Shape Constraint Language (SHACL)

Cours « Recherche d'Information et Graphe de données »

Nicolas Delestre

INSA ROUEN NORMANDIE

CC BY NC SA

Monde ouvert vs Monde fermé 1 / 4

Soit le document RDF suivant

```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix schema: <http://schema.org/> .
@prefix prov: <http://www.w3.org/ns/prov#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@prefix insarouen: <https://data.insa-rouen.fr/> .
@prefix iti: <https://data.insa-rouen.fr/iti/> .
@prefix enseignant_iti: <https://data.insa-rouen.fr/iti/person/> .
@prefix cours_iti: <https://data.insa-rouen.fr/iti/cours/> .
@prefix litis: <https://data.insa-rouen.fr/litis/> .

enseignant_iti:NicolasDelestre
    rdf:type schema:Person ;
    schema:name "Nicolas Delestre" ;
    schema:jobTitle "enseignant"@fr ;
    schema:worksFor insarouen:etablissement ;
    schema:affiliation insarouen:etablissement ;
    schema:memberOf iti:departement ;
    schema:memberOf litis:laboratoire ;
    schema:image <https://data.insa-rouen.fr/iti/person/NicolasDelestre/photo.jpg> ;
    schema:teaches cours_iti:algo .
    schema:teaches cours_iti:python .
    schema:teaches cours_iti:compil .
    schema:teaches cours_iti:i4 .
    schema:teaches cours_iti:rigraphe .
```

BY NC SA

Monde ouvert vs Monde fermé 2 / 4

Paradigme du Monde fermé

- Dans un monde fermé l'absence d'une information signifie que l'information est fausse, qu'elle n'existe pas
- C'est par exemple le cas des bases de données relationnelles ou du langage Prolog

Paradigme du Monde ouvert

- Dans un monde ouvert l'absence d'une information ne signifie pas que l'information est fausse
- Dans ce paradigme, pour qu'une information soit fausse, il faut l'affirmer

INSA ROUEN NORMANDIE

BY NC SA

Monde ouvert vs Monde fermé 3 / 4

RDF est un monde ouvert

- Dans le Web des données, le fait qu'une information ne soit pas présente dans un graphe RDF ne signifie pas que l'information est fausse, elle peut être présente dans un autre graphe RDF

Par exemple

- Le fait qu'il n'y ait pas de numéro de téléphone parmi les propriétés qui décrivent enseignant_iti:NicolasDelestre ne signifie pas que Nicolas Delestre n'a pas de numéro de téléphone
- Il se peut qu'un autre graphe, celui de l'annuaire de l'INSA Rouen Normandie par exemple, publie cette information

INSA ROUEN NORMANDIE

CC BY NC SA

Monde ouvert vs Monde fermé 4 / 4

En pratique

- Le concepteur d'un schéma RDF veut quelques fois indiquer qu'une propriété, dans un document ou une base RDF, est obligatoire pour décrire une ressource d'une certaine classe
- Le consommateur de triplés RDF veut souvent savoir si l'absence d'une propriété dans un graphe RDF signifie :
  - que la ressource n'a pas cette propriété
  - qu'elle est peut être définie ailleurs

Mais

- RDFS n'oblige pas à l'utilisation d'une propriété pour les instances d'une classe
- RDF ne propose pas de négation de propriété

CC BY NC SA

SHACL : Shape Constraint Language

SHACL

- SHACL est un langage de validation de données RDF exprimé en RDF
- Il permet de définir des contraintes sur les données RDF pour s'assurer qu'elles respectent un certain schéma

Documents du W3C

- SHACL – Shapes Constraint Language (1.0, 2017) : définition du langage
- SHACL Core – SHACL Core Vocabulary (1.0, 2017) : vocabulaire de base pour définir des contraintes
- SHACL-SPARQL – SHACL SPARQL Extension Vocabulary (1.0, 2017) : vocabulaire pour définir des contraintes à l'aide de requêtes SPARQL
- SHACL Advanced Features – SHACL Advanced Features Vocabulary (1.0, 2017) : vocabulaire pour définir des contraintes avancées

En 2026

version 1.2 (pas de 1.1 ? ! ?) en cours de validation (*Working draft*)

BY NC SA

Vocabulaire

Forme (*shape*)

- Une forme est une ressource (au sens RDF, appelé aussi nœud dans le vocabulaire SHACL) qui décrit les contraintes à respecter pour les ressources d'une certaine classe
- Deux types de formes :
  - Forme de nœud (*node shape*) : ressource qui décrit les contraintes à respecter pour une ressource ou les ressources d'une certaine classe
  - Forme de propriété (*property shape*) : ressource qui décrit les contraintes à respecter pour les propriétés d'une certaine classe

Préfixes utilisés dans les exemples

```
@prefix sh:      <http://www.w3.org/ns/shacl#> .
@prefix rdf:     <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs:    <http://www.w3.org/2000/01/rdf-schema#> .
@prefix xsd:     <http://www.w3.org/2001/XMLSchema#> .
@prefix schema: <https://schema.org/> .
@prefix iti_regles: <https://data.insa-rouen.fr/iti/regles> .
```

Ex : forme de nœud et de propriété

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

Forme de nœud 1 / 2

- La ressource est une (rdf:type) sh:NodeShape
- Elle peut s'appliquer (*target*) :
  - à une classe : sh:targetClass
  - à une instance : sh:targetNode
  - au sujet d'une propriété : sh:targetSubjectsOf
  - au objet d'une propriété : sh:targetObjectsOf

Contraintes

Contrainte	Signification
sh:closed	Interdit les propriétés non explicitement définies pour le nœud
sh:ignoredProperties	Liste des propriétés autorisées même en mode fermé
sh:property	Associe des *PropertyShapes* au *NodeShape*
sh:sparql	Définit une contrainte SPARQL au niveau du nœud
sh:rule	Définit une règle d'inférence appliquée au nœud (SHACL-AF)

cc BY NC SA

Forme de nœud 2 / 2

Exemple

```
_:PersonShape
  a sh:NodeShape ;
  sh:targetClass ex:Person ;
  sh:closed true ;
  sh:ignoredProperties ( rdf:type ) ;
  sh:property [
    sh:path ex:name ;
    sh:minCount 1 ;
  ] .
```

Signification

- Chaque ressource de type ex:Person doit avoir au moins une propriété ex:name
- Seule la propriété définie (ex:name) est autorisée
- La propriété de type (rdf:type) est tout de même autorisée

INSA ROUEN NORMANDIE

CC BY NC SA

Forme de propriété

- La ressource est une sh:PropertyShape
- Elle doit avoir au moins une contrainte de chemin (sh:path)
- Dans le vocabulaire SHACL, l'objet du ou des chemins se nomme « valeurs » de la forme de propriété

INSA ROUEN NORMANDIE

CC BY NC SA

Contraintes 1 / 4

Sur le type de la valeur

Contrainte	Signification
sh:class	Chaque valeur de la propriété doit être une instance de la classe indiquée
sh:datatype	Chaque valeur littérale de la propriété doit avoir le type de données indiqué
sh:nodeKind	Chaque valeur de la propriété doit être d'une nature donnée (IRI, littéral, nœud anonyme, etc.)

Sur les cardinalités

Contrainte	Signification
sh:minCount	Le nombre de valeurs de la propriété doit être au moins égal à ce minimum
sh:maxCount	Le nombre de valeurs de la propriété doit être au plus égal à ce maximum

INSA ROUEN NORMANDIE

cc BY NC SA

Contraintes 2 / 4

Sur la cardinalité du range

Contrainte	Signification
sh:minExclusive	Chaque valeur de la propriété doit être strictement supérieure à la borne donnée
sh:minInclusive	Chaque valeur de la propriété doit être supérieure ou égale à la borne donnée
sh:maxExclusive	Chaque valeur de la propriété doit être strictement inférieure à la borne donnée
sh:maxInclusive	Chaque valeur de la propriété doit être inférieure ou égale à la borne donnée

Sur les chaîne de caractères

Contrainte	Signification
sh:minLength	Chaque valeur textuelle de la propriété doit avoir une longueur minimale donnée
sh:maxLength	Chaque valeur textuelle de la propriété doit avoir une longueur maximale donnée
sh:pattern	Chaque valeur textuelle de la propriété doit correspondre à l'expression rationnelle donnée
sh:languageIn	Chaque littéral avec balise de langue doit avoir une langue appartenant à la liste donnée
sh:uniqueLang	Il ne doit pas y avoir deux valeurs de la propriété avec la même balise de langue

BY NC SA

Contraintes 3 / 4

Sur les valeurs de la propriété

Contrainte	Signification
sh:equals	Les valeurs de la propriété doivent être les mêmes que celles d'une autre propriété
sh:disjoint	Les valeurs de la propriété ne doivent pas apparaître parmi celles d'une autre propriété
sh:lessThan	Chaque valeur de la propriété doit être strictement inférieure à la valeur correspondante d'une autre propriété
sh:lessThanOrEquals	Chaque valeur de la propriété doit être inférieure ou égale à la valeur correspondante d'une autre propriété

Sur la composition logique de contrainte

Contrainte	Signification
sh:and	Les valeurs de la propriété doivent satisfaire toutes les contraintes d'une liste donnée
sh:or	Les valeurs de la propriété doivent satisfaire au moins une des contraintes d'une liste donnée
sh:xone	Les valeurs de la propriété doivent satisfaire exactement une des contraintes d'une liste donnée
sh:not	Les valeurs de la propriété ne doivent pas satisfaire la contrainte donnée

BY NC SA

Contraintes 4 / 4

Sur les contraintes de chemin

Contrainte	Signification
sh:node	Référence une autre forme de nœud pour contraindre les valeurs de la propriété
sh:qualifiedValueShape	Chaque valeur de la propriété doit satisfaire une forme de nœud donnée
sh:qualifiedMinCount	Le nombre de valeurs de la propriété qui satisfont une forme donnée doit être au moins égal à ce minimum
sh:qualifiedMaxCount	Le nombre de valeurs de la propriété qui satisfont une forme donnée doit être au plus égal à ce maximum

Autres

Contrainte	Signification
sh:hasValue	La propriété doit avoir au moins une valeur égale à la valeur indiquée
sh:in	Chaque valeur de la propriété doit appartenir à la liste fournie

ROUEN NORMANDIE

CC BY NC SA

Exemple de règles 1 / 3

Une personne doit avoir un nom et si elle travaille elle a un poste

```
iti_regles:ReglesPourUnePersonne
    a sh:NodeShape ;
    sh:targetClass schema:Person ;

    sh:property [
        sh:path schema:name ;
        sh:minCount 1 ;
        sh:datatype xsd:string ;
    ] ;

    sh:or (
        [
            sh:property [
                sh:path schema:worksFor ;
                sh:maxCount 0 ;
            ]
        ]
        [
            sh:property [
                sh:path schema:worksFor ;
                sh:minCount 1 ;
                sh:class schema:
                    CollegeOrUniversity ;
            ] ;
            sh:property [
                sh:path schema:jobTitle ;
                sh:minCount 1 ;
            ] ;
            sh:property [
                sh:path schema:affiliation ;
                sh:minCount 1 ;
                sh:class schema:
                    CollegeOrUniversity ;
            ]
        ]
    ) .
```

ROUEN NORMANDIE

CC BY NC SA

Exemple de règles 2 / 3

Décomposer les formes de nœuds ⇒ deux formes de nœuds

```
iti_regles:ReglesPourUnePersonne
    a sh:NodeShape ;
    sh:targetClass schema:Person ;
    sh:property [
        sh:path schema:name ;
        sh:minCount 1 ;
        sh:datatype xsd:string ;
    ] .
```

```
iti_regles:ReglesPourUnEmploye
    a sh:NodeShape ;
    sh:targetClass schema:Person ;
    sh:or (
        [
            sh:property [
                sh:path schema:worksFor ;
                sh:maxCount 0 ;
            ]
        ]
        [
            sh:property [
                sh:path schema:worksFor ;
                sh:minCount 1 ;
            ] ;
            sh:property [
                sh:path schema:jobTitle ;
                sh:minCount 1 ;
            ] ;
            sh:property [
                sh:path schema:affiliation ;
                sh:minCount 1 ;
            ]
        ]
    ) .
```

Exemple de règles 3 / 3

Séparer règles des contraintes métiers

```
iti_regles:AuMoinsUnNom
    a sh:NodeShape ;
    sh:property [
        sh:path schema:name ;
        sh:minCount 1 ;
    ] .

iti_regles:ReglesPourUnePersonne
    a sh:NodeShape ;
    sh:targetClass schema:Person ;
    sh:and ( iti:AuMoinsUnNom iti:
         SiEmployeAlorsQualificationDuPoste ) .
```

```
iti_regles:SiEmployeAlorsQualificationDuPoste
    a sh:NodeShape ;
    sh:or (
        [
            sh:property [
                sh:path schema:worksFor ;
                sh:maxCount 0 ;
            ]
        ]
        [
            sh:property [
                sh:path schema:worksFor ;
                sh:minCount 1 ;
            ] ;
            sh:property [
                sh:path schema:jobTitle ;
                sh:minCount 1 ;
            ] ;
        ]
    ) .
```

CC BY NC SA

SHACL-SPARQL

Pourquoi utiliser SHACL-SPARQL ?

- SHACL Core couvre les contraintes les plus courantes : cardinalités, types, listes de valeurs, comparaisons simples, etc
- Certaines contraintes métiers sont parfois difficiles à exprimer uniquement avec le vocabulaire de base
- SHACL-SPARQL permet d'écrire des contraintes personnalisées sous forme de requêtes SPARQL

Principe

- Pour chaque nœud ciblé, la requête recherche les cas qui violent la règle
- Si la requête retourne une solution, alors la contrainte n'est pas respectée
- Le rapport de validation peut associer un message explicite à chaque violation

cc BY NC SA

Structure d'une contrainte SHACL-SPARQL 1 / 2

Syntaxe

```
ex:LaRegle
    a sh:NodeShape ;
    sh:targetClass ex:MaClasse ;
    sh:sparql [
        a sh:SPARQLConstraint ;
        sh:message "Message affiche si la contrainte est violee"@fr ;
        sh:select """
            SELECT $this
            WHERE {
                ... motif qui caracterise une violation ...
            }
        """ ;
    ] .
```

CC BY NC SA

Structure d'une contrainte SHACL-SPARQL 2 / 2

Remarques

- $this désigne le nœud courant validé
- La requête ne décrit pas les cas corrects mais les cas en erreur
- Il est possible d'utiliser :
  - sh:prefixes : pour ajouter des préfixes (instance de `sh:PrefixDeclaration`) utilisés dans requête SPARQL, par exemple :

```
ex:PrefixeSchemaOrg a sh:PrefixDeclaration ;
  sh:declare [
    sh:prefix "schema" ;
    sh:namespace "http://schema.org/"^^xsd:anyURI ;
  ] .
```

  - sh:severity : pour indiquer le niveau de gravité de la violation (sh:Violation, sh:Warning ou sh:Info)
  - plusieurs sh:sparql dans un même sh:NodeShape

BY NC SA

Exemple SHACL-SPARQL 1 / 2

Une date de fin ne doit pas être antérieure à la date de début

```
ex:ProjetDoitAvoirUneDateDeFinPosterieureADateDebut
    a sh:NodeShape ;
    sh:targetClass ex:Projet ;
    sh:sparql [
        a sh:SPARQLConstraint ;
        sh:message "La date de fin doit etre postérieure ou égale a la date de dé
            but"@fr ;
        sh:select """
            SELECT $this
            WHERE {
                $this ex:dateDebut ?debut ;
                      ex:dateFin ?fin .
                FILTER (?fin < ?debut)
            }
        """ ;
    ] .
```

cc BY NC SA

Exemple SHACL-SPARQL 2 / 2

Un membre du département ITI doit avoir un email INSA

```
iti_regles:MembreItiAUnEmailINSA
    a sh:NodeShape ;
    sh:targetClass schema:Person ;
    sh:sparql [
        a sh:SPARQLConstraint ;
        sh:message "Une personne membre du departement ITI doit avoir un courriel
            @insa-rouen.fr"@fr ;
        sh:select """
            SELECT $this
            WHERE {
                $this schema:memberOf iti:departement .
                FILTER NOT EXISTS {
                    $this schema:email ?mail .
                    FILTER regex(str(?mail), "@insa-rouen\\.fr$", "i")
                }
            }
        """ ;
    ] .
```

SHACL Rules

Validation vs inférence

- SHACL est d'abord conçu pour **valider** un graphe RDF
- SHACL-AF (*Advanced Features*) ajoute aussi des **règles** permettant d'enrichir un graphe par inférence locale
- Une règle n'exprime pas directement une erreur : elle produit de nouveaux triplets à partir de motifs déjà présents

Deux grandes familles

- sh:TripleRule : la règle génère un triplet *sujet-prédicat-objet* ;
- sh:SPARQLRule : la règle utilise une requête SPARQL CONSTRUCT

INSA ROUEN NORMANDIE

CC BY NC SA

Exemple de sh:TripleRule

Inférer qu'une personne ayant schema:teaches est un enseignant au sens de Wikidata

```
iti_regle:UnePersonneQuiEnseigneEstUnEnseignant
    a sh:NodeShape ;
    sh:targetClass schema:Person ;

    sh:rule [
        a sh:TripleRule ;
        sh:subject sh:this ;
        sh:predicate wdt:P31 ; # instance de
        sh:object wd:Q37226 ; # teacher
        sh:condition [
            sh:path schema:teaches ;
            sh:minCount 1 ;
        ] ;
    ] .
```

ROUEN NORMANDIE

CC BY NC SA

Exemple de sh:SPARQLRule

Inférer qu'un membre d'ITI appartenant à un laboratoire est aussi un chercheur

```
iti_rule:EnseignantQuiEstAussiChercheur
    a sh:NodeShape ;
    sh:targetClass schema:Person ;

    sh:rule [
        a sh:SPARQLRule ;
        sh:construct """
            CONSTRUCT {
                $this wdt:P31 wd:Q1650915 .
            }
            WHERE {
                $this schema:memberOf iti:
                    departement .
                $this schema:memberOf ?lab .
                ?lab a schema:
                    ResearchOrganization .
            }
        """ ;
    ] .
```

```
ex:ResearcherRuleShape
    a sh:NodeShape ;
    sh:targetClass schema:Person ;

    sh:rule [
        a sh:TripleRule ;
        sh:subject sh:this ;
        sh:predicate wdt:P31 ;
        sh:object wd:Q1650915 ;

        sh:condition [
            sh:and (

                # condition 1 : membre du dé
                    partement ITI
                [
                    sh:path schema:memberOf ;
                    sh:hasValue iti:departement ;
                ]

                # condition 2 : membre d'au moins
                    un laboratoire
                [
                    sh:path schema:memberOf ;
                    sh:qualifiedValueShape [
                        sh:class schema:
                            ResearchOrganization
```

Deux outils très utiles

PySHACL

- Validateur SHACL écrit en Python
- Pratique pour l'automatisation, les tests et l'intégration dans une chaîne de traitement de données
- Peut exécuter SHACL Core, SHACL-SPARQL et plusieurs fonctionnalités avancées (mais pas toutes)

TopBraid SHACL API et TopBraid

- TopBraid SHACL API : implémentation open source (Java) de SHACL utilisant Apache Jena
- TopBraid (payant) : environnement graphique orienté modélisation et validation de graphes RDF, qui facilite l'édition des shapes, l'exécution des validations et l'inspection des rapports

BY NC SA

Utiliser PySHACL en ligne de commande

Exemple minimal

```
pyshacl \
    -s shapes.ttl \
    -m \
    -i rdfs \
    donnees.ttl
```

Lecture des options

- -s : fichier des shapes
- -m : produit aussi le rapport RDF de validation
- -i rdfs : active une étape d'inférence RDFS avant validation

INSA ROUEN NORMANDIE

CC BY NC SA

Utiliser PySHACL en Python

Script Python minimal

```python
from pyshacl import validate

conforms, report_graph, report_text = validate(
    data_graph="donnees.ttl",
    shacl_graph="shapes.ttl",
    inference="rdfs",
    advanced=True,
)

print(conforms)
print(report_text)
```

Cas d'usage

tests unitaires, validation avant publication, intégration continue, génération de rapports

INSA ROUEN NORMANDIE

CC BY NC SA

Conclusion

Ce que nous avons vu

- SHACL permet d'exprimer un schéma de validation pour des graphes RDF
- SHACL Core couvre déjà la majorité des contraintes usuelles
- SHACL-SPARQL étend ce socle pour les contraintes métiers complexes
- SHACL Rules permettent d'enrichir un graphe avant ou pendant la validation
- PySHACL et TopBraid implémentent toute ou parties de SHACL

INSA ROUEN NORMANDIE

cc BY NC SA