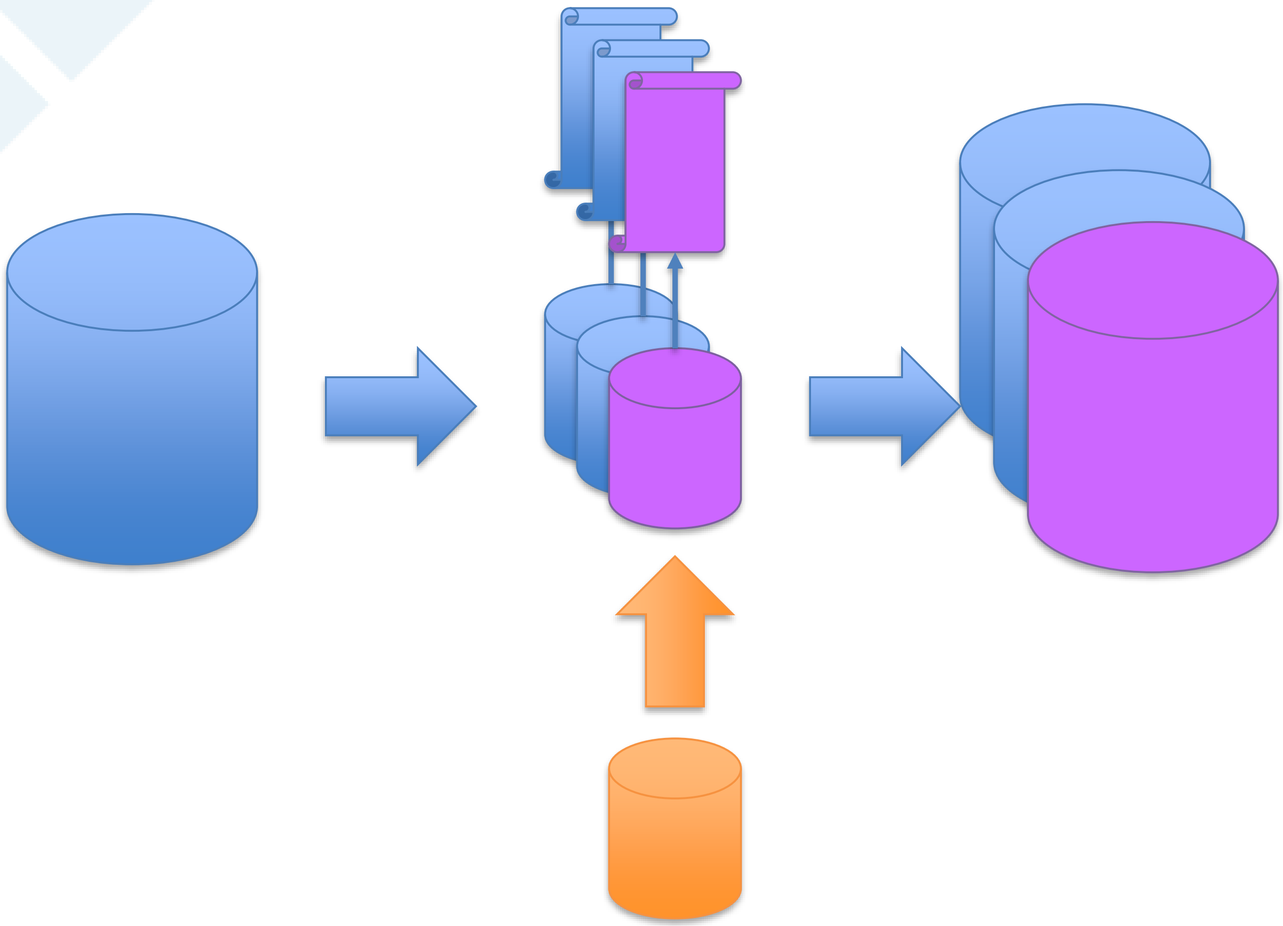
© Copyright 2018 TopQuadrant Inc.



Pano Maria, Jesse Bakker (SEMANT!CS2017)

© Copyright 2018 TopQuadrant Inc.

SHACL als schema



© Copyright 2018 TopQuadrant Inc.

SPARQL Constraint Component

© Copyright 2018 TopQuadrant Inc.

SHACL SPARQL

- sh:SPARQLConstraintComponent
  - a constraint component that can be used to express restrictions on data based on a SPARQL SELECT query

© Copyright 2018 TopQuadrant Inc.

SPARQL Constraint Component Example

```
ex:LanguageExampleShape a sh:NodeShape ;
sh:targetClass ex:Country ;
sh:sparql [
		a sh:SPARQLConstraint ; # This triple is optional
		sh:message "Values are literals with German language tag." ;
		sh:prefixes ex: <http://example.com> ;
		sh:select """ SELECT $this (ex:germanLabel AS ?path) ?value
				WHERE {
						$this ex:germanLabel ?value .
						FILTER (!isLiteral(?value) || !langMatches(lang(?value), "de"))
						} """ ;
		] .
```

The target of this shape are all SHACL instances of ex:Country.

For those nodes (represented by the variable **this**), the SPARQL query walks through the values of **ex:germanLabel.** For any value that is not a literals or has a language tag that is not “de”, there is a validation result.

© Copyright 2018 TopQuadrant Inc.

TopQuadrant™

Other Types of Validators

- SPARQL queries is one option for validation
- JavaScript is another built-in option
- Validators in other languages could be developed

© Copyright 2018 TopQuadrant Inc.

Inferencing with SHACL

TopQuadrant™

SHACL Inference Mechanisms

- Triple Rules
  - Specify inferred statement as a triple
- SPARQL Rules
  - Specify inferred statement as a SPARQL CONSTRUCT query
- Property Values Extension
  - Very similar to Triple Rules with some additional “syntactic sugar”
  - Specify inferred values as part of a property shape
  - Support dynamic inferencing

© Copyright 2018 TopQuadrant Inc.

Triple Rule Example: Inferring a new Type

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

What are the focus nodes of this shape?

sh:this means every focus node of the shape that meets conditions (if any)

Inferred statement

© Copyright 2018 TopQuadrant Inc.

SPARQL Rule Example – calculating area

```
ex:Rectangle a rdfs:Class, sh:NodeShape ;
rdfs:label "Rectangle" ;
sh:property  [ sh:path ex:height ;
               sh:datatype xsd:integer ;
               sh:maxCount 1 ;
               sh:minCount 1 ;
               sh:name "height" ; ] ;
sh:property [ sh:path ex:width ;
               sh:datatype xsd:integer ;
               sh:maxCount 1 ;
               sh:minCount 1 ;
               sh:name "width" ; ] ;
sh:rule [ a sh:SPARQLRule ;
               sh:construct """ CONSTRUCT {$this ex:area ?area. }
                                WHERE { $this ex:width ?width .
                                        $this ex:height ?height .
                                        BIND (?width * ?height AS ?area) . } """ ;
               sh:prefixes
                 [ sh:declare
                      [ sh:prefix "ex" ; sh:namespace "http://example.com/ns#"^^xsd:anyURI ; ] ; ] ;
                  [ sh:declare
                      [ sh:prefix "rdf" ; sh:namespace "http://www.w3.org/1999/02/22-rdf-syntax-ns#"^^xsd:anyURI ; ] ; ] ;
               sh:condition ex:Rectangle ;
          ] .
```

sh:declare can also be stated at the graph level. Then, refer to the <base URI of the graph> in the prefixes statement

Not needed, included only to show that a condition can be specified

© Copyright 2018 TopQuadrant Inc.

Area Calculation Using a Triple Rule

```
ex:RectangleRulesShape a sh:NodeShape ;
sh:targetClass ex:Rectangle ;
sh:rule [
        a sh:TripleRule ;
        sh:subject sh:this ;
        sh:predicate ex:area ; # Computes the values of the ex:area property at the focus nodes
        sh:object [
                sparlq:multiply ( [ sh:path ex:width ] [ sh:path ex:height ] ) ;
                ] ;
        sh:condition ex:RectangleShape ; # Rule only applies to Rectangles that conform to
ex:RectangleShape. In other words have exactly one width and height and the values of these are
integers.
        ] .

ex:RectangleShape a sh:NodeShape ;
sh:targetClass ex:Rectangle ;
sh:property [ sh:path ex:width ; sh:datatype xsd:integer ; sh:minCount 1 ; sh:maxCount 1 ; ] ;
sh:property [ sh:path ex:height ; sh:datatype xsd:integer ; sh:minCount 1 ; sh:maxCount 1 ; ] .
```

As an example, we are doing this slightly differently – with an explicit target. Plus we have separated the shape with a rule from the shape that defines properties

Uses a SHACL function. Users can define functions themselves. A useful collection of functions is available in the sparql: namespace at http://datashapes.org/sparql

TopQuadrant™

Triple Rules vs SPARQL Rules

- Triple Rules are declarative, making it easier for an engine to understand and thus optimize its use cases
- Triple Rules can produce multiple triples for the same subject/predicate
- Recommendation is, when possible, to use Triple Rules rather than SPARQL Rules.
- The downside: when one needs to infer values for more than one property (sh:predicate), it will require a rule per property.

© Copyright 2018 TopQuadrant Inc.

Example of using Property Values: Inferring Children using Parents

```
schema:Person
sh:property [
        sh:path schema:children ;
        sh:class schema:Person ;
        sh:values [
                sh:path [
                        sh:inversePath schema:parent ;
                        ]
                ]
        ] .
```

- Values of schema:children will be inferred using inverse of the values of schema:parent
- If we did a triple rule, we would have said the following at the NodeShape:

  sh:rule [a sh:TripleRule ;

  sh:subject sh:this ;

  sh:predicate schema:children;

  sh:object [sh:path [sh:inversePath schema:parent;] ;] ;] .
- Here, we are only specifying the object, so this is a less verbose option

© Copyright 2018 TopQuadrant Inc.

Default Values

- sh:defaultValue – will make the same inferences as sh:values, but only if the property has no values
- Population ”by default”
- If a values is added, inference does not happen – default is overridden by the value

© Copyright 2018 TopQuadrant Inc.