Modelling Constraints with SHACL

Sebastián Ferrada

sebastian.ferrada@liu.se

March 21, 2023

LINKÖPING UNIVERSITY

RDF Graphs

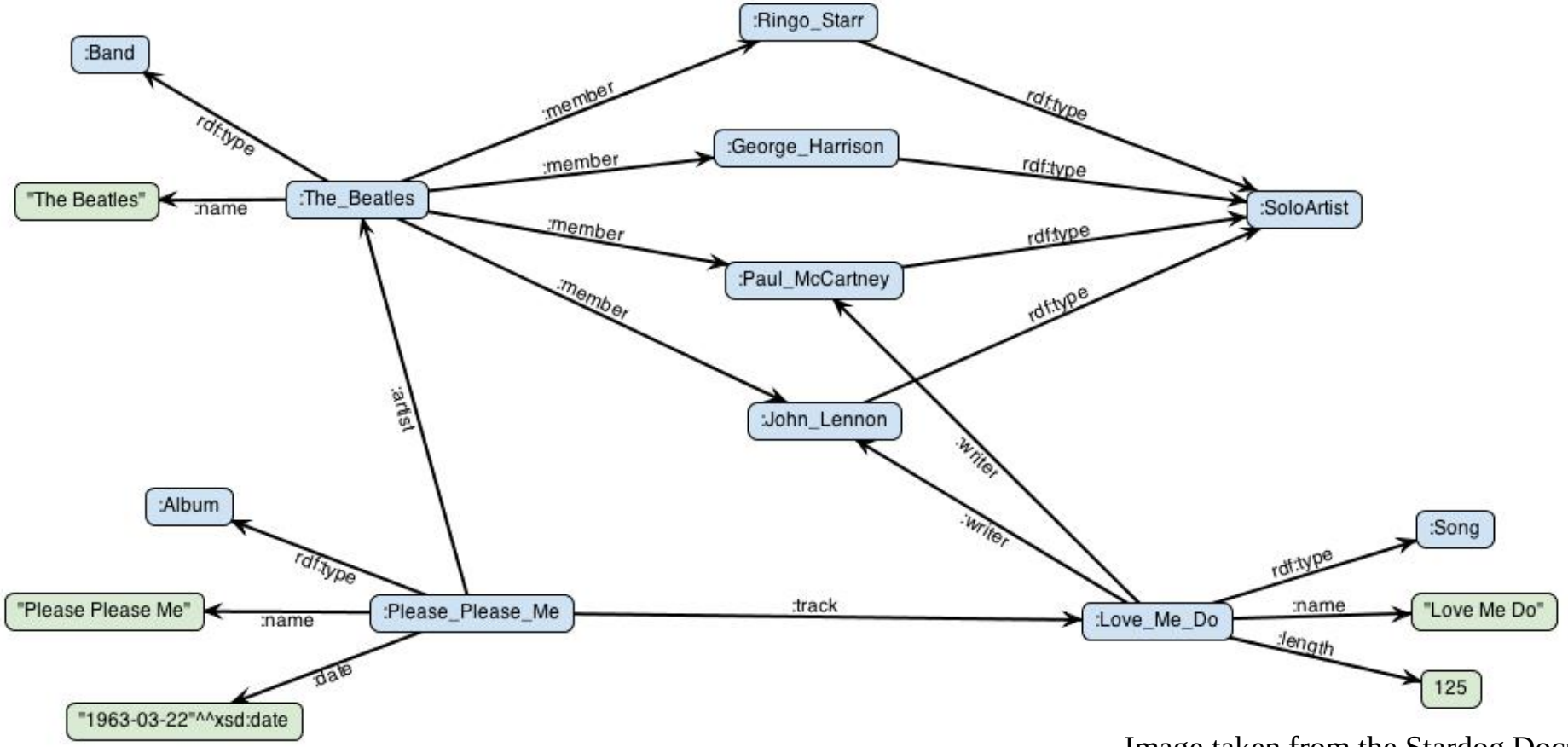


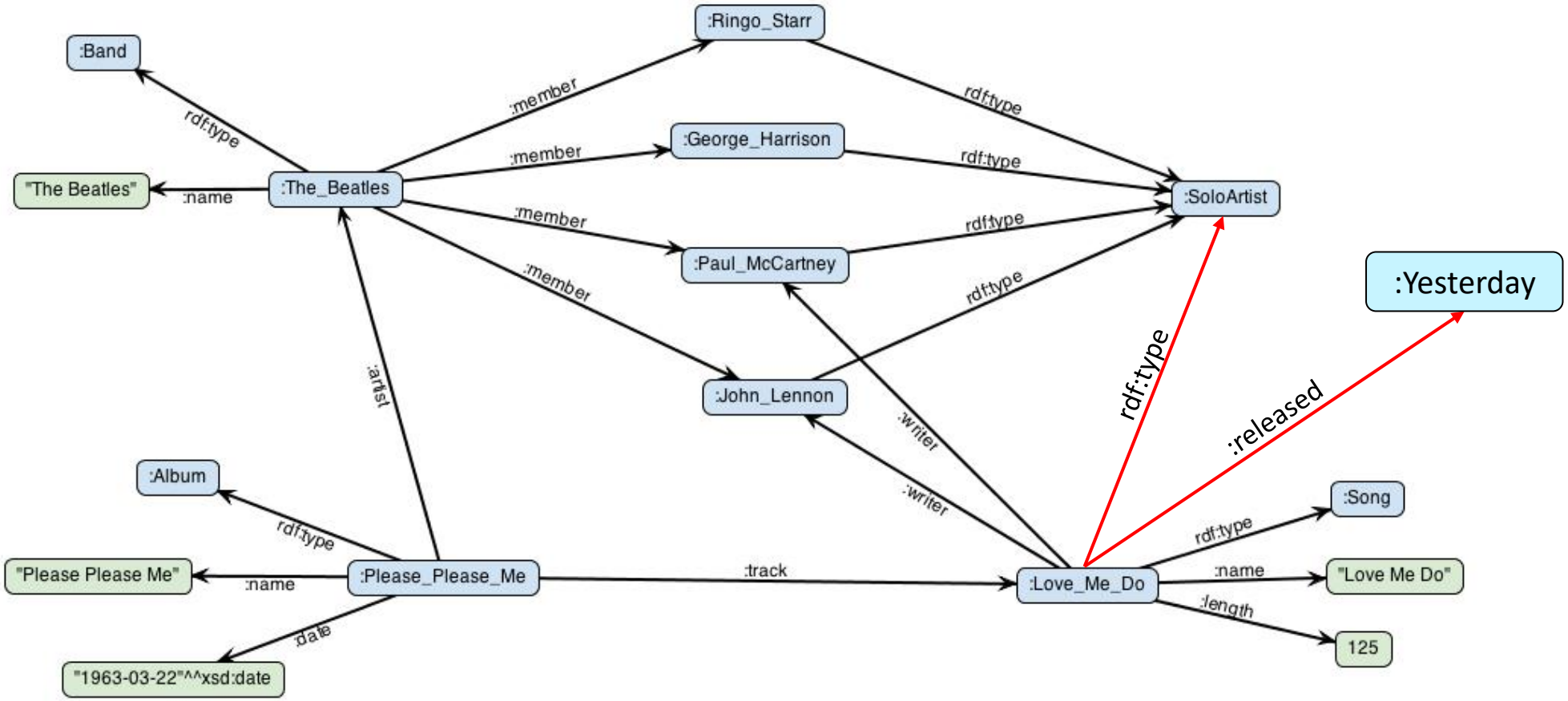
Image taken from the Stardog Documentation
https://docs.stardog.com/tutorials/virtual-graph-mappings

RDF Graphs

Very flexible! Add entities and properties as needed, no mandatory fields, no primary keys, etc!

Very flexible! We can introduce inconsistent/incoherent wrong/duplicate data without noticing

RDF Graphs


:Band
:Ringo_Starr
rdf:type
:member
rdf:type
:George_Harrison
:member
rdf:type
"The Beatles"
:name
:The_Beatles
:SoloArtist
:member
rdf:type
:Paul_McCartney
:member
rdf:type
:Yesterday
:John_Lennon
:artist
:writer
rdf:type
:released
:writer
:Album
:Song
rdf:type
rdf:type
"Please Please Me"
:name
:Please_Please_Me
:track
:Love_Me_Do
:name
"Love Me Do"
:length
:date
125
"1963-03-22"^^xsd:date

How to prevent inconsistencies?

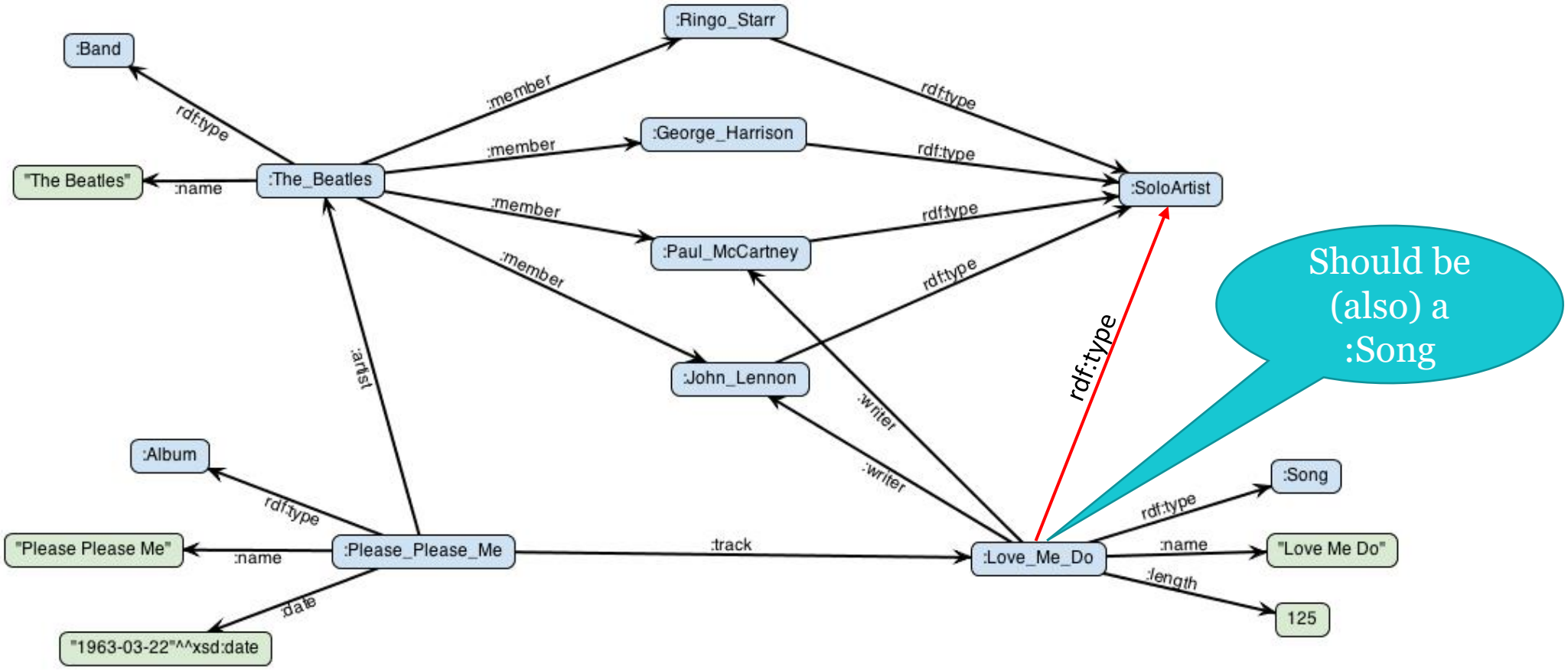
RDF Schema

- Allows to set a vocabulary for the data:
  - Class and property definition
  - Subclasses and subproperties
  - Range and domain definition

- Limited expressivity

```
:track rdfs:range :Song
```

RDF Graphs


:Band
:Ringo_Starr
rdf:type
:member
rdf:type
:member
:George_Harrison
rdf:type
"The Beatles"
:name
:The_Beatles
:SoloArtist
:member
rdf:type
:Paul_McCartney
:member
rdf:type
:John_Lennon
:artist
:writer
:writer
rdf:type
Should be (also) a :Song
:Album
rdf:type
:Song
rdf:type
"Please Please Me"
:name
:Please_Please_Me
:track
:Love_Me_Do
:name
"Love Me Do"
:length
125
:date
"1963-03-22"^^xsd:date

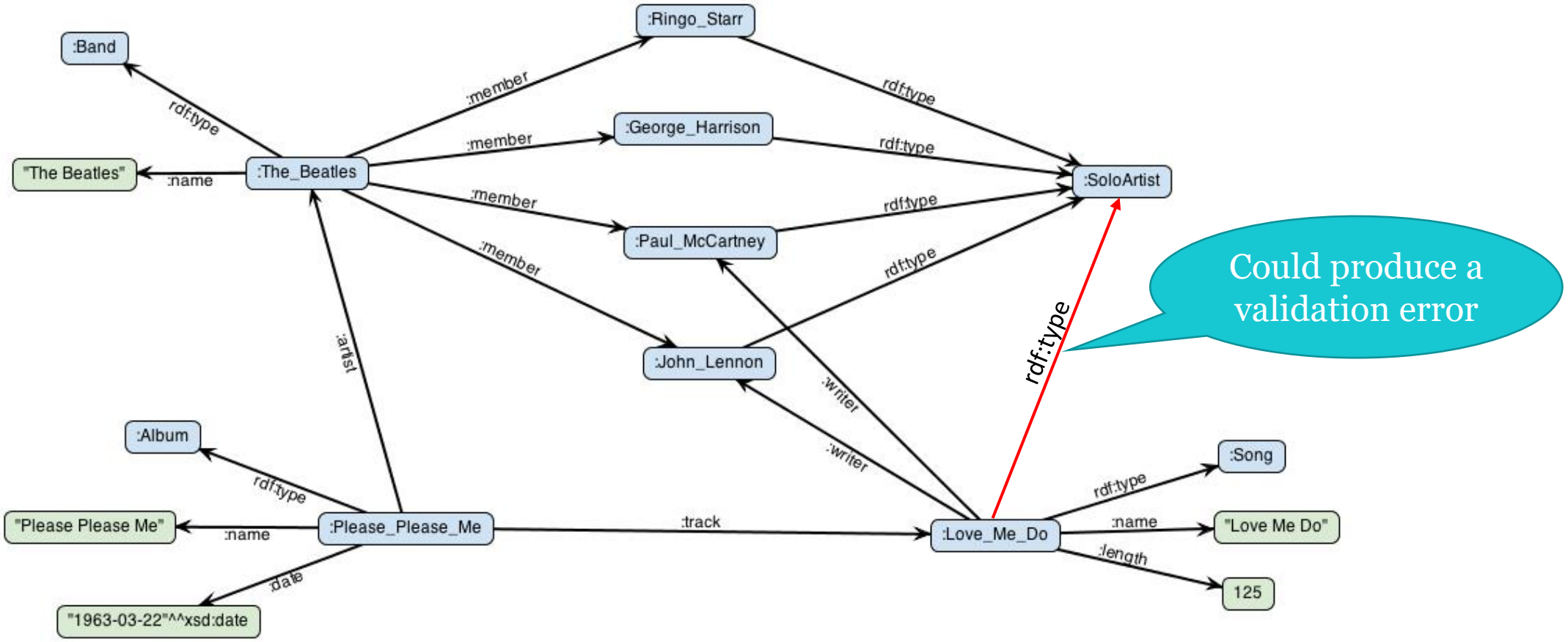
Web Ontology Language

- Targets logic and inferencing:
  - max/min cardinality
  - Intersection/union/difference
  - Class and property equivalence
  - Property transitivity and symmetry, etc.

- Can do validation using certain tools

```
:Song owl:disjointWith :SoloArtist
```

RDF Graphs


:Band
:Ringo_Starr
rdf:type
:member
rdf:type
:member
:George_Harrison
rdf:type
"The Beatles"
:name
:The_Beatles
:SoloArtist
:member
rdf:type
:Paul_McCartney
:member
rdf:type
:artist
:John_Lennon
rdf:type
:writer
:writer
Could produce a validation error
:Album
rdf:type
:Song
rdf:type
"Please Please Me"
:name
:Please_Please_Me
:track
:Love_Me_Do
:name
"Love Me Do"
:length
:date
125
"1963-03-22"^^xsd:date

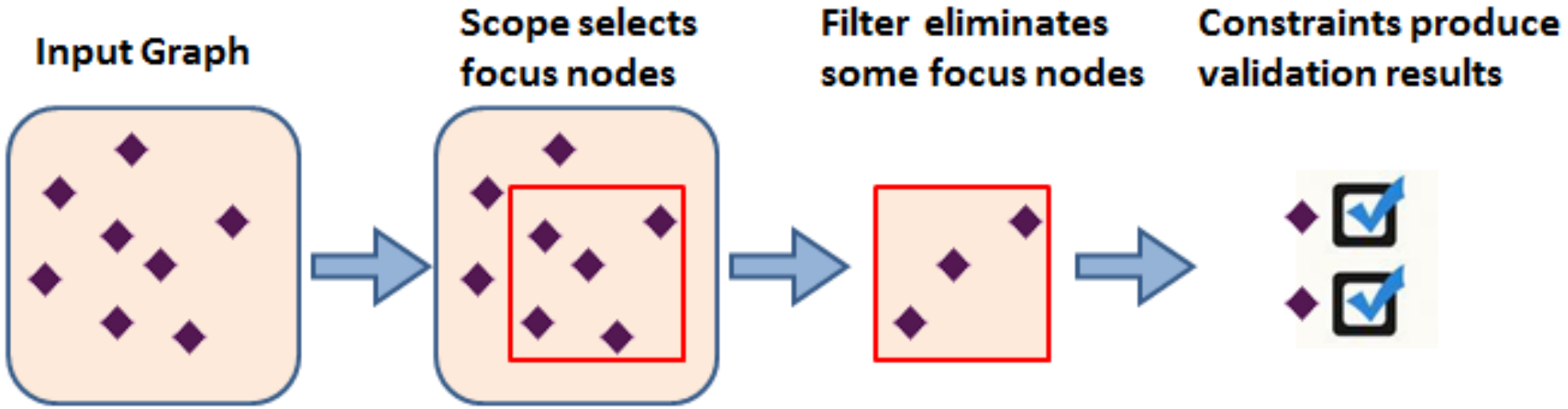
SHApes Constraint Language

- Allows RDF data to be validated against shapes

- Very expressive:
  - Constraints on cardinality, datatypes, ranges, patterns, etc.

- There is a shapes graph and a data graph
- In the example, we can define what a valid Song shape looks like

SHACL

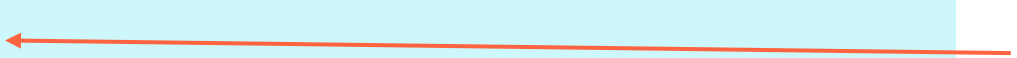


Produces a validation report!

SHACL – Define a Shape

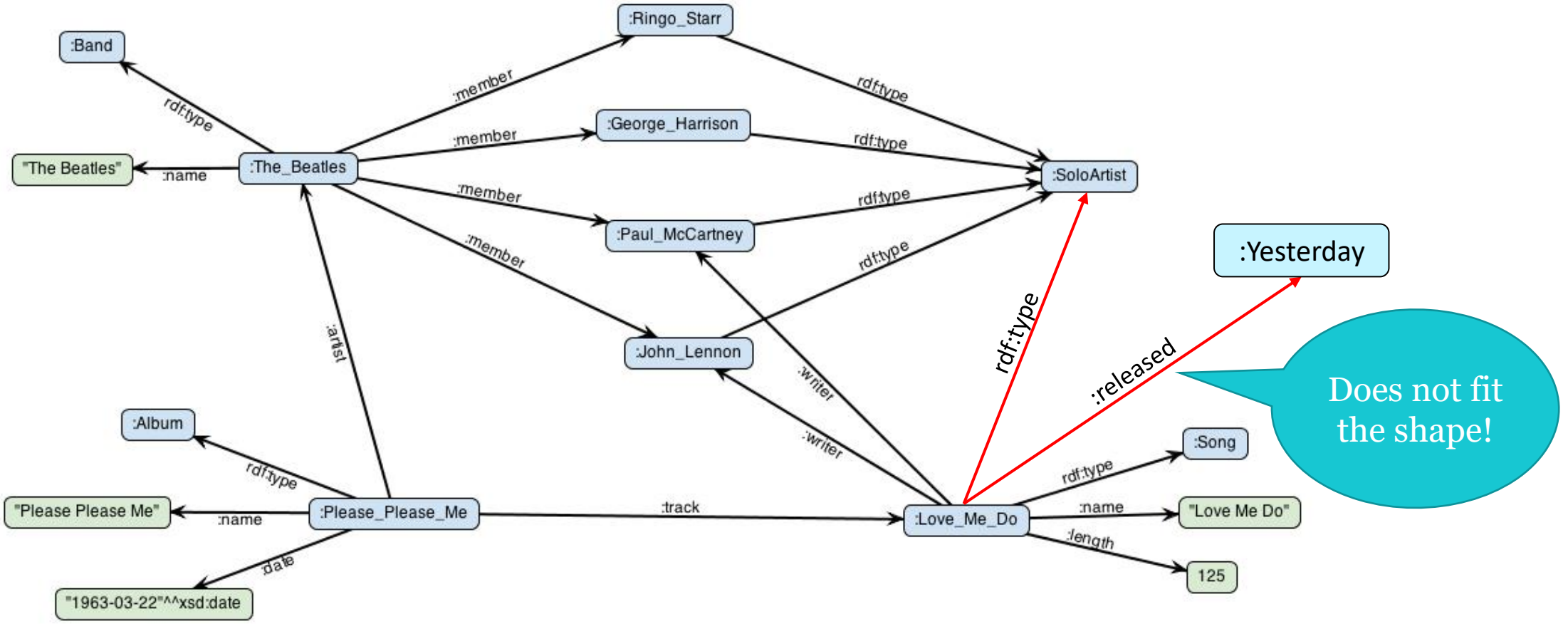
```
@prefix sh: <http://www.w3.org/ns/shacl#> .

ex:SongShape a sh:NodeShape ;
     sh:targetClass ex:Song ;
     sh:property [
          sh:path ex:released ;
          sh:datatype xsd:date .
     ] .
```



Square Brackets indicate a blank node

RDF Graphs


:Band
:Ringo_Starr
rdf:type
:member
rdf:type
:member
:George_Harrison
rdf:type
"The Beatles"
:name
:The_Beatles
:SoloArtist
:member
rdf:type
:Paul_McCartney
:member
rdf:type
:Yesterday
:John_Lennon
:artist
:writer
rdf:type
:released
:writer
:Album
:Song
rdf:type
rdf:type
"Please Please Me"
:name
:Please_Please_Me
:track
:Love_Me_Do
:name
"Love Me Do"
:length
:date
125
"1963-03-22"^^xsd:date
Does not fit the shape!

SHACL – Define a Shape

```
@prefix sh: <http://www.w3.org/ns/shacl#> .


ex:SongShape a sh:NodeShape ;
     sh:targetClass ex:Song ;
     sh:property [
          sh:path ex:writer ;
          sh:class ex:SoloArtist ;
          sh:minCount 1 .
     ] ..
```

SHACL - Targets

- A shape can target all instances of a class, nodes participating in a given relationship, or specific nodes:

```
ex:SongShape a sh:NodeShape ;
sh:targetClass ex:Song .
```

Targets all instances of ex:Song

```
ex:SongShape a sh:NodeShape ;
sh:targetNode ex:Love_Me_Do .
```

Targets a specific entity, ex:Love_Me_Do

```
ex:SongShape a sh:NodeShape ;
sh:targetSubjectsOf ex:writer
```

Targets all subjects of triples with predicate of ex:writer

```
ex:SongShape a sh:NodeShape ;
sh:targetObjectsOf ex:track .
```

Targets all objects of triples with predicate ex:track

SHACL – Validation Reports

- All OK!

```
[
      a sh:ValidationReport ;
      sh:conforms true
] .
```

SHACL – Validation Reports

- Invalid nodes found

```
[
  a sh:ValidationResult;
  sh:focusNode l:935;
  sh:resultMessage
   "Value does not have datatype <http://www.w3.org/2001/XMLSchema#string>";
  sh:resultPath n0:birthday;
  sh:resultSeverity sh:Violation;
  sh:sourceConstraintComponent sh:DatatypeConstraintComponent;
  sh:sourceShape [];
  sh:value "1948-10-09"^^XML:date
].
```

SHACL – Core Constraints

Type	Constraints
Cardinality	`minCount, maxCount`
Property Value Type	`class, datatype, nodeKind`
Property Values	`node, in, hasValue`
Value Range	`minInclusive, maxInclusive, minExclusive, maxExclusive`
String Values	`minLength, maxLength, pattern, uniqueLang`
Logical	`and, or, not, xone`
Closed Shapes	`closed, ignoredProperties`
Property Pair Constraint	`equals, disjoint, lessThan, lessThanOrEquals`
Non-validating	`name, value, defaultValue`
Qualified Shapes	`qualifiedValueShape, qualifiedMinCount, qualifiedMaxCount`

SHACL – Core Constraints

Constraint the max and min amount of values a property can have

Type	Constraints
Cardinality	`minCount, maxCount`
Property Value Type	`class, datatype, nodeKind`
Property Values	`node, in, hasValue`
Value Range	`minInclusive, maxInclusive, minExclusive, maxExclusive`
String Values	`minLength, maxLength, pattern, uniqueLang`
Logical	`and, or, not, xone`
Closed Shapes	`closed, ignoredProperties`
Property Pair Constraint	`equals, disjoint, lessThan, lessThanOrEquals`
Non-validating	`name, value, defaultValue`
Qualified Shapes	`qualifiedValueShape, qualifiedMinCount, qualifiedMaxCount`

SHACL – Cardinality Constraints

```
ex:SongShape a sh:NodeShape ;
     sh:targetClass ex:Song ;
     sh:property [
          sh:path ex:writer ;
          sh:class ex:SoloArtist ;
          sh:minCount 1 .
     ] .
```

SHACL – Core Constraints

Constraint the types of values a property can have

Type	Constraints
Cardinality	`minCount, maxCount`
Property Value Type	`class, datatype, nodeKind`
Property Values	`node, in, hasValue`
Value Range	`minInclusive, maxInclusive, minExclusive, maxExclusive`
String Values	`minLength, maxLength, pattern, uniqueLang`
Logical	`and, or, not, xone`
Closed Shapes	`closed, ignoredProperties`
Property Pair Constraint	`equals, disjoint, lessThan, lessThanOrEquals`
Non-validating	`name, value, defaultValue`
Qualified Shapes	`qualifiedValueShape, qualifiedMinCount, qualifiedMaxCount`

SHACL – Value Type Constraints

```
ex:SongShape a sh:NodeShape ;
      sh:targetClass ex:Song ;
      sh:property [
            sh:path ex:name ;
            sh:datatype xsd:string .
      ] .
```

```
ex:SongShape a sh:NodeShape ;
      sh:targetClass ex:Song ;
      sh:property [
            sh:path ex:writer ;
            sh:class ex:SoloArtist ;
            sh:minCount 1 .
      ] .
```

SHACL – Core Constraints

Constraint the values that a property can have

Type	Constraints
Cardinality	`minCount, maxCount`
Property Value Type	`class, datatype, nodeKind`
Property Values	`node, in, hasValue`
Value Range	`minInclusive, maxInclusive, minExclusive, maxExclusive`
String Values	`minLength, maxLength, pattern, uniqueLang`
Logical	`and, or, not, xone`
Closed Shapes	`closed, ignoredProperties`
Property Pair Constraint	`equals, disjoint, lessThan, lessThanOrEquals`
Non-validating	`name, value, defaultValue`
Qualified Shapes	`qualifiedValueShape, qualifiedMinCount, qualifiedMaxCount`

SHACL – Core Constraints

Constraint the values a property can have to a given range

Type	Constraints
Cardinality	`minCount, maxCount`
Property Value Type	`class, datatype, nodeKind`
Property Values	`node, in, hasValue`
Value Range	`minInclusive, maxInclusive, minExclusive, maxExclusive`
String Values	`minLength, maxLength, pattern, uniqueLang`
Logical	`and, or, not, xone`
Closed Shapes	`closed, ignoredProperties`
Property Pair Constraint	`equals, disjoint, lessThan, lessThanOrEquals`
Non-validating	`name, value, defaultValue`
Qualified Shapes	`qualifiedValueShape, qualifiedMinCount, qualifiedMaxCount`

SHACL – Core Constraints

Constraint the values a string valued property can have

Type	Constraints
Cardinality	`minCount, maxCount`
Property Value Type	`class, datatype, nodeKind`
Property Values	`node, in, hasValue`
Value Range	`minInclusive, maxInclusive, minExclusive, maxExclusive`
String Values	`minLength, maxLength, pattern, uniqueLang`
Logical	`and, or, not, xone`
Closed Shapes	`closed, ignoredProperties`
Property Pair Constraint	`equals, disjoint, lessThan, lessThanOrEquals`
Non-validating	`name, value, defaultValue`
Qualified Shapes	`qualifiedValueShape, qualifiedMinCount, qualifiedMaxCount`

SHACL – String Values

```
ex:PersonShape a sh:NodeShape ;
      sh:targetClass ex:Person ;
      sh:property [
            sh:path ex:personalNumber ;
            sh:pattern “\\d{8}-\\d{4}”
      ].
```

```
ex:Alice a ex:Person;
ex:personalNumber “19990505-3221”
```

```
ex:Mike a ex:Person;
ex:personalNumber “012112-3244”
```

SHACL – Core Constraints

Logic operations among constraints

Type	Constraints
Cardinality	`minCount, maxCount`
Property Value Type	`class, datatype, nodeKind`
Property Values	`node, in, hasValue`
Value Range	`minInclusive, maxInclusive, minExclusive, maxExclusive`
String Values	`minLength, maxLength, pattern, uniqueLang`
Logical	`and, or, not, xone`
Closed Shapes	`closed, ignoredProperties`
Property Pair Constraint	`equals, disjoint, lessThan, lessThanOrEquals`
Non-validating	`name, value, defaultValue`
Qualified Shapes	`qualifiedValueShape, qualifiedMinCount, qualifiedMaxCount`

SHACL – Logical Operators

- A song must have at least one writer or be a track on an album

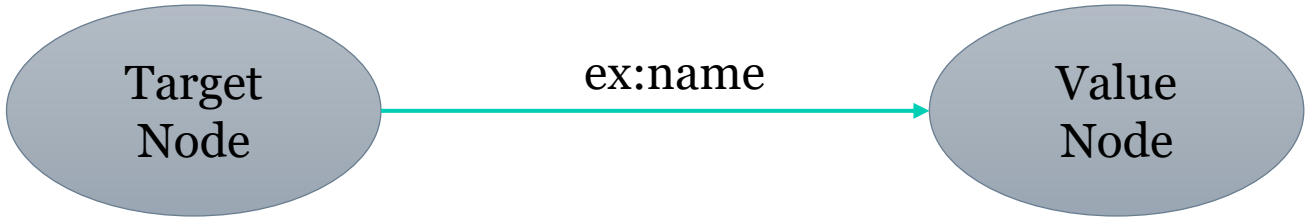
```
ex:SongShape a sh:NodeShape ;
      sh:targetClass ex:Song ;
      sh:or (
            [sh:property [
                  sh:path ex:writer ;
                  sh:class ex:SoloArtist ;
                  sh:minCount 1 .]]
            [sh:property [
                  sh:inversePath ex:track ;
                  sh:class ex:Album]]).
```

Tangent: SHACL Paths

SHACL – Paths

- Paths are relative to the target nodes, e.g.,

```
sh:property [sh:path ex:name ; ...] .
```

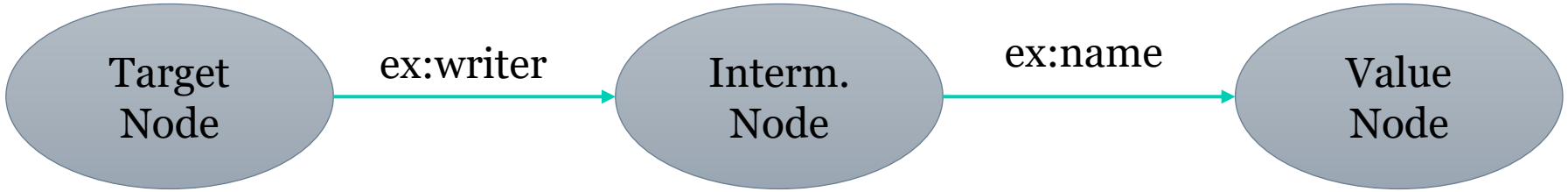


SHACL – Inverse Paths

```
sh:property [sh:inversePath ex:track ; ...]
```

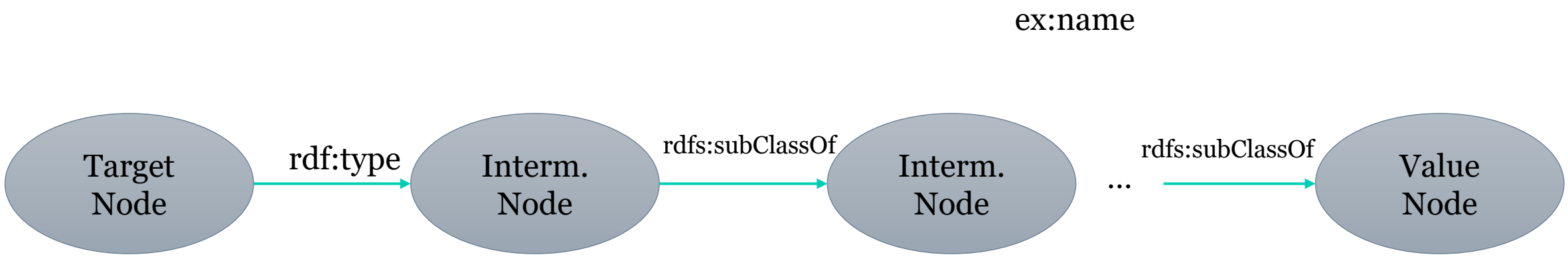
SHACL – Concatenate Paths

```
sh:property [sh:path (ex:writer ex:name) ; ...] .
```



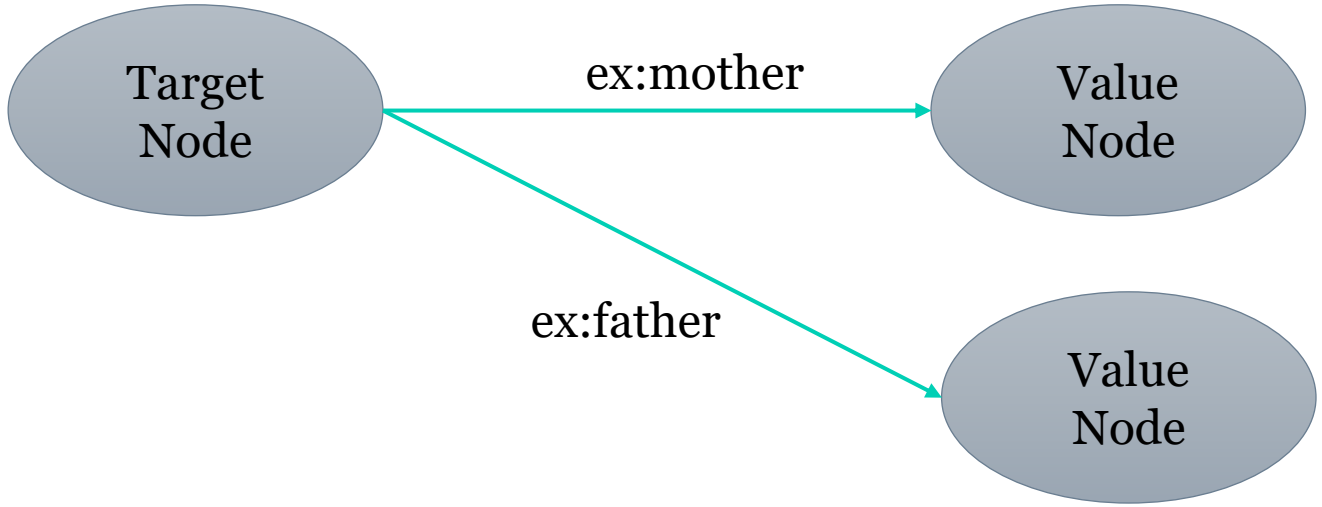
SHACL – Zero or More

```
sh:property [sh:path (rdf:type [ sh:zeroOrMorePath rdfs:subClassOf];]
```



SHACL – Alternative Paths

```
sh:property [sh:alternativePath ( ex:father ex:mother );]
```



End of the Tangent

SHACL – Logical Operators

- A song must have at least one writer or be a track on an album

```
ex:SongShape a sh:NodeShape ;
      sh:targetClass ex:Song ;
      sh:or (
            [sh:property [
                  sh:path ex:writer ;
                  sh:class ex:SoloArtist ;
                  sh:minCount 1 .]]
            [sh:property [
                  sh:inversePath ex:track ;
                  sh:class ex:Album]]).
```

SHACL – Core Constraints

Nodes can only have the properties mentioned in the shape

Type	Constraints
Cardinality	`minCount, maxCount`
Property Value Type	`class, datatype, nodeKind`
Property Values	`node, in, hasValue`
Value Range	`minInclusive, maxInclusive, minExclusive, maxExclusive`
String Values	`minLength, maxLength, pattern, uniqueLang`
Logical	`and, or, not, xone`
Closed Shapes	`closed, ignoredProperties`
Property Pair Constraint	`equals, disjoint, lessThan, lessThanOrEquals`
Non-validating	`name, value, defaultValue`
Qualified Shapes	`qualifiedValueShape, qualifiedMinCount, qualifiedMaxCount`

SHACL – Closed Shapes

```
ex:SongShape a sh:NodeShape ;
      sh:targetNode ex:Love_Me_Do;
      sh:property [
            sh:path ex:writer ;
            sh:class ex:SoloArtist ;
            sh:minCount 1 .
      ] .
      sh:closed true .
```

```
ex:Love_Me_Do ex:writer
                  ex:John_Lennon .
```

```
ex:Love_Me_Do a ex:Song ;
       ex:name “Love me do”;
       ex:length 145 ;
       ex:writer ex:John_Lennon .
```

SHACL – Closed Shapes

```
ex:SongShape a sh:NodeShape ;
      sh:targetNode ex:Love_Me_Do;
      sh:property [
            sh:path ex:writer ;
            sh:class ex:SoloArtist ;
            sh:minCount 1 .
      ] .
      sh:closed true ;
      sh:ignoredProperties
      (ex:name ex:length rdf:type) .
```

```
ex:Love_Me_Do ex:writer
                  ex:John_Lennon .
```

```
ex:Love_Me_Do a ex:Song ;
        ex:name “Love me do”;
        ex:length 145 ;
        ex:writer ex:John_Lennon .
```

SHACL – Core Constraints

Constraint on value nodes in relation to other properties

Type	Constraints
Cardinality	`minCount, maxCount`
Property Value Type	`class, datatype, nodeKind`
Property Values	`node, in, hasValue`
Value Range	`minInclusive, maxInclusive, minExclusive, maxExclusive`
String Values	`minLength, maxLength, pattern, uniqueLang`
Logical	`and, or, not, xone`
Closed Shapes	`closed, ignoredProperties`
Property Pair Constraint	`equals, disjoint, lessThan, lessThanOrEquals`
Non-validating	`name, value, defaultValue`
Qualified Shapes	`qualifiedValueShape, qualifiedMinCount, qualifiedMaxCount`

SHACL – Property Pair Constraint

```
ex:EmployeeShape a sh:NodeShape ;
      sh:targetNode (ex:Mike ex:Alice );
      sh:property [
            sh:path ex:birthDate ;
            sh:lessThan ex:contractStart
      ].
```

```
ex:Alice a ex:Person;
ex:birthDate “1994-05-05” ;
ex:contractStart “2021-01-01”.
```

```
ex:Mike a ex:Person;
ex:birthDate “2001-21-12” ;
ex:contractStart “2001-01-01”.
```

SHACL – Core Constraints

Optional constraints for recommended values or to add metadata, not validated.

Type	Constraints
Cardinality	`minCount, maxCount`
Property Value Type	`class, datatype, nodeKind`
Property Values	`node, in, hasValue`
Value Range	`minInclusive, maxInclusive, minExclusive, maxExclusive`
String Values	`minLength, maxLength, pattern, uniqueLang`
Logical	`and, or, not, xone`
Closed Shapes	`closed, ignoredProperties`
Property Pair Constraint	`equals, disjoint, lessThan, lessThanOrEquals`
Non-validating	`name, value, defaultValue`
Qualified Shapes	`qualifiedValueShape, qualifiedMinCount, qualifiedMaxCount`

SHACL – Non-validating Characteristics

```
ex:PersonFormShape a sh:NodeShape ;
     sh:property [
          sh:path ex:firstName ;
          sh:name "first name" ;
          sh:description "The person's given name(s)" ;
          sh:order 0 ;
          sh:group ex:NameGroup ;
     ] .
```

SHACL – Core Constraints

Constraints to the number of value nodes that conform to a constraint

Type	Constraints
Cardinality	`minCount, maxCount`
Property Value Type	`class, datatype, nodeKind`
Property Values	`node, in, hasValue`
Value Range	`minInclusive, maxInclusive, minExclusive, maxExclusive`
String Values	`minLength, maxLength, pattern, uniqueLang`
Logical	`and, or, not, xone`
Closed Shapes	`closed, ignoredProperties`
Property Pair Constraint	`equals, disjoint, lessThan, lessThanOrEquals`
Non-validating	`name, value, defaultValue`
Qualified Shapes	`qualifiedValueShape, qualifiedMinCount, qualifiedMaxCount`

SHACL – Qualified Shapes

```
ex:QualifiedValueShapeExampleShape a sh:NodeShape ;
      sh:targetNode ex:ValidResource ;
      sh:property [
            sh:path ex:parent ;
            sh:minCount 2 ;
            sh:maxCount 2 ;
            sh:qualifiedValueShape [
                  sh:path ex:gender ;
                  sh:hasValue ex:female ;
            ] ;
            sh:qualifiedMinCount 1 ;
] .
```

SHACL – Custom SPARQL Constraints

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

Useful Resources

- SHACL Specification: https://www.w3.org/TR/shacl/

- SHACL By Example: https://www.slideshare.net/jelabra/shacl-by-example

Hands-on Exercises

- https://www.ida.liu.se/research/semanticweb/events/SemWebCourse2023/HandsOnSHACL.shtml