# 5. SHACL: Shapes Constraint Language

# What is SHACL?

- Used to validate ontology data
- Ensures data follows rules

Example:

```
“Every Product must have a price”
“Every Person must have exactly one birth date”
```

# SPARQL vs SHACL: Different Purposes

| Term | Full Name | Purpose |
|---|---|---|
| **SPARQL** | SPARQL Protocol And RDF Query Language | **Query** (search/retrieve data) |
| **SHACL** | Shapes Constraint Language | **Validation** (check data quality/shape) |

Example:

```
SPARQL: "Find all Students with age > 20?"
SHACL:  "Does this Person have a name? Is age ≤ 150?"
```

## What SHACL Does

SHACL validates whether RDF data conforms to defined **"shapes"** (constraints):

In this example, a Person has a name of string type and age ≤ 150.

```
# Define "Person" shape
ex:PersonShape a sh:NodeShape ;
  sh:targetClass ex:Person ;
  sh:property [
    sh:path ex:name ;
    sh:minCount 1 ;      # Required
    sh:datatype xsd:string ;
  ] ;
  sh:property [
    sh:path ex:age ;
    sh:minCount 1 ;
    sh:maxInclusive 150 ; # Age ≤ 150
  ] .
```

## Running SHACL Validation

1. Protege SHACL Plugin installation requires many restrictions and may not work properly.
2. Free version of GraphDB does not support SHACL validation.
3. Use PySHACL library in Python for SHACL validation.

## Install & Run PySHACL

1. Install PySHACL:

```
pip install pyshacl rdflib
```

2. Run PySHACL CLI

```
pyshacl -df turtle food-safety-shacl.ttl
```

## 3. (Optional) Run PySHACL in Python

```
from rdflib import Graph

from pyshacl import validate
from rdflib import Graph

# 1. 데이터 + Shape
data_graph = Graph()
data_graph.parse("mushroom-ban-core.ttl", format="turtle")

# 2. SHACL
conforms, results_graph, results_text = validate(
    data_graph,
    shacl_graph=data_graph,  # 같은 파일에 Shape 포함
    inference='rdfs',
    abort_on_first=False,
)

# 3. 결과 출력
print("=" * 50)
print("SHACL Validation Results")
print("=" * 50)
print(f"\nConforms: {conforms}")
print("\nValidation Report:")
print(results_text)
```

## ex1: SHACL Validation with PySHACL

- In this example, we define a rule that **"Persons should not eat poisonous food."**
- We specify that a Person should eat only Apple & Banana to make the contraint simple.

# food-safety-shacl.ttl (RDF + SHACL)

```
@prefix ex: <http://example.org/food#> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix sh: <http://www.w3.org/ns/shacl#> .

# Classes
ex:Person a rdfs:Class .
ex:Mushroom a rdfs:Class .
ex:Food a rdfs:Class .

# Properties
ex:eats a rdf:Property .

# Instances
ex:Alice a ex:Person .
ex:PoisonMushroom a ex:Mushroom .
ex:Apple a ex:Food .
ex:Banana a ex:Food .

# Data
ex:Alice ex:eats ex:PoisonMushroom .

# SHACL Shape
ex:NoMushroomRule a sh:NodeShape ;
    sh:targetClass ex:Person ;
    sh:property [
        sh:path ex:eats ;
        sh:in ( ex:Apple ex:Banana )
    ] .
```

# mushroom-ban-complete.ttl (with more constraints)

````
We can add more constraints, such as:

1. Person must have a name (string)
2. Person should not be older than 150

```turtle
# Rule 1: Person can eat only Apple or Banana
ex:NoMushroomRule a sh:NodeShape ;
    sh:targetClass ex:Person ;
    sh:property [
        sh:path ex:eats ;
        sh:in ( ex:Apple ex:Banana ) ;
        sh:message "Person can only eat Apple or Banana!"
    ] .

# Rule 2: Person must have a name
ex:PersonNameRule a sh:NodeShape ;
    sh:targetClass ex:Person ;
    sh:property [
        sh:path ex:hasName ;
        sh:minCount 1 ;
        sh:message "Person must have a name!"
    ] .

# Rule 3: Age must be between 0 and 150
ex:PersonAgeRule a sh:NodeShape ;
    sh:targetClass ex:Person ;
    sh:property [
        sh:path ex:hasAge ;
        sh:datatype xsd:integer ;
        sh:minInclusive 0 ;
        sh:maxInclusive 150 ;
        sh:message "Age must be between 0 and 150!"
    ] .
````
````

## Validation Report

When we run the SHACL validation on `mushroom-ban-complete.ttl`, we get the following report:

1. False with two Errors

```
Conforms: False
Results (2):

```

## 2. Detailed Errors: 1. Person eats invalid food

```
Constraint Violation in InConstraintComponent (<http://www.w3.org/ns/shacl#InConstraintComponent>):
Severity: sh:Violation
Source Shape: [ sh:in ( ex:Apple ex:Banana ) ; sh:message Literal("Person can only eat Apple or Banana!") ; sh:path ex:eats ]
Focus Node: ex:Bob
Value Node: ex:PoisonMushroom
Result Path: ex:eats
Message: Person can only eat Apple or Banana!
```

## 3. Detailed Errors: 2. Person missing name

```
Constraint Violation in MinCountConstraintComponent (<http://www.w3.org/ns/shacl#MinCountConstraintComponent>):
Severity: sh:Violation
Source Shape: [ sh:message Literal("Person must have a name!") ; sh:minCount Literal("1", datatype=xsd:integer) ; sh:path ex:hasName ]
Focus Node: ex:Charlie
Result Path: ex:hasName
Result Path: ex:hasName
Message: Person must have a name!
```

# ex2: SHACL Validation with Separate Shape File

1. data.ttl (RDF Data)

```
ex:Alice a ex:Person ;
  ex:name "Alice" ;
  ex:age 30 .
```

2. shapes.ttl (SHACL Shapes)

```
ex:PersonShape a sh:NodeShape ;
  sh:targetClass ex:Person ;
  sh:property [
    sh:path ex:name ;
    sh:minCount 1 ;
    sh:datatype xsd:string ;
  ] .
```

3. Run using SHACL cli.

```
pyshacl -s shapes.ttl -df turtle data.ttl
```

4. (Optional) Run using PySHACL in Python

```
# 1. Data
data_graph = Graph()
data_graph.parse("data.ttl", format="turtle")

# 2. Shape graph load (separate file)
shapes_graph = Graph()
shapes_graph.parse("shapes.ttl", format="turtle")

# 3. SHACL
conforms, results_graph, results_text = validate(
    data_graph,
    shacl_graph=shapes_graph,  # 다른 그래프 사용!
    inference='rdfs',
    abort_on_first=False,
)
```

# ex3: Better Constraints for Food Safety

1. shapes.ttl (SHACL Shapes) with better rules

### Rule 1: Person cannot eat poisonous food

```
ex:NoPoisonousRule a sh:NodeShape ;
    sh:targetClass ex:Person ;
    sh:property [
        sh:path ex:eats ;
        sh:class ex:Food ;                    # must be Food
        sh:not [                               # and must not be
            sh:class ex:PoisonousFood          # PoisonousFood
        ] ;
        sh:message "Person cannot eat poisonous food!"
    ] .
```

## Rule 2: Person cannot eat mushrooms

```
ex:NoMushroomRule a sh:NodeShape ;
    sh:targetClass ex:Person ;
    sh:property [
        sh:path ex:eats ;
        sh:not [
            sh:class ex:Mushroom              # Mushroom should be avoided
        ] ;
        sh:message "Person cannot eat mushrooms!"
    ] .
```

2. data.ttl (RDF Data)

### PoisonousFood and Mushroom class hierarchy

```
ex:Person a rdfs:Class .
ex:Food a rdfs:Class .
ex:PoisonousFood a rdfs:Class ;
                 rdfs:subClassOf ex:Food .  # ← PoisonousFood Food
ex:Mushroom a rdfs:Class ;
            rdfs:subClassOf ex:PoisonousFood .  # ← Mushroom PoisonousFood
```

## Data instances

```
ex:Apple a ex:Food .
ex:Banana a ex:Food .
ex:PoisonMushroom a ex:Mushroom .
ex:ToxicBerry a ex:PoisonousFood .
```

Test inputs:

```
ex:Bob ex:eats ex:Apple .            # ✓ Valid
ex:Alice ex:eats ex:PoisonMushroom . # ❌ Invalid
ex:Alice ex:eats ex:ToxicBerry .     # ❌ Invalid
```

We expect three violations:We expect three violations:

1. PoisonMushroom → NoPoisonousRule violation ❌
2. PoisonMushroom → NoMushroomRule violation ❌ (same food, different rule)
3. ToxicBerry → NoPoisonousRule violation ❌

## Results

- Alice cannot eat poisonous food (PoisonMushroom).

```
Constraint Violation in NotConstraintComponent
(http://www.w3.org/ns/shacl#NotConstraintComponent):
 Severity: sh:Violation
 Source Shape: [
    sh:class ex:Food ; sh:message Literal("Person cannot eat poisonous food!") ;
    sh:not [ sh:class ex:PoisonousFood ] ; sh:path ex:eats ]
 Focus Node: ex:Alice
 Value Node: ex:PoisonMushroom
 Result Path: ex:eats
 Message: Person cannot eat poisonous food!
```

- Alice cannot eat poisonous food (ToxicBerry).

```
Constraint Violation in NotConstraintComponent
(http://www.w3.org/ns/shacl#NotConstraintComponent):
 Severity: sh:Violation
 Source Shape: [
    sh:class ex:Food ; sh:message Literal("Person cannot eat poisonous food!") ;
    sh:not [ sh:class ex:PoisonousFood ] ; sh:path ex:eats ]
 Focus Node: ex:Alice
 Value Node: ex:ToxicBerry
 Result Path: ex:eats
 Message: Person cannot eat poisonous food!
```

- Alice cannot eat PoisonMushroom (explicitly).

```
Constraint Violation in NotConstraintComponent
(http://www.w3.org/ns/shacl#NotConstraintComponent):
 Severity: sh:Violation
 Source Shape: [
    sh:message Literal("Person cannot eat mushrooms!") ;
    sh:not [ sh:class ex:Mushroom ] ; sh:path ex:eats ]
 Focus Node: ex:Alice
 Value Node: ex:PoisonMushroom
 Result Path: ex:eats
 Message: Person cannot eat mushrooms!
```