Introduction to SHACL

Robin Keskisärkkä
robin.keskisarkka@liu.se

Takeaways

- How SHACL fits into the Semantic Web technology stack
- Basic understanding of the core features in SHACL
- Hands-on experience in using SHACL to validate RDF graphs

About RDF

- The standard language for describing data on the Semantic Web

- Data is represented as directed graphs

- Allows for integration of different data sources

- Many serialization formats
  - XML, JSON-LD, NT, N-Triples, Turtle, …

What about a schema language for RDF?

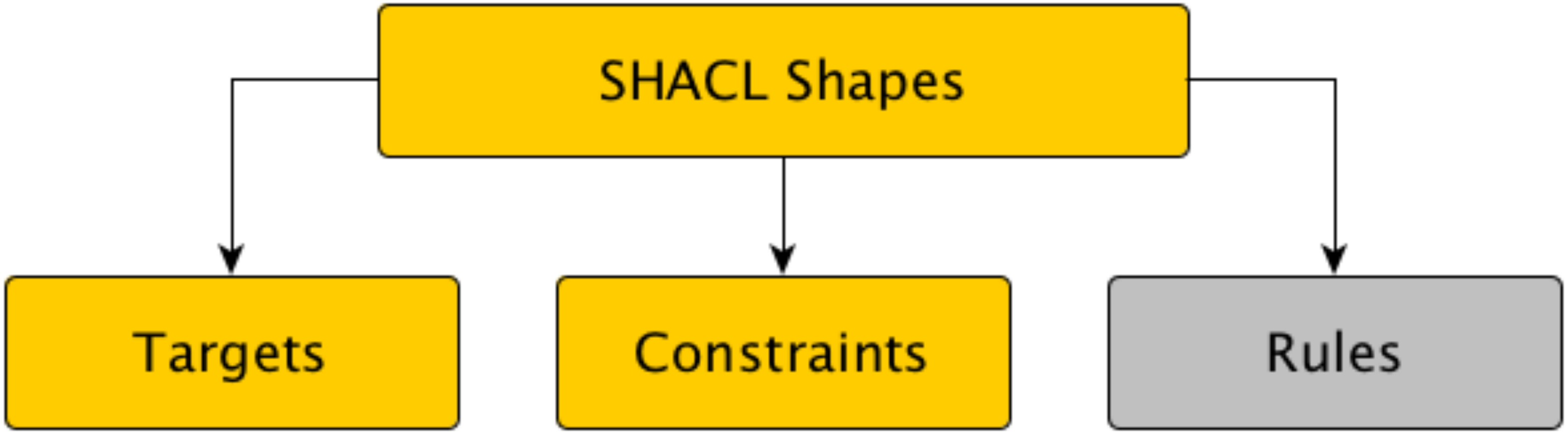
- RDF Schema
  - A bit of a misnomer
  - Should really be something like “RDF Vocabulary Definition Language”
  - Limited expressivity
- OWL
  - Targets logic modeling and inferencing, not validation
  - Open-world assumption
  - No unique name assumption
- Alternative approaches
  - OWL under the closed-world assumption
- Semi-official specifications (W3C submissions)
  - SPIN, Resource Shapes, ShEx, …

SHACL

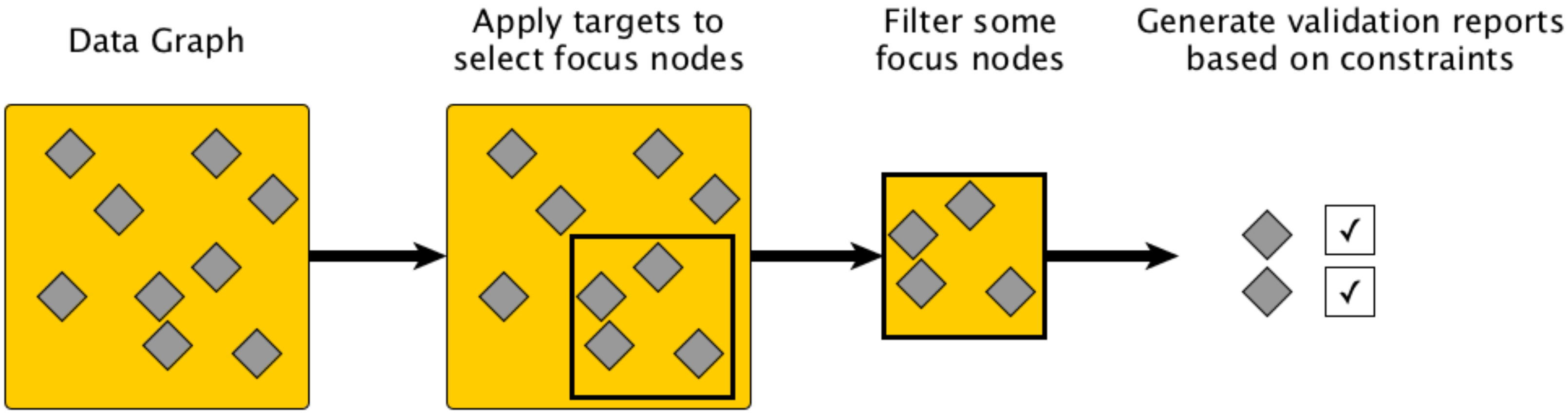
What is SHACL?

- **Sha**pes **C**onstraint **L**anguage
- W3C recommendation since July 2017
- Allows RDF data to be validated against *shapes*
  - Data graph
  - Shapes graph
- … and more (user-interface generation, inference, SPARQL extensions, etc.)

Overview



Validation process


Data Graph
Apply targets to select focus nodes
Filter some focus nodes
Generate validation reports based on constraints
✓
✓

Turtle

- Shorthand expressions
  - Semi-colon denotes repeated use of preceding subject
  - Comma denotes repeated use of the preceding subject and property
  - Parentheses denote lists or collections
  - Brackets denote blank nodes (anonymous resources)

SHACL Example

```
ex:PonyShape
    a sh:NodeShape ;
    sh:targetClass ex:RainbowPony ;
    sh:property ex:PonyPropertyShape .

ex:PonyPropertyShape
    a sh:PropertyShape ;
    sh:path ex:color ;
    sh:in ( ex:Pink ex:Purple ) .
```

SHACL Example

```
ex:PonyShape
    a sh:NodeShape ;
    sh:targetClass ex:RainbowPony ;
    sh:property [
       sh:path ex:color ;
       sh:in ( ex:Pink ex:Purple )
    ] .
```

Targets

- Targets are applied to Node Shapes and specify the nodes that are to be validated
- Targets can be specified in several ways
  - targetClass – All instances of a class
  - targetNode – Specific nodes
  - targetObjectsOf – All object of a specific property
  - targetSubjectsOf – All subjects of a specific property

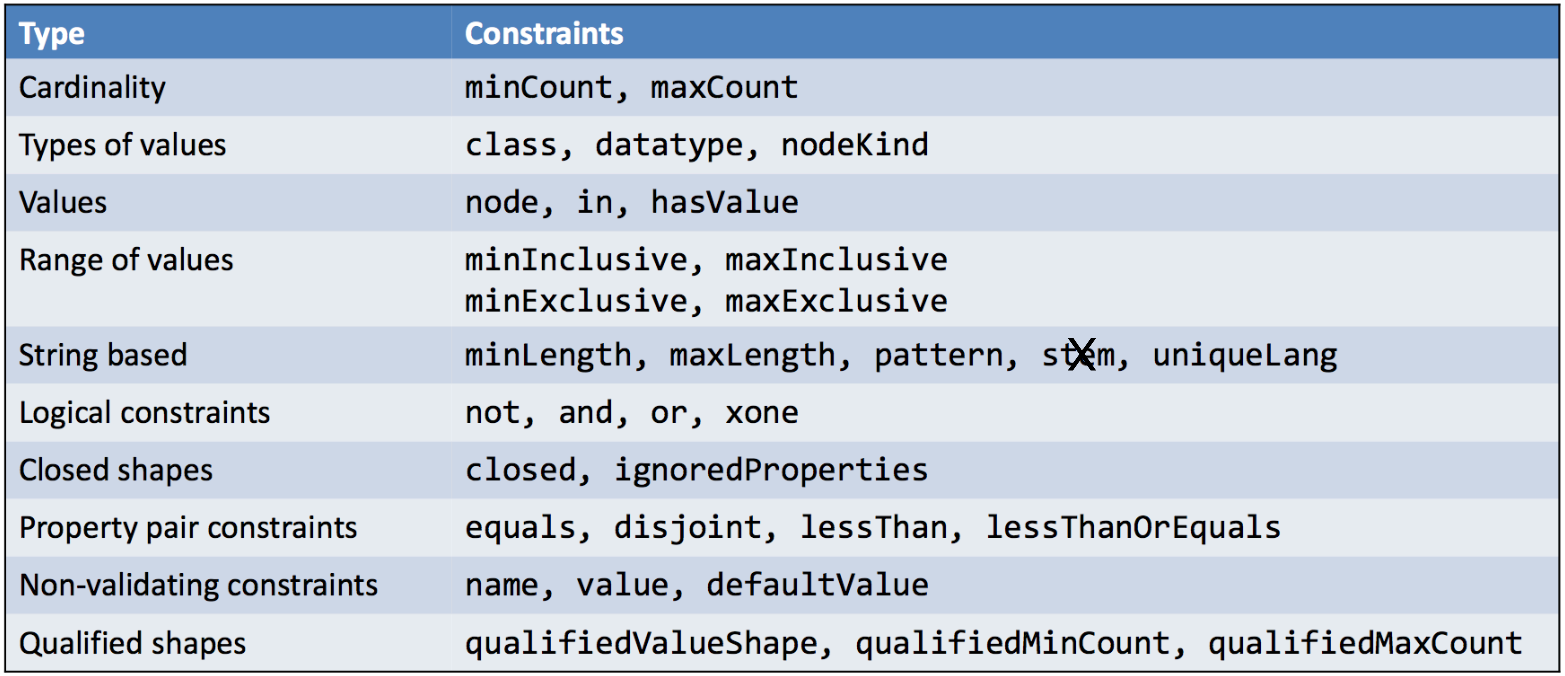
Targets

```
ex:PonyShape1
    a sh:NodeShape ;
    sh:targetClass ex:Pony .

ex:PonyShape2
    a sh:NodeShape ;
    sh:targetNode ex:Pinkie_Pie, ex:Rarity .

ex:PonyShape3
    a sh:NodeShape ;
    sh:targetSubjectsOf ex:hasFriend .
```

SHACL Core Constraints



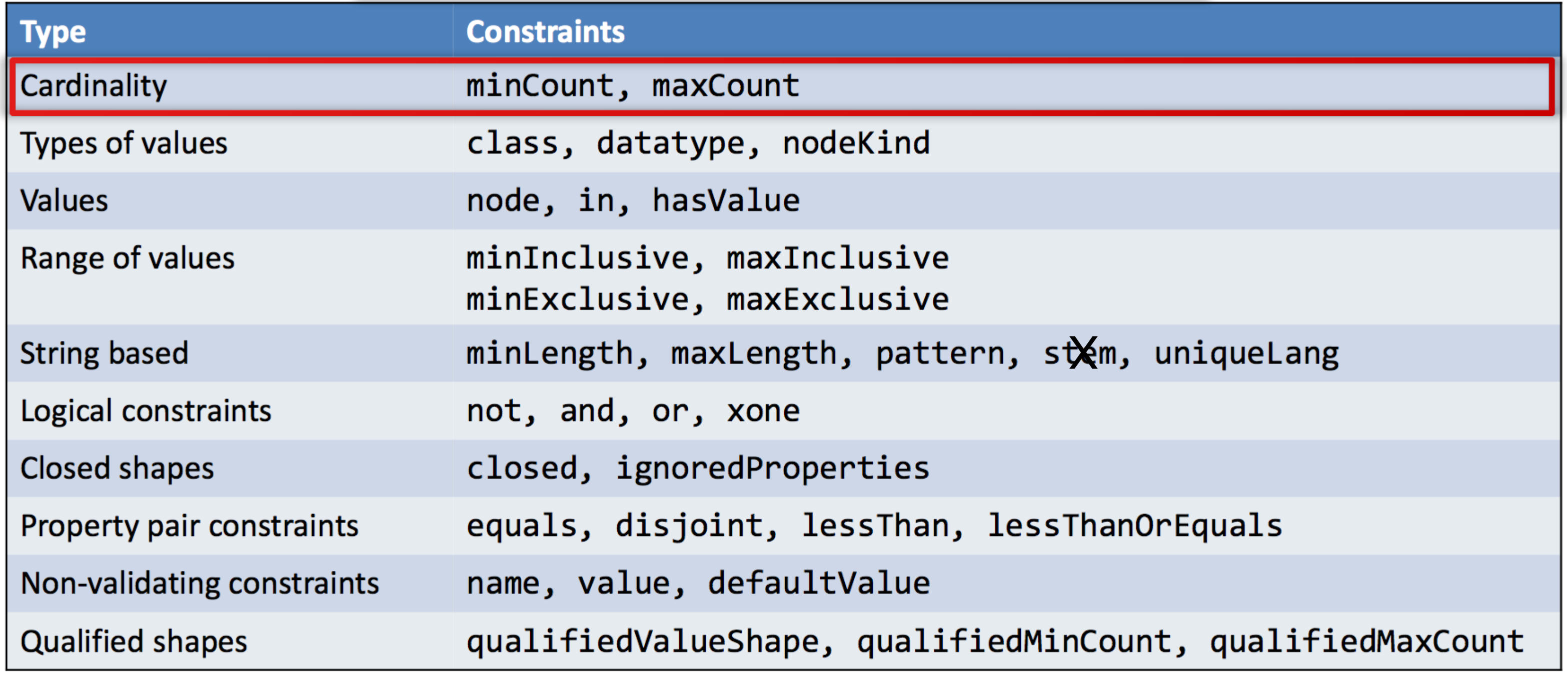
Type	Constraints
Cardinality	`minCount, maxCount`
Types of values	`class, datatype, nodeKind`
Values	`node, in, hasValue`

Range of values	`minInclusive, maxInclusive`
	`minExclusive, maxExclusive`

String based	`minLength, maxLength, pattern,` ~~`stem`~~`, uniqueLang`
Logical constraints	`not, and, or, xone`
Closed shapes	`closed, ignoredProperties`
Property pair constraints	`equals, disjoint, lessThan, lessThanOrEquals`
Non-validating constraints	`name, value, defaultValue`
Qualified shapes	`qualifiedValueShape, qualifiedMinCount, qualifiedMaxCount`

SHACL Core Constraints

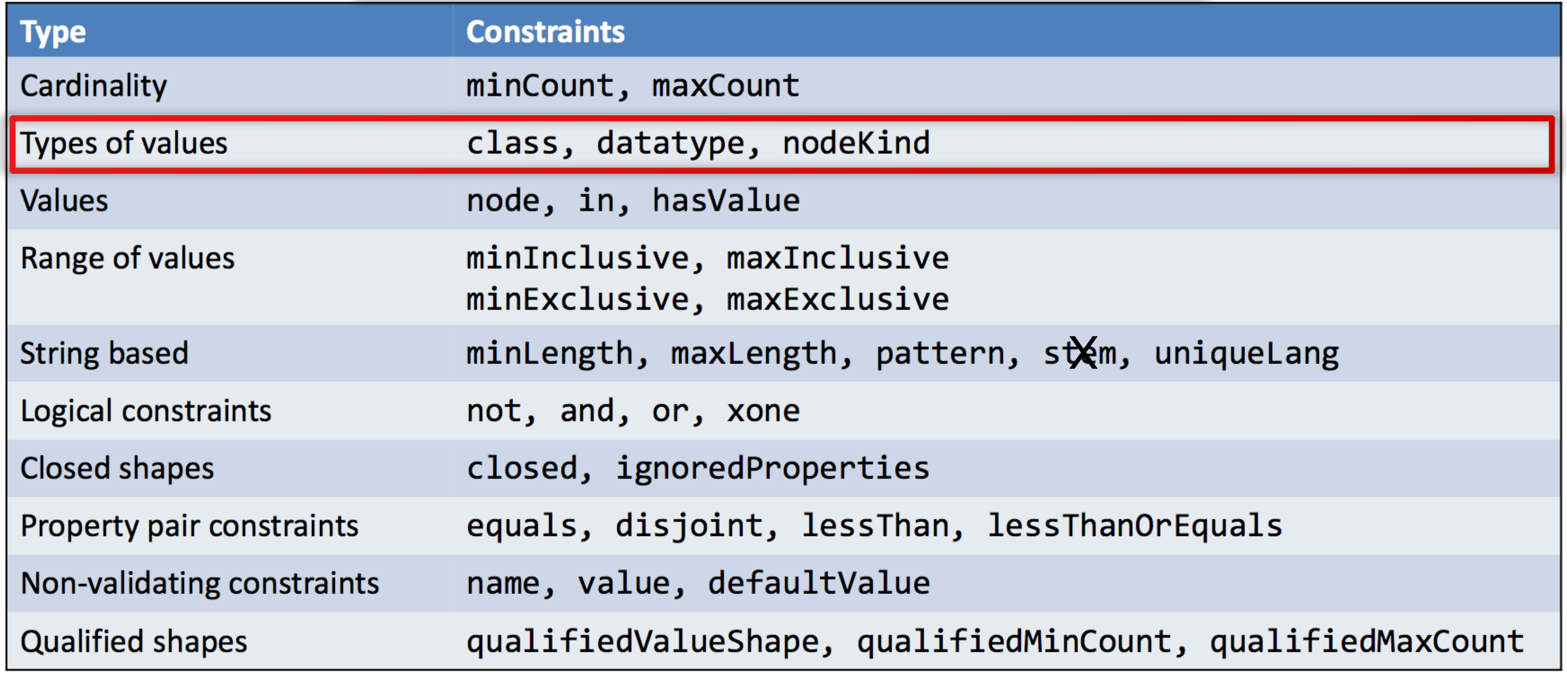
Constraint on the number of value nodes (i.e., individual focus nodes) that satisfy the condition.



Type	Constraints
Cardinality	minCount, maxCount
Types of values	class, datatype, nodeKind
Values	node, in, hasValue
Range of values	minInclusive, maxInclusive minExclusive, maxExclusive
String based	minLength, maxLength, pattern, ~~stem~~, uniqueLang
Logical constraints	not, and, or, xone
Closed shapes	closed, ignoredProperties
Property pair constraints	equals, disjoint, lessThan, lessThanOrEquals
Non-validating constraints	name, value, defaultValue
Qualified shapes	qualifiedValueShape, qualifiedMinCount, qualifiedMaxCount

SHACL Core Constraints

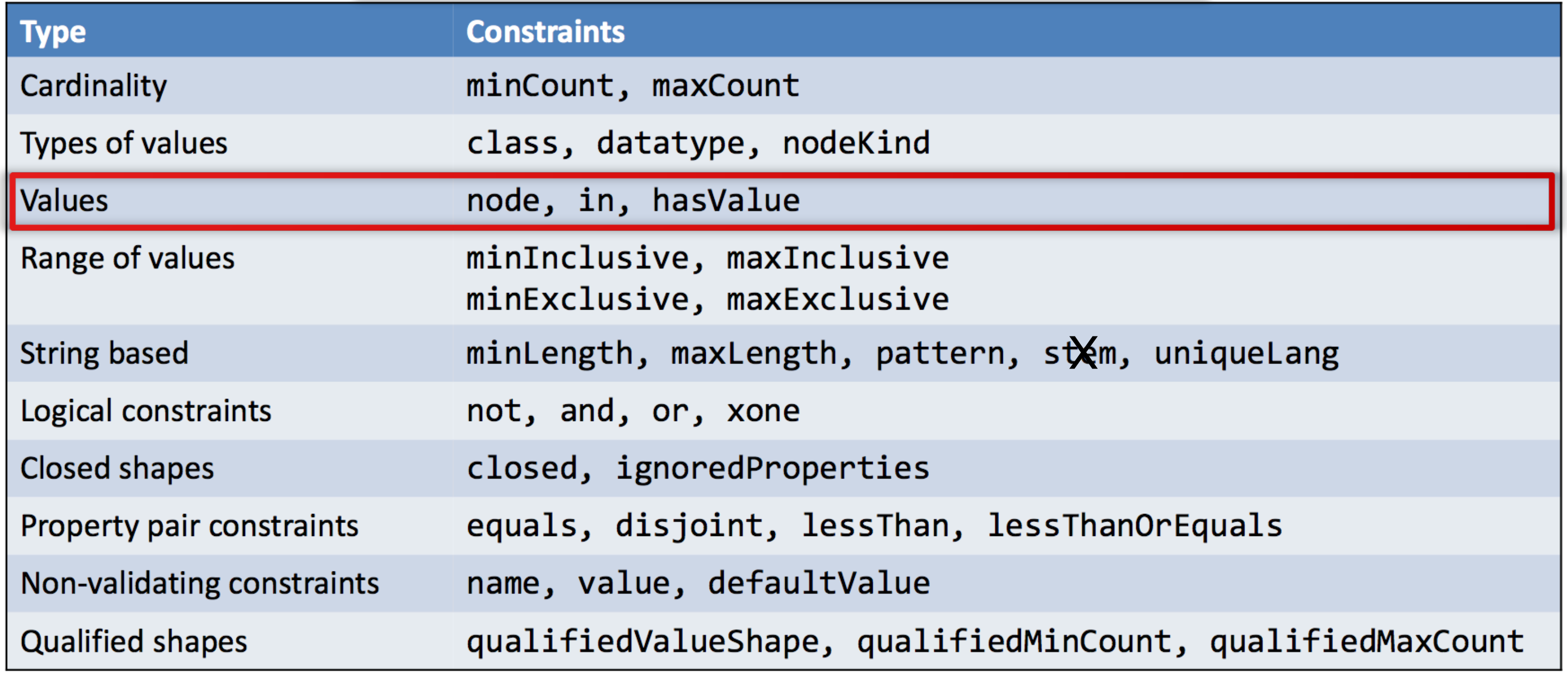
Constraints that restrict the type of value nodes.



Type	Constraints
Cardinality	minCount, maxCount
Types of values	class, datatype, nodeKind
Values	node, in, hasValue
Range of values	minInclusive, maxInclusive minExclusive, maxExclusive
String based	minLength, maxLength, pattern, ~~stem~~, uniqueLang
Logical constraints	not, and, or, xone
Closed shapes	closed, ignoredProperties
Property pair constraints	equals, disjoint, lessThan, lessThanOrEquals
Non-validating constraints	name, value, defaultValue
Qualified shapes	qualifiedValueShape, qualifiedMinCount, qualifiedMaxCount

SHACL Core Constraints

Specifies a value to which a value node must conform.



Type	Constraints
Cardinality	minCount, maxCount
Types of values	class, datatype, nodeKind
Values	node, in, hasValue
Range of values	minInclusive, maxInclusive minExclusive, maxExclusive
String based	minLength, maxLength, pattern, ~~stem~~, uniqueLang
Logical constraints	not, and, or, xone
Closed shapes	closed, ignoredProperties
Property pair constraints	equals, disjoint, lessThan, lessThanOrEquals
Non-validating constraints	name, value, defaultValue
Qualified shapes	qualifiedValueShape, qualifiedMinCount, qualifiedMaxCount

SHACL Core Constraints

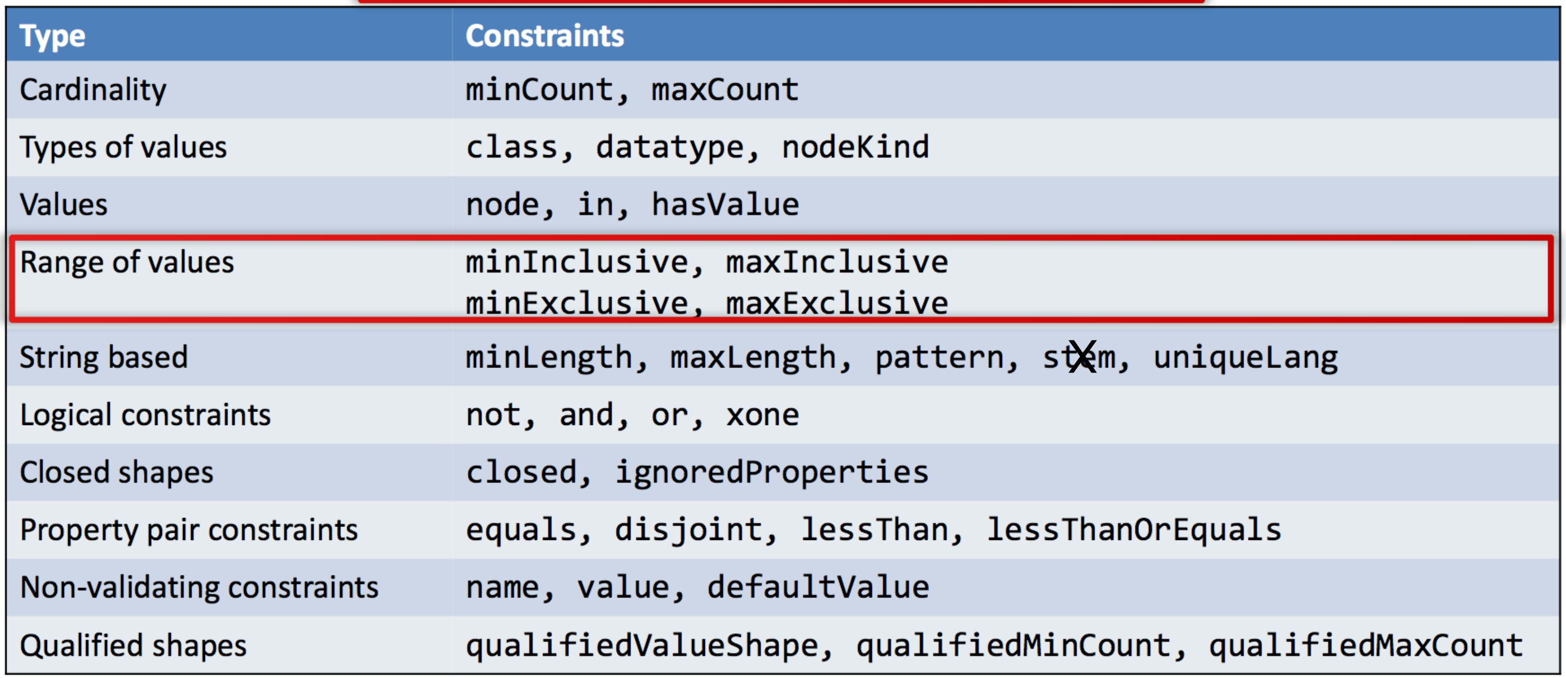
Specifies a value to which a value node must conform.

Type	Constraints
Cardinality	minCount, maxCount
Types of values	class, datatype, nodeKind
Values	node, in, hasValue
Range of values	minInclusive, maxInclusive minExclusive, maxExclusive
String based	minLength, maxLength, pattern, stem, uniqueLang
Logical constraints	[illegible]
Closed shapes	closed, ignoredProperties
Property pair constraints	equals, disjoint, lessThan, lessThanOrEquals
Non-validating constraints	name, value, defaultValue
Qualified shapes	qualifiedValueShape, qualifiedMinCount, qualifiedMaxCount

Warning: Not the same!

SHACL Core Constraints

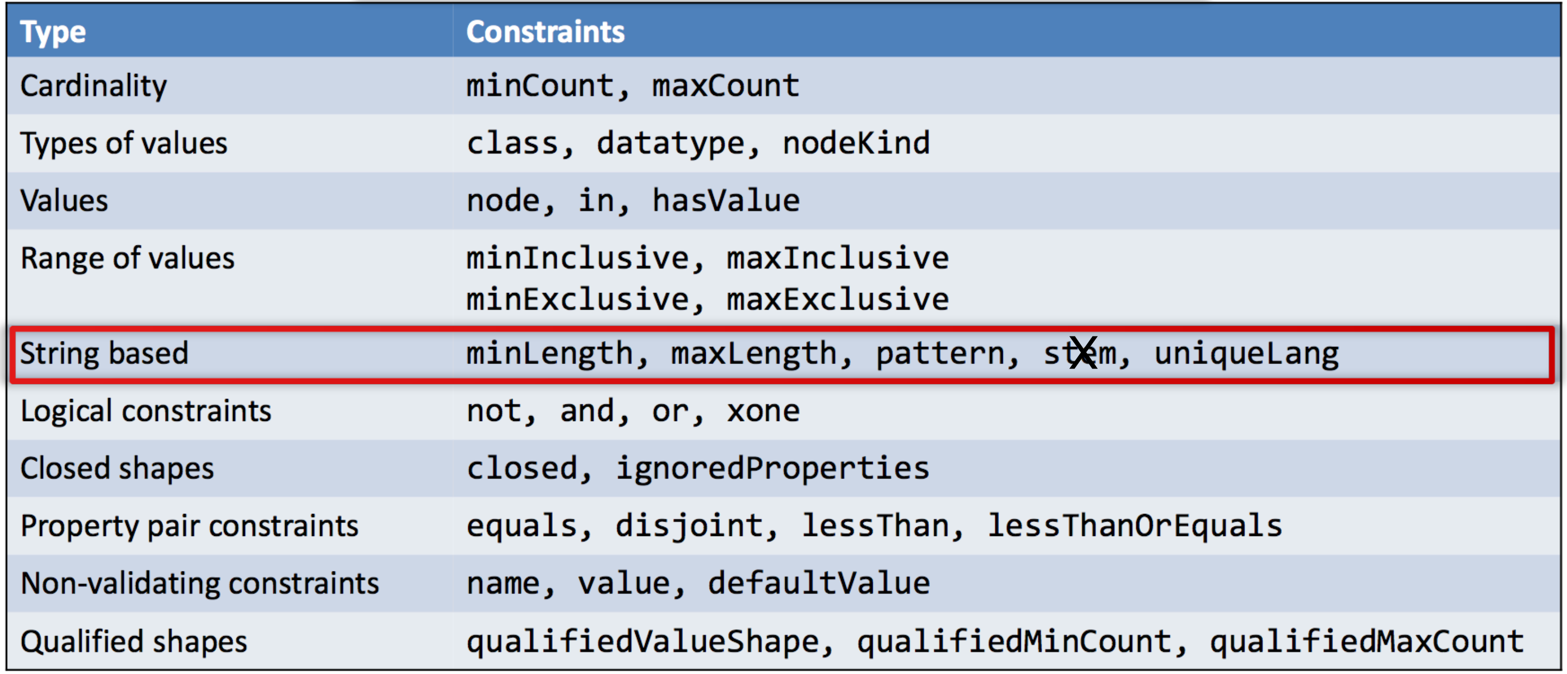
Constraints on value nodes comparable using <, <=, > and >=.



Type	Constraints
Cardinality	minCount, maxCount
Types of values	class, datatype, nodeKind
Values	node, in, hasValue
Range of values	minInclusive, maxInclusive minExclusive, maxExclusive
String based	minLength, maxLength, pattern, ~~stem~~, uniqueLang
Logical constraints	not, and, or, xone
Closed shapes	closed, ignoredProperties
Property pair constraints	equals, disjoint, lessThan, lessThanOrEquals
Non-validating constraints	name, value, defaultValue
Qualified shapes	qualifiedValueShape, qualifiedMinCount, qualifiedMaxCount

SHACL Core Constraints

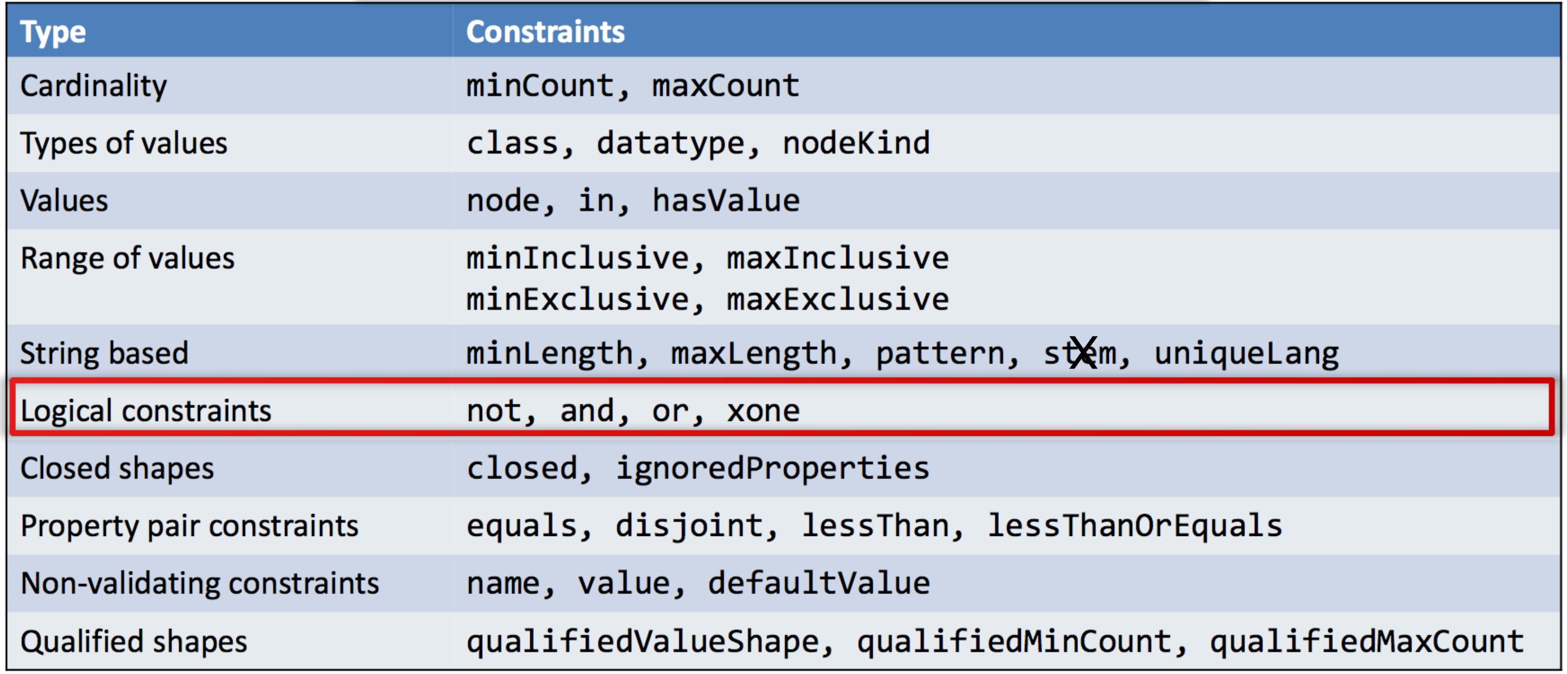
Constraints on the string representation of value nodes.



Type	Constraints
Cardinality	minCount, maxCount
Types of values	class, datatype, nodeKind
Values	node, in, hasValue
Range of values	minInclusive, maxInclusive minExclusive, maxExclusive
String based	minLength, maxLength, pattern, ~~stem~~, uniqueLang
Logical constraints	not, and, or, xone
Closed shapes	closed, ignoredProperties
Property pair constraints	equals, disjoint, lessThan, lessThanOrEquals
Non-validating constraints	name, value, defaultValue
Qualified shapes	qualifiedValueShape, qualifiedMinCount, qualifiedMaxCount

SHACL Core Constraints

Logical operators and, or, not, and exclusive or. These expect shape constraints.



Type	Constraints
Cardinality	minCount, maxCount
Types of values	class, datatype, nodeKind
Values	node, in, hasValue
Range of values	minInclusive, maxInclusive minExclusive, maxExclusive
String based	minLength, maxLength, pattern, ~~stem~~, uniqueLang
Logical constraints	not, and, or, xone
Closed shapes	closed, ignoredProperties
Property pair constraints	equals, disjoint, lessThan, lessThanOrEquals
Non-validating constraints	name, value, defaultValue
Qualified shapes	qualifiedValueShape, qualifiedMinCount, qualifiedMaxCount

SHACL Core Constraints

```
ex:Twilight_Sparkle
    a ex:MagicPony ;
    ex:knows ex:Magic .

ex:Rarity
    a ex:MagicPony ;
    ex:hasHorn true .
```

```
ex:MagicPonyShape
    a sh:NodeShape ;
    sh:targetClass ex:MagicPony ;
    sh:or (
        [ sh:path ex:knows ;
          sh:hasValue ex:Magic ]
        [ sh:path ex:hasHorn ;
          sh:hasValue true ]
    ) .
```

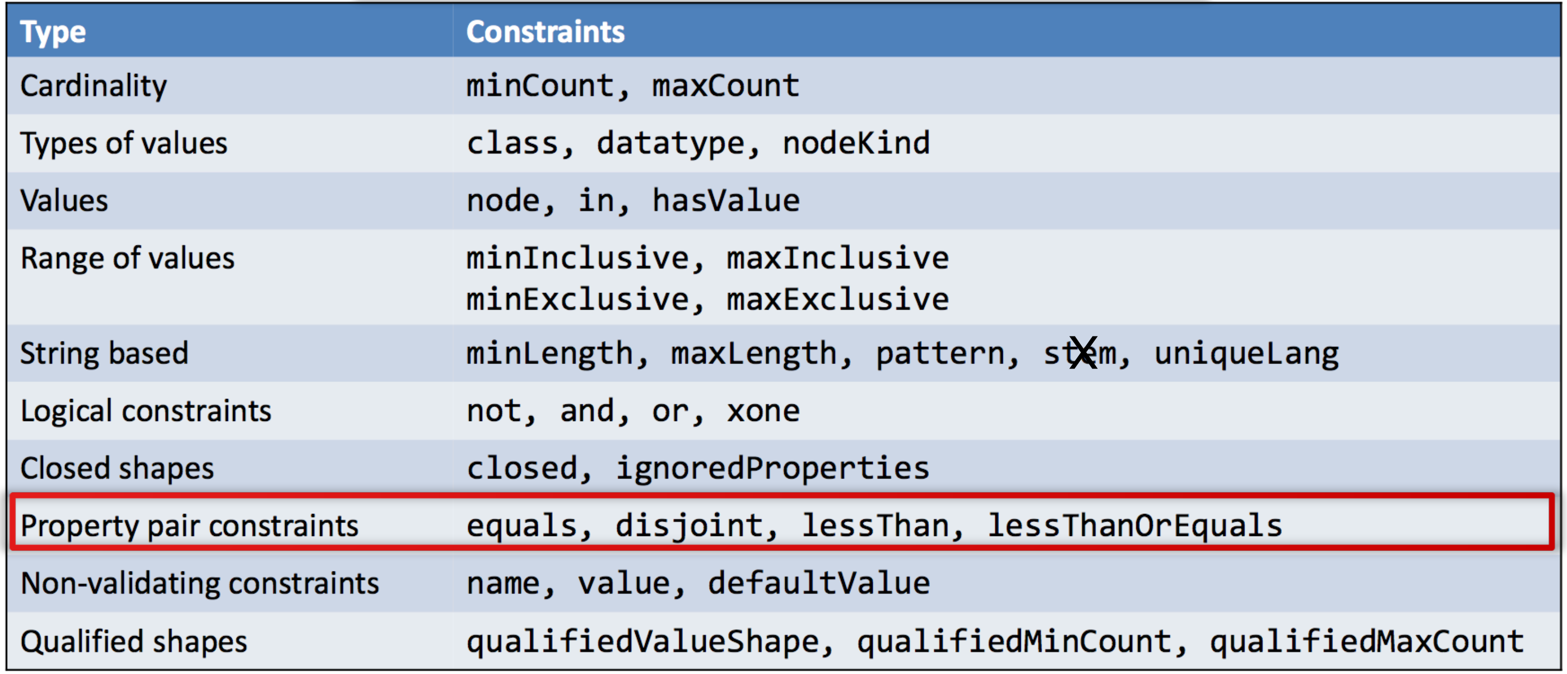
SHACL Core Constraints

Limit the flexibility of RDF data model.
closed: Only properties listed in the shape are allowed.
ignoredProperties: Permitted in addition to listed ones.

Type	Constraints
Cardinality	minCount, maxCount
Types of values	class, datatype, nodeKind
Values	node, in, hasValue
Range of values	minInclusive, maxInclusive minExclusive, maxExclusive
String based	minLength, maxLength, pattern, ~~stem~~, uniqueLang
Logical constraints	not, and, or, xone
Closed shapes	closed, ignoredProperties
Property pair constraints	equals, disjoint, lessThan, lessThanOrEquals
Non-validating constraints	name, value, defaultValue
Qualified shapes	qualifiedValueShape, qualifiedMinCount, qualifiedMaxCount

SHACL Core Constraints

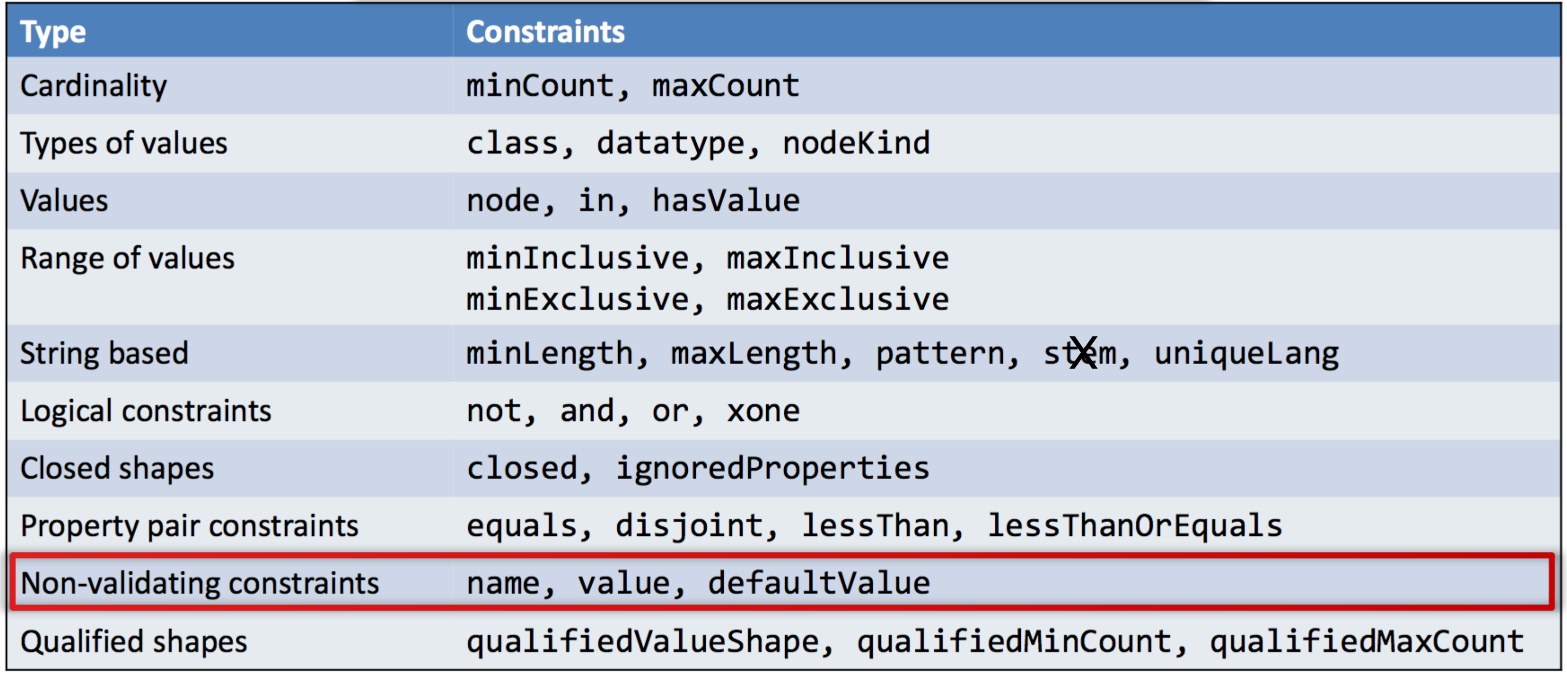
Conditions on value nodes in relation to other properties. Can only be used by property shapes.



Type	Constraints
Cardinality	minCount, maxCount
Types of values	class, datatype, nodeKind
Values	node, in, hasValue
Range of values	minInclusive, maxInclusive minExclusive, maxExclusive
String based	minLength, maxLength, pattern, ~~stem~~, uniqueLang
Logical constraints	not, and, or, xone
Closed shapes	closed, ignoredProperties
Property pair constraints	equals, disjoint, lessThan, lessThanOrEquals
Non-validating constraints	name, value, defaultValue
Qualified shapes	qualifiedValueShape, qualifiedMinCount, qualifiedMaxCount

SHACL Core Constraints

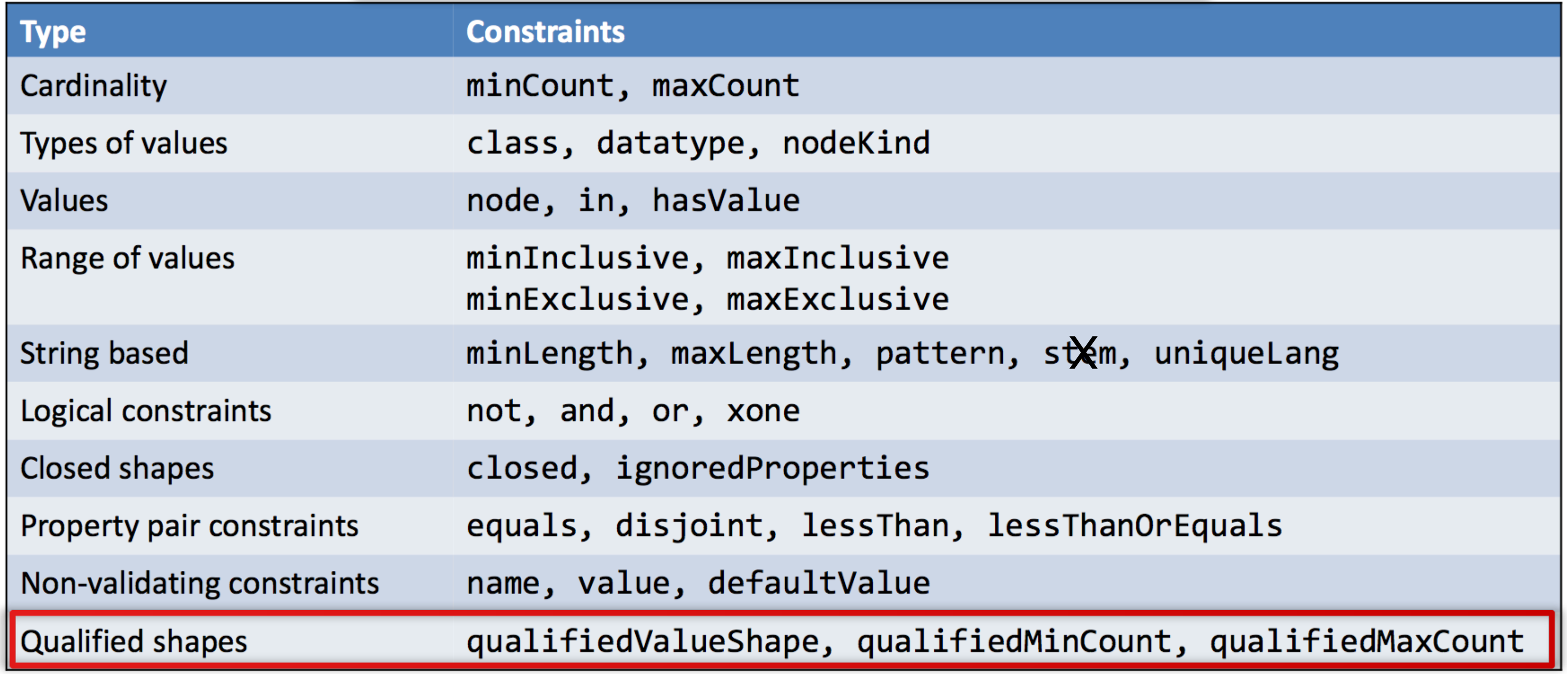
Non-validating properties are optional and not validated by default.



Type	Constraints
Cardinality	minCount, maxCount
Types of values	class, datatype, nodeKind
Values	node, in, hasValue
Range of values	minInclusive, maxInclusive minExclusive, maxExclusive
String based	minLength, maxLength, pattern, ~~stem~~, uniqueLang
Logical constraints	not, and, or, xone
Closed shapes	closed, ignoredProperties
Property pair constraints	equals, disjoint, lessThan, lessThanOrEquals
Non-validating constraints	name, value, defaultValue
Qualified shapes	qualifiedValueShape, qualifiedMinCount, qualifiedMaxCount

SHACL Core Constraints

Specifies conditions about the number of value nodes that conform to the given shape.



Type	Constraints
Cardinality	minCount, maxCount
Types of values	class, datatype, nodeKind
Values	node, in, hasValue
Range of values	minInclusive, maxInclusive minExclusive, maxExclusive
String based	minLength, maxLength, pattern, ~~stem~~, uniqueLang
Logical constraints	not, and, or, xone
Closed shapes	closed, ignoredProperties
Property pair constraints	equals, disjoint, lessThan, lessThanOrEquals
Non-validating constraints	name, value, defaultValue
Qualified shapes	qualifiedValueShape, qualifiedMinCount, qualifiedMaxCount

Property paths

-
  ```
  SPARQL        ex:parent
  SHACL         ex:parent
  ```

-
  ```
  SPARQL        ^ex:parent
  SHACL         [ sh:inversePath ex:parent ]
  ```

-
  ```
  SPARQL        ex:parent/ex:firstName
  SHACL         ( ex:parent ex:firstName )
  ```

-
  ```
  SPARQL        rdf:type/rdfs:subClassOf*
  SHACL         ( rdf:type [ sh:zeroOrMorePath rdfs:subClassOf ] )
  ```

-
  ```
  SPARQL        ex:father|ex:mother
  SHACL         [ sh:alternativePath ( ex:father ex:mother ) ]
  ```

Useful resources

- This document (available from the course website)
- *SHACL by example*
  - https://www.slideshare.net/jelabra/shacl-by-example
  - Not based on the final version of SHACL, but very close and with many examples.
- SHACL specification
  - https://www.w3.org/TR/shacl/
  - Relatively heavy reading
  - Use the table of contents to read up on specifics
- Turtle specification
  - https://www.w3.org/TR/turtle/
- Regular expressions online
  - https://regex101.com/

Hands-on exercises

- Instructions on the course website
- SHACL Tutorial Playground
  - No installation required
  - https://www.ida.liu.se/~robke04/SHACLTutorial/
- Tasks of increasing in difficulty with more than a single correct solution
- *Keep a backup of the shapes graph you are working on!*