Laboratory: Constraint Rules

Pablo Mollá Chárlez

January 31, 2025

1 SHACL Validation of an RDF Graph

SHACL Validation Processor: We will use the open-access processor available at the following address: https://shacl.org/playground/.

New Dataset: Consider the graph given in the file rdf-graph.ttl as an example. The **dataset** contains the following information:

```
@prefix ex: <http://example.org/> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix schema: <http://schema.org/> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .

# Define the individual Paul as an instance of schema:Student
ex:Paul a schema:Student ;
    ex:hasSSN "123-45-6789"^^xsd:string ;
    schema:email "paul@example.org"^^xsd:string ;
    ex:doctoralAdviser ex:Pierre ;
    ex:doctoralAdviser ex:Anne ;
    ex:phdCandidateIn ex:CS ;
    schema:affiliation ex:UPS .

# Define the supervisor Pierre
ex:Pierre a schema:Person ;
    ex:phdIn ex:CS ;
    schema:affiliation ex:IPP .

# Define the supervisor Anne
ex:Anne a schema:Person ;
    ex:phdIn ex:Mathematics ;
    schema:affiliation ex:UPS .

# Define the university UPS
ex:UPS a ex:ResearchOrganization ;
    schema:name "Université Paris-Saclay"@fr ;
    schema:locatedin ex:France.

# Define the organization IPP
ex:IPP a schema:ResearchOrganization ;
    schema:name "Institut Polytechnique de Paris"@fr ;
     schema:locatedin ex:France.
```

Propose SHACL shapes (constraints) to express the following constraints. Verify the validity of the solutions by extending the graph. In the report, it is important to include these extensions in addition to the SHACL constraints.

What Are We Doing?

- We have an RDF file (rdf-graph.ttl) describing people (students and supervisors), their universities, and affiliations. The goal is to express a rule (constraint) that every object, students, supervisors, universities or affiliations must satisfy. We use the SHACL (Shapes Constraint Language) to define these rules in a structured way, so that any RDF data must respect these constraints to be considered valid. By uploading both our RDF file and our SHACL shapes file to the SHACL Playground, we can see if our data is valid or if something is missing.

Let's solve each one of the following constraints:

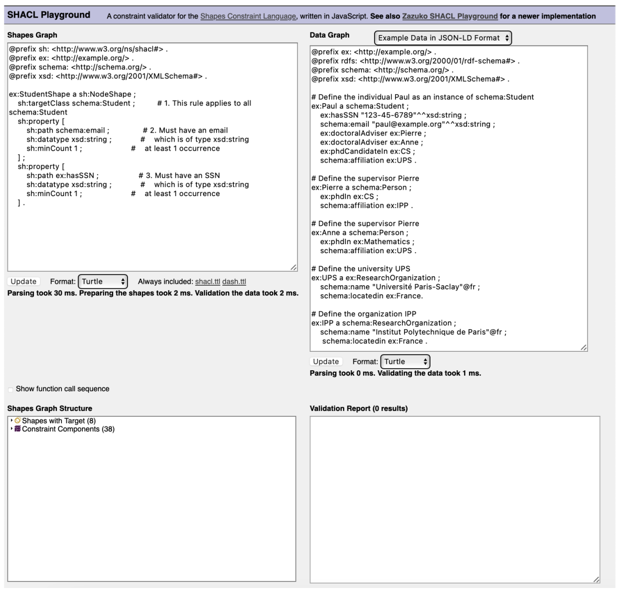
1. **Exercise** Define a constraint indicating that each student must have an email and a Social Security Number (SSN).

   Currently, Paul is our only **schema:Student** and he already has both **schema:email** and **ex:hasSSN**. He should pass the new SHACL constraint that we are going to define as follows:

```
@prefix sh: <http://www.w3.org/ns/shacl#> .
@prefix ex: <http://example.org/> .
@prefix schema: <http://schema.org/> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .

# This line declares ex:StudentShape as a SHACL NodeShape, i.e.
# it defines the set of constraints applied to schema:Student
ex:StudentShape a sh:NodeShape ;
    sh:targetClass schema:Student ;          # 1. This rule applies to all schema:Student
    sh:property [
        sh:path schema:email ;               # 2. Must have an email
        sh:datatype xsd:string ;             #    which is of type xsd:string
        sh:minCount 1 ;                      #    at least 1 occurrence
    ] ;
    sh:property [
        sh:path ex:hasSSN ;                  # 3. Must have an SSN
        sh:datatype xsd:string ;             #    which is of type xsd:string
        sh:minCount 1 ;                      #    at least 1 occurrence
    ] .
```

   In order to verify whether the rule is appropiately working, we can use the SHACL Playground. As it can be observed, no violations of the rule appear as outcome.



SHACL Playground A constraint validator for the Shapes Constraint Language, written in JavaScript. **See also Zazuko SHACL Playground for a newer implementation**

Shapes Graph

```
@prefix sh: <http://www.w3.org/ns/shacl#> .
@prefix ex: <http://example.org/> .
@prefix schema: <http://schema.org/> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .

ex:StudentShape a sh:NodeShape ;
   sh:targetClass schema:Student ;        # 1. This rule applies to all
schema:Student
   sh:property [
      sh:path schema:email ;             # 2. Must have an email
      sh:datatype xsd:string ;          #    which is of type xsd:string
      sh:minCount 1 ;                  #    at least 1 occurrence
   ] ;
   sh:property [
      sh:path ex:hasSSN ;               # 3. Must have an SSN
      sh:datatype xsd:string ;          #    which is of type xsd:string
      sh:minCount 1 ;                  #    at least 1 occurrence
   ] .
```

Update Format: Turtle Always included: shacl.ttl dash.ttl

Parsing took 30 ms. Preparing the shapes took 2 ms. Validation the data took 2 ms.

Show function call sequence

Data Graph Example Data in JSON-LD Format

```
@prefix ex: <http://example.org/> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix schema: <http://schema.org/> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .

# Define the individual Paul as an instance of schema:Student
ex:Paul a schema:Student ;
   ex:hasSSN "123-45-6789"^^xsd:string ;
   schema:email "paul@example.org"^^xsd:string ;
   ex:doctoralAdviser ex:Pierre ;
   ex:doctoralAdviser ex:Anne ;
   ex:phdCandidateIn ex:CS ;
   schema:affiliation ex:UPS .

# Define the supervisor Pierre
ex:Pierre a schema:Person ;
   ex:phdIn ex:CS ;
   schema:affiliation ex:IPP .

# Define the supervisor Pierre
ex:Anne a schema:Person ;
   ex:phdIn ex:Mathematics ;
   schema:affiliation ex:UPS .

# Define the university UPS
ex:UPS a ex:ResearchOrganization ;
   schema:name "Université Paris-Saclay"@fr ;
   schema:locatedin ex:France.

# Define the organization IPP
ex:IPP a schema:ResearchOrganization ;
   schema:name "Institut Polytechnique de Paris"@fr ;
    schema:locatedin ex:France .
```

Update Format: Turtle

Parsing took 0 ms. Validating the data took 1 ms.

Shapes Graph Structure

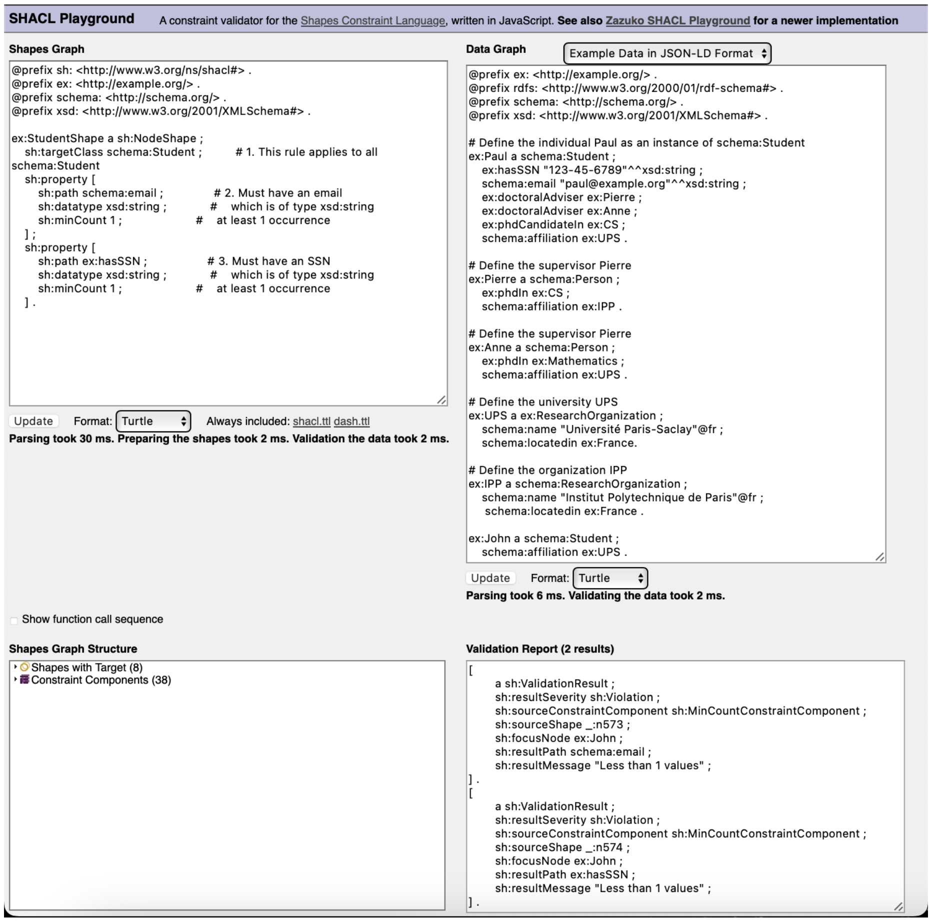
- Shapes with Target (8)
- Constraint Components (38)

Validation Report (0 results)

Figure 1: Check Query 1

However, if we introduce, for instance a new student in our RDF file, we would obtain the following error:

```
ex:John a schema:Student ;
    schema:affiliation ex:UPS .
```


SHACL Playground
A constraint validator for the Shapes Constraint Language, written in JavaScript. See also Zazuko SHACL Playground for a newer implementation

Shapes Graph

```
@prefix sh: <http://www.w3.org/ns/shacl#> .
@prefix ex: <http://example.org/> .
@prefix schema: <http://schema.org/> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .

ex:StudentShape a sh:NodeShape ;
   sh:targetClass schema:Student ;         # 1. This rule applies to all
schema:Student
   sh:property [
      sh:path schema:email ;             # 2. Must have an email
      sh:datatype xsd:string ;           #    which is of type xsd:string
      sh:minCount 1 ;                  #    at least 1 occurrence
   ] ;
   sh:property [
      sh:path ex:hasSSN ;                # 3. Must have an SSN
      sh:datatype xsd:string ;           #    which is of type xsd:string
      sh:minCount 1 ;                  #    at least 1 occurrence
   ] .
```

Update Format: Turtle Always included: shacl.ttl dash.ttl

Parsing took 30 ms. Preparing the shapes took 2 ms. Validation the data took 2 ms.

Data Graph Example Data in JSON-LD Format

```
@prefix ex: <http://example.org/> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix schema: <http://schema.org/> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .

# Define the individual Paul as an instance of schema:Student
ex:Paul a schema:Student ;
   ex:hasSSN "123-45-6789"^^xsd:string ;
   schema:email "paul@example.org"^^xsd:string ;
   ex:doctoralAdviser ex:Pierre ;
   ex:doctoralAdviser ex:Anne ;
   ex:phdCandidateIn ex:CS ;
   schema:affiliation ex:UPS .

# Define the supervisor Pierre
ex:Pierre a schema:Person ;
   ex:phdIn ex:CS ;
   schema:affiliation ex:IPP .

# Define the supervisor Pierre
ex:Anne a schema:Person ;
   ex:phdIn ex:Mathematics ;
   schema:affiliation ex:UPS .

# Define the university UPS
ex:UPS a ex:ResearchOrganization ;
   schema:name "Université Paris-Saclay"@fr ;
   schema:locatedin ex:France.

# Define the organization IPP
ex:IPP a schema:ResearchOrganization ;
   schema:name "Institut Polytechnique de Paris"@fr ;
    schema:locatedin ex:France .

ex:John a schema:Student ;
   schema:affiliation ex:UPS .
```

Update Format: Turtle

Parsing took 6 ms. Validating the data took 2 ms.

Show function call sequence

Shapes Graph Structure

Shapes with Target (8)
Constraint Components (38)

Validation Report (2 results)

```
[
      a sh:ValidationResult ;
      sh:resultSeverity sh:Violation ;
      sh:sourceConstraintComponent sh:MinCountConstraintComponent ;
      sh:sourceShape _:n573 ;
      sh:focusNode ex:John ;
      sh:resultPath schema:email ;
      sh:resultMessage "Less than 1 values" ;
] .
[
      a sh:ValidationResult ;
      sh:resultSeverity sh:Violation ;
      sh:sourceConstraintComponent sh:MinCountConstraintComponent ;
      sh:sourceShape _:n574 ;
      sh:focusNode ex:John ;
      sh:resultPath ex:hasSSN ;
      sh:resultMessage "Less than 1 values" ;
] .
```

Figure 2: Error Query 1

2. **Exercise** Ensure that the **SSN is a string** and that the **email is a string**.

```
ex:StudentShape a sh:NodeShape ;
  sh:targetClass schema:Student ;
  sh:property [
    sh:path ex:hasSSN ;
    sh:datatype xsd:string
  ] ;
  sh:property [
    sh:path schema:email ;
    sh:datatype xsd:string
  ] .
```

3. **Exercise** Add a constraint to enforce that each student must have **exactly one SSN**.

```
ex:ExactlyOneSSNShape a sh:NodeShape ;
  sh:targetClass schema:Student ;
  sh:property [
    sh:path ex:hasSSN ;
    sh:minCount 1 ;
    sh:maxCount 1
  ] .
```

4. **Exercise** Add a constraint to specify that a **student may have at most one email**.

```
ex:MaxOneEmailShape a sh:NodeShape ;
  sh:targetClass schema:Student ;
  sh:property [
    sh:path schema:email ;
    sh:maxCount 1
  ] .
```

5. **Exercise** Specify a constraint to enforce that the **email must follow a valid format** (e.g., name@example.com).

```
ex:ValidEmailShape a sh:NodeShape ;
  sh:targetClass schema:Student ;
  sh:property [
    sh:path schema:email ;
    sh:pattern "^[^@]+@[^@]+\\.[^@]+$" ;
    sh:message "Email must match a name@domain pattern."
  ] .
```

6. **Exercise** Add a constraint ensuring that the **SSN** is unique—two students cannot have the same **SSN**.

```
ex:UniqueSSNShape a sh:NodeShape ;
  sh:targetClass schema:Student ;
  sh:sparql [
    sh:message "SSN must be unique among all students.";
    sh:select """
      SELECT ?this
      WHERE {
        ?this ex:hasSSN ?ssn .
        ?other ex:hasSSN ?ssn ;
               a schema:Student .
        FILTER (?other != ?this)
      }
    """
  ] .
```

Another solution for this constraint would be:

```
ex:InverseSSNConstraintShape a sh:NodeShape ;
  sh:targetSubjectsOf ex:hasSSN ;
  sh:sparql [
    a sh:SPARQLConstraint ;
    sh:message "The SSN must be unique." ;
    sh:select """
      SELECT $this
      WHERE {
        $this ex:hasSSN ?ssn .
        ?other ex:hasSSN ?ssn .
        FILTER ($this != ?other)
      }
    """ ;
  ] .
```

They both enforce a "unique SSN" rule with SPARQL, but the targets differ:

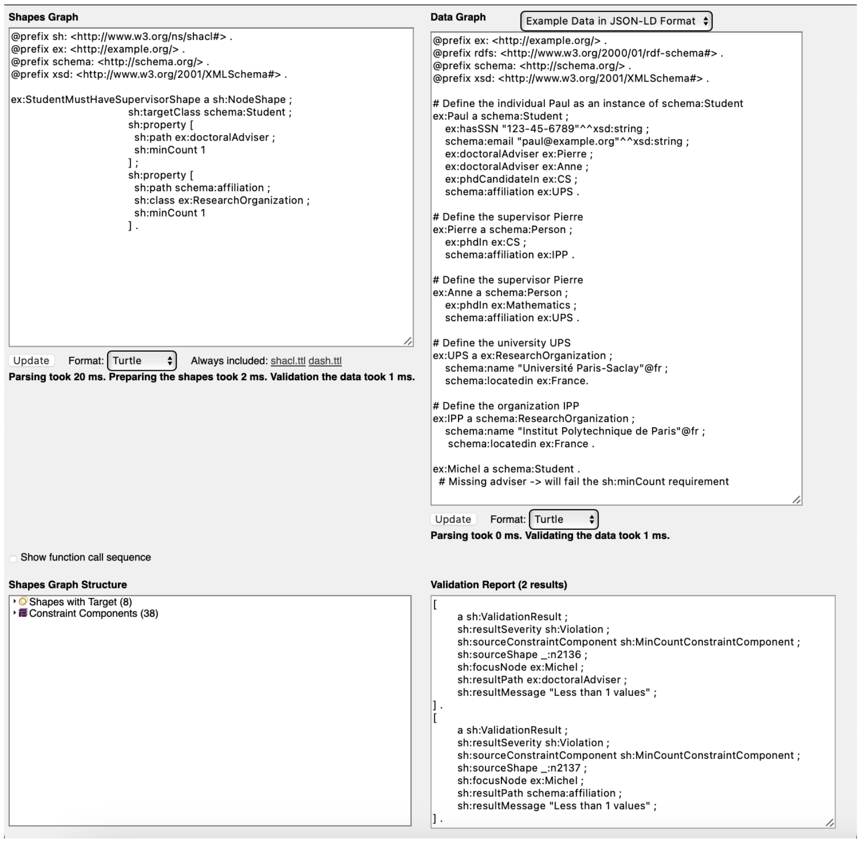
- **ex:InverseSSNConstraintShape**: applies to any subject that has **ex:hasSSN**.
- **ex:UniqueSSNShape**: applies only to instances of **schema:Student** and checks uniqueness among other **schema:Students**.

So they are not exactly the same constraint.

7. **Exercise** Add a constraint to enforce that a student must have a supervisor and be affiliated with a **schema:ResearchOrganization**.

```
ex:StudentMustHaveSupervisorShape a sh:NodeShape ;
  sh:targetClass schema:Student ;
  sh:property [
    sh:path ex:doctoralAdviser ;
    sh:minCount 1
  ] ;
  sh:property [
    sh:path schema:affiliation ;
    sh:class schema:ResearchOrganization ;
    sh:minCount 1
  ] .
```

In this exercise, when we define a student like Michel with no affiliation (**schema:ResearchOrganization**) nor doctoral adviser (**ex:doctoralAdviser**), we obtain:


Shapes Graph
Data Graph
Example Data in JSON-LD Format
Update
Format: Turtle
Always included: shacl.ttl dash.ttl
Show function call sequence
Shapes Graph Structure
Shapes with Target (8)
Constraint Components (38)
Validation Report (2 results)

```
@prefix sh: <http://www.w3.org/ns/shacl#> .
@prefix ex: <http://example.org/> .
@prefix schema: <http://schema.org/> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .

ex:StudentMustHaveSupervisorShape a sh:NodeShape ;
                    sh:targetClass schema:Student ;
                    sh:property [
                      sh:path ex:doctoralAdviser ;
                      sh:minCount 1
                    ] ;
                    sh:property [
                      sh:path schema:affiliation ;
                      sh:class ex:ResearchOrganization ;
                      sh:minCount 1
                    ] .
```

Parsing took 20 ms. Preparing the shapes took 2 ms. Validation the data took 1 ms.

```
@prefix ex: <http://example.org/> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix schema: <http://schema.org/> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .

# Define the individual Paul as an instance of schema:Student
ex:Paul a schema:Student ;
   ex:hasSSN "123-45-6789"^^xsd:string ;
   schema:email "paul@example.org"^^xsd:string ;
   ex:doctoralAdviser ex:Pierre ;
   ex:doctoralAdviser ex:Anne ;
   ex:phdCandidateIn ex:CS ;
   schema:affiliation ex:UPS .

# Define the supervisor Pierre
ex:Pierre a schema:Person ;
   ex:phdIn ex:CS ;
   schema:affiliation ex:IPP .

# Define the supervisor Pierre
ex:Anne a schema:Person ;
   ex:phdIn ex:Mathematics ;
   schema:affiliation ex:UPS .

# Define the university UPS
ex:UPS a ex:ResearchOrganization ;
   schema:name "Université Paris-Saclay"@fr ;
   schema:locatedin ex:France.

# Define the organization IPP
ex:IPP a schema:ResearchOrganization ;
   schema:name "Institut Polytechnique de Paris"@fr ;
    schema:locatedin ex:France .

ex:Michel a schema:Student .
 # Missing adviser -> will fail the sh:minCount requirement
```

Parsing took 0 ms. Validating the data took 1 ms.

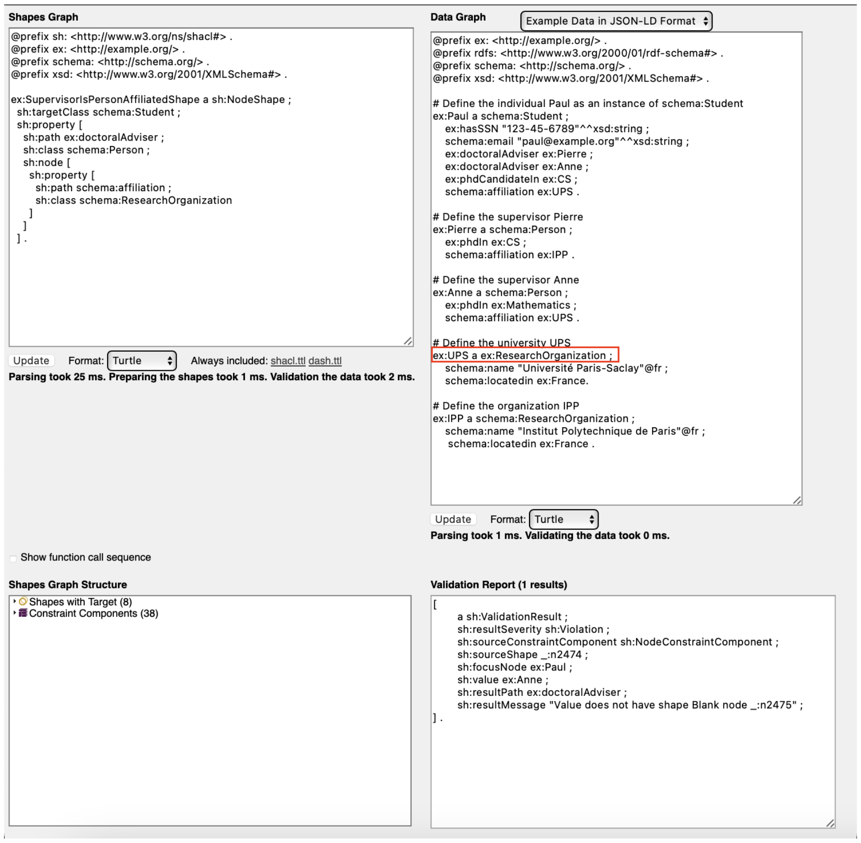
```
[
     a sh:ValidationResult ;
     sh:resultSeverity sh:Violation ;
     sh:sourceConstraintComponent sh:MinCountConstraintComponent ;
     sh:sourceShape _:n2136 ;
     sh:focusNode ex:Michel ;
     sh:resultPath ex:doctoralAdviser ;
     sh:resultMessage "Less than 1 values" ;
] .
[
     a sh:ValidationResult ;
     sh:resultSeverity sh:Violation ;
     sh:sourceConstraintComponent sh:MinCountConstraintComponent ;
     sh:sourceShape _:n2137 ;
     sh:focusNode ex:Michel ;
     sh:resultPath schema:affiliation ;
     sh:resultMessage "Less than 1 values" ;
] .
```

Figure 3: Check Query 7

8. **Exercise** Ensure that a student's supervisor is a person affiliated with a research unit **schema:ResearchOrganizati**

```
ex:SupervisorIsPersonAffiliatedShape a sh:NodeShape ;
  sh:targetClass schema:Student ;
  sh:property [
    sh:path ex:doctoralAdviser ;
    sh:class schema:Person ;
    sh:node [
      sh:property [
        sh:path schema:affiliation ;
        sh:class schema:ResearchOrganization
      ]
    ]
  ] .
```

In this query, as we are forcing to check that the supervisor of the student is a person affiliated with a research unit **schema:ResearchOrganization** (as IPP) and not **ex:ResearchOrganization** (as UPS), the SHACL processor finds a violation with one of the supervisor's affiliation.



Shapes Graph

```
@prefix sh: <http://www.w3.org/ns/shacl#> .
@prefix ex: <http://example.org/> .
@prefix schema: <http://schema.org/> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .

ex:SupervisorIsPersonAffiliatedShape a sh:NodeShape ;
 sh:targetClass schema:Student ;
 sh:property [
  sh:path ex:doctoralAdviser ;
  sh:class schema:Person ;
  sh:node [
   sh:property [
    sh:path schema:affiliation ;
    sh:class schema:ResearchOrganization
   ]
  ]
 ] .
```

Update Format: Turtle Always included: shacl.ttl dash.ttl

Parsing took 25 ms. Preparing the shapes took 1 ms. Validation the data took 2 ms.

Data Graph Example Data in JSON-LD Format

```
@prefix ex: <http://example.org/> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix schema: <http://schema.org/> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .

# Define the individual Paul as an instance of schema:Student
ex:Paul a schema:Student ;
   ex:hasSSN "123-45-6789"^^xsd:string ;
   schema:email "paul@example.org"^^xsd:string ;
   ex:doctoralAdviser ex:Pierre ;
   ex:doctoralAdviser ex:Anne ;
   ex:phdCandidateIn ex:CS ;
   schema:affiliation ex:UPS .

# Define the supervisor Pierre
ex:Pierre a schema:Person ;
   ex:phdIn ex:CS ;
   schema:affiliation ex:IPP .

# Define the supervisor Anne
ex:Anne a schema:Person ;
   ex:phdIn ex:Mathematics ;
   schema:affiliation ex:UPS .

# Define the university UPS
ex:UPS a ex:ResearchOrganization ;
   schema:name "Université Paris-Saclay"@fr ;
   schema:locatedin ex:France.

# Define the organization IPP
ex:IPP a schema:ResearchOrganization ;
   schema:name "Institut Polytechnique de Paris"@fr ;
    schema:locatedin ex:France .
```

Update Format: Turtle

Parsing took 1 ms. Validating the data took 0 ms.

Show function call sequence

Shapes Graph Structure

Shapes with Target (8)
Constraint Components (38)

Validation Report (1 results)

```
[
      a sh:ValidationResult ;
      sh:resultSeverity sh:Violation ;
      sh:sourceConstraintComponent sh:NodeConstraintComponent ;
      sh:sourceShape _:n2474 ;
      sh:focusNode ex:Paul ;
      sh:value ex:Anne ;
      sh:resultPath ex:doctoralAdviser ;
      sh:resultMessage "Value does not have shape Blank node _:n2475" ;
] .
```

Figure 4: **schema:ResearchOrganization** vs. **ex:ResearchOrganization**

9. **Exercise** Define a constraint to enforce that an **organization must have a name** (rdfs:label).

```
@prefix sh: <http://www.w3.org/ns/shacl#> .
@prefix ex: <http://example.org/> .
@prefix schema: <http://schema.org/> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .


ex:OrganizationShape a sh:NodeShape ;
  sh:targetClass schema:ResearchOrganization ;
  sh:property [
    sh:path rdfs:label ;
    sh:minCount 1
  ] .
```

In this query, make sure to bind **rdfs** prefix to corresponding link. Besides, notice that this rule will produce a violation as in the original data we don't have such property **rdfs:label**, instead we have **schema:name**.

10. **Exercise** Ensure that an organization has only **one denomination per language**.

```
ex:OrganizationLabelLangShape a sh:NodeShape ;
  sh:targetClass schema:ResearchOrganization ;
  sh:property [
    sh:path rdfs:label ;
    sh:uniqueLang true
  ] .
```

To visualize whether the rule works or not, we need to modify the original data, in particular, only the **IPP** object to ease what we want to show:

```
# Defining new version of the organization IPP
# Make sure ex:IPP is a schema:ResearchOrganization
ex:IPP a schema:ResearchOrganization ;
    rdfs:label "Institut Polytechnique de Paris"@fr ;
    # Same language -> will cause a violation
    rdfs:label "Institut Polytechnique de Paris (duplicate)"@fr ;
    schema:locatedin ex:France .
```

Another option to see the violation would be to just replace in the rule **rdfs:label** by **schema:name** and duplicate the name of any of the 2 institutions (use a different name). Something to notice is that by duplicating the name, **schema:name "Institut Polytechnique de Paris"@fr ;** and only changing the language (fr → en) still doesn't produce any violation.

11. **Exercise** All PhD advisors (see the predicate **ex:doctoralAdviser**) must hold a doctorate.

```
ex:PhDAdvisorMustHoldDoctorateShape a sh:NodeShape ;
  sh:targetObjectsOf ex:doctoralAdviser ;
  sh:property [
    sh:path ex:phdIn ;
    sh:minCount 1
  ] .
```

12. **Exercise All supervisors** of a PhD student **must be affiliated with the same university** as the student.

```
ex:SameAffiliationShape a sh:NodeShape ;
  sh:targetClass schema:Student ;
  sh:sparql [
    sh:message "All advisers must have the same affiliation as the student." ;
    sh:select """
      SELECT ?this
      WHERE {
        ?this schema:affiliation ?uni ;
              ex:doctoralAdviser ?adv .
        ?adv schema:affiliation ?advUni .
        FILTER(?advUni != ?uni)
      }
    """
  ] .
```

13. **Exercise At least one of the supervisors** of a PhD student must be **affiliated with the same university** as the student.

```
ex:AtLeastOneSameUniShape a sh:NodeShape ;
  sh:targetClass schema:Student ;
  sh:sparql [
    sh:message "At least one supervisor must share the same affiliation as the student." ;
    sh:select """
      SELECT ?this
      WHERE {
        ?this schema:affiliation ?uni .
        FILTER NOT EXISTS {
          ?this ex:doctoralAdviser ?adv .
          ?adv schema:affiliation ?uni .
        }
      }
    """
  ] .
```

14. **Exercise** PhD students enrolled in a German university cannot have more than one supervisor.

```
ex:GermanUniSupervisorLimitShape a sh:NodeShape ;
  sh:targetClass schema:Student ;
  sh:sparql [
    sh:message "PhD students in a German university cannot have more than one supervisor." ;
    sh:select """
      SELECT ?this
      WHERE {
        ?this schema:affiliation ?org ;
              ex:doctoralAdviser ?adv .
        ?org schema:locatedin ex:Germany .
      }
      GROUP BY ?this
      HAVING (COUNT(?adv) > 1)
    """
  ] .
```