

Recherche d'information et représentation des connaissances

Interrogation sémantique en SPARQL

Ollivier Haemmerlé
Nathalie Hernandez
Cassia Trojahn dos Santos
Camille Pradel

Introduction

Rappel : le Web Sémantique

- Basé sur le Web actuel
- Interprétable à la fois par les humains et les machines
- Standards proposés par le W3C :



- RDF : description de ressources (URI et triplets)

SUJET

-Ressource
-Nœud vide

PREDICAT

-Ressource

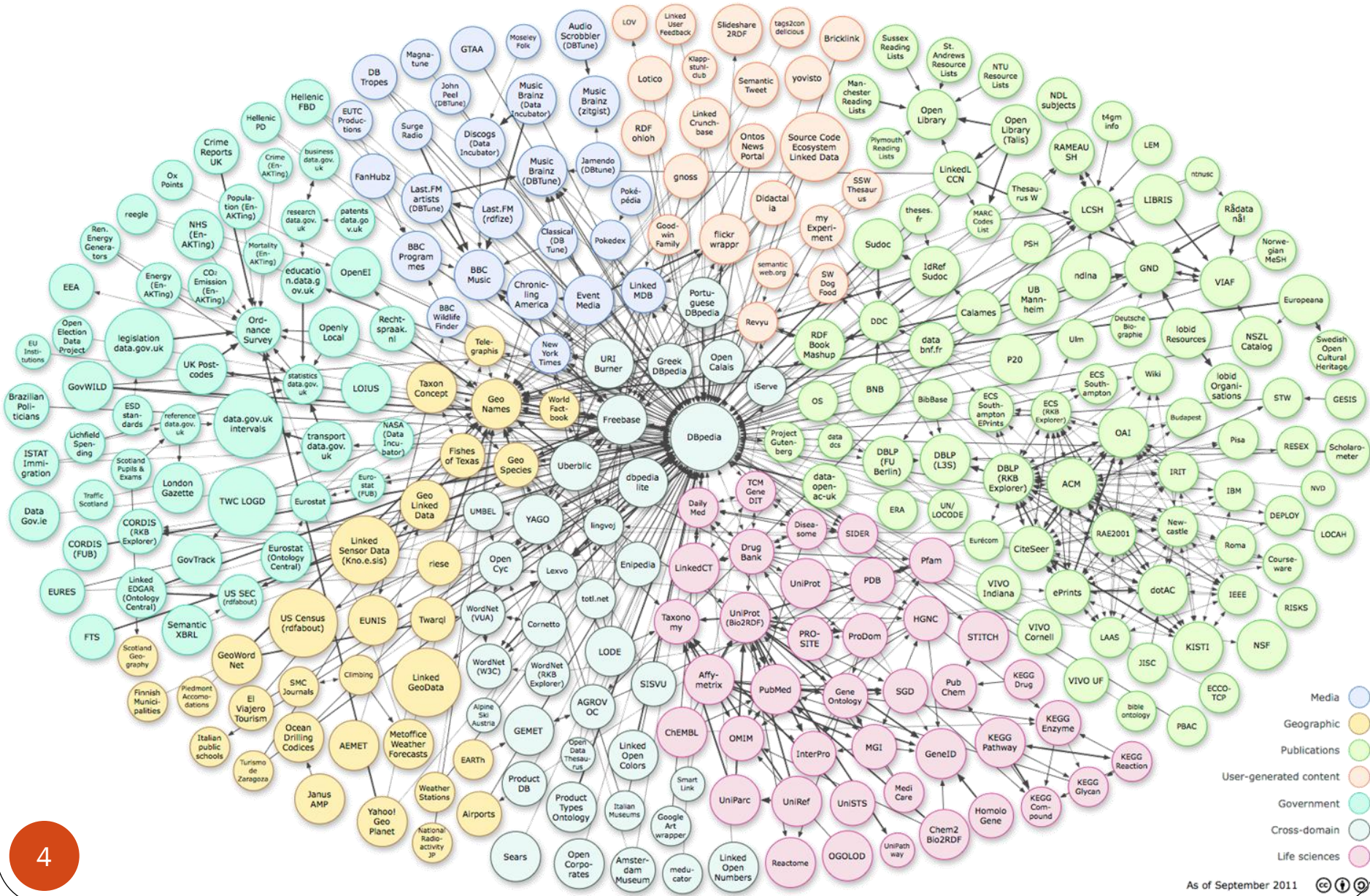
OBJET

-Ressource
-Nœud vide
-Littéral

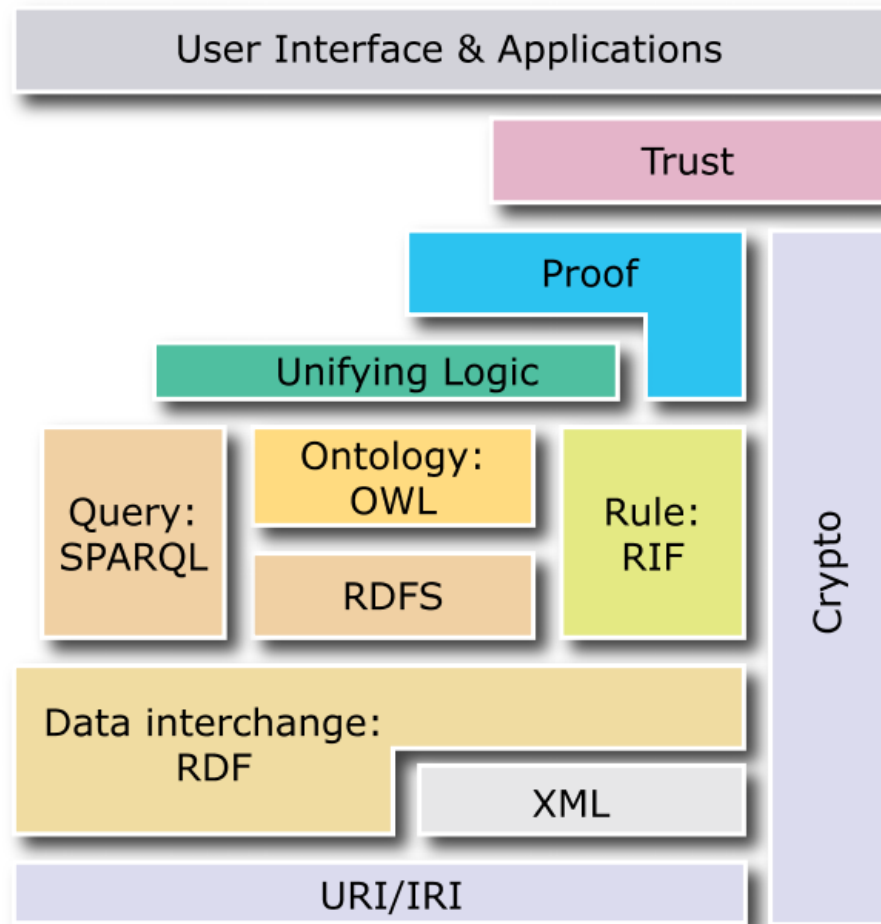


- OWL et RDFS : vocabulaires et ontologies

Le web de données



Layer cake



Historique de SPARQL

- 1999: RDF devient un standard
- 2004: création du RDF Data Access Working Group.
- Fin 2004: premier brouillon de la spécification SPARQL
- Début 2008: SPARQL devient une recommandation officielle
- 2009: fin du RDF-DAWG – création du SPARQL Working Group
- Fin 2009: premier brouillon de SPARQL 1.1
- Mars 2013: SPARQL 1.1 devient une recommandation officielle

Spécifications SPARQL

- SPARQL 1.0
 - **SPARQL Query Language for RDF**
 - SPARQL Protocol for RDF
 - SPARQL Query Results XML Format
- SPARQL 1.1
 - SPARQL Federation Extension
 - **SPARQL Update**
 - SPARQL Service Description
 - SPARQL Query Results JSON Format
 - SPARQL Graph Store HTTP Protocol
 - SPARQL Entailment Regimes
 - SPARQL New Features and Rationale

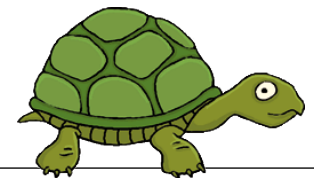
Introduction

- SPARQL = *SPARQL Protocol and RDF Query Language*
- Protocole d'échange de requêtes et de résultats
- Spécification du format des résultats
- Langage d'interrogation du web sémantique
- Avec SPARQL 1.1: langage de mise à jour

Langage d'interrogation du web sémantique

Syntaxe Turtle

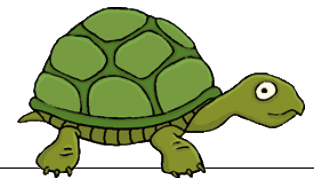
- Terse RDF Triple Language: format de sérialisation pour RDF
- Alternative au format RDF/XML
- Sous-ensemble de la notation N3
- Populaire car compacte et simple d'emploi
- Spécifications: <http://www.w3.org/TeamSubmission/turtle/>
 - Liste de triplets (« factorisables » avec « ; » et « , »)
 - URI: `<ur i / de / la / ressource>`
 - La déclaration de préfixes permet d'alléger ensuite l'écriture des triplets
 - Noms qualifiés: `pref i x : nom`
 - Nœuds vides (blank nodes): `_ : id`
 - Littéraux typés: `"valeur"^^type`



Syntaxe Turtle

- Un exemple:

```
@prefix dbp: <http://dbpedia.org/property/> .  
@prefix dbr: <http://dbpedia.org/resource/> .  
@prefix foaf: <http://xmlns.com/foaf/0.1/> .  
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .  
  
dbr:Albert_Einstein foaf:name "Albert Einstein" ;  
                    dbp:dateOfBirth "1879-03-14"^^xsd:date ;  
                    dbp:birthPlace dbr:Germany ;  
                    foaf:father _:father .  
_:father foaf:name "Hermann Einstein" .
```



Structure d'une requête SPARQL

PREFIX ... ← Déclare des préfixes pour les espaces de nom

SELECT ... ← {
Spécifie le format du résultat attendu
Autres possibilités: ASK, CONSTRUCT, DESCRIBE

FROM ... ← Indique le jeu de données cible

WHERE {...} ← {
Le motif recherché dans les solution
Notation Turtle, mais avec des variables

Premier exemple

- Exemple: la liste des classes présentes dans l'ontologie


PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>

PREFIX owl: <http://www.w3.org/2002/07/owl#>

SELECT ?class **WHERE**

```
{  
  ?class rdf:type owl:Class  
}
```

variable



Résultat

```
<?xml version='1.0' encoding='Cp1252'?>
```

```
[...]
```

```
<results>
```

```
<result>
```

```
<binding name='class'>
```

```
<uri>http://ontologies.alwaysdata.net/space#artefact</uri>
```

```
</binding>
```

```
</result>
```

```
<result>
```

```
<binding name='class'>
```

```
<uri>http://ontologies.alwaysdata.net/space#natural_object</uri>
```

```
</binding>
```

```
</result>
```

```
<result>
```

```
<binding name='class'>
```

```
<uri>http://ontologies.alwaysdata.net/space#celestial_body</uri>
```

```
</binding>
```

```
</result>
```

```
<result>
```

```
<binding name='class'>
```

← Sérialisation XML

Equivalent à un tableau

(ici d'une seule colonne car une seule variable était précisée après SELECT)

| ?class |
|---|
| http://ontologies.alwaysdata.net/space#artefact |
| http://ontologies.alwaysdata.net/space#natural_object |
| http://ontologies.alwaysdata.net/space#celestial_body |
| uri>http://ontologies.alwaysdata.net/space#star |
| uri>http://ontologies.alwaysdata.net/space#planet |

Deuxième exemple

- Exemple: la liste des satellites naturels du soleil avec leur rayon

PREFIX : <http://ontologies.alwaysdata.net/space#>

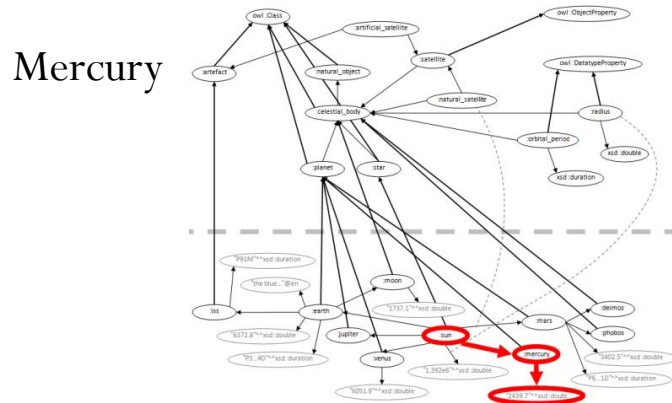
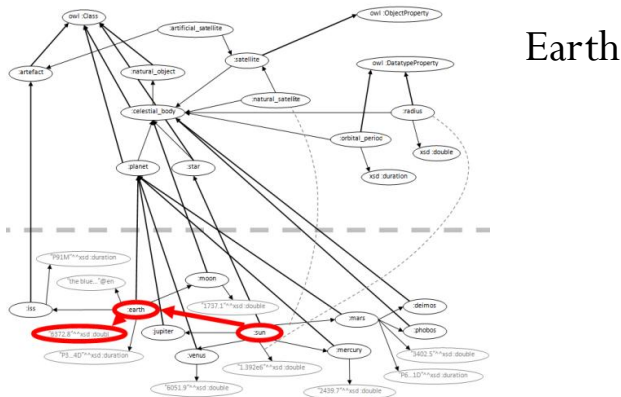
```
SELECT * WHERE
{
  :Sun :hasNaturalSatellite ?sat.
  ?sat :hasRadius ?rad.
}
```

Fait référence à toutes les variables de la requête



Deuxième exemple

- « Matching »: appariement du graphe de requête



Deuxième exemple

- Résultat

| ?sat | ?rad |
|--------|--------|
| :venus | 6051.9 |
| _:b0 | 2439.7 |
| _:b1 | 3402.5 |
| :earth | 6372.8 |

Troisième exemple

- Exemple: connaître tous les triplets présents dans la base de connaissances

```
SELECT * WHERE  
{  
  ?subject ?property ?object.  
}
```

Attention à la surcharge!

Types de données et fonctions

Littéraux

- SPARQL supporte les types définis dans XML Schema:
 - xsd:integer
 - xsd:decimal
 - xsd:float
 - xsd:double
 - xsd:string
 - xsd:boolean
 - xsd:dateTime

Littéraux et types de données

- Gestion du type et du tag de langage

```
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
```

```
@prefix ex: http://example.org/ .
```

```
ex:s1 ex:p "test" .
```

```
ex:s2 ex:p "test"^^xsd:string .
```

```
ex:s3 ex:p "test"@en .
```

```
ex:s4 ex:p "42"^^xsd:integer .
```

```
ex:s5 ex:p "test"^^<http://example.org/datatype1> .
```

```
{ ?subject ex:p "test" . }
```

```
{ ?subject ex:p "042"^^xsd:integer . }
```

```
{ ?subject ex:p 42 . }
```

Opérateurs et fonctions

- Opérateurs de comparaison: `=`, `<=`, `>=`, `!=`
- Opérateurs booléens: `&&`, `||`
- Opérations sur des termes RDF: `isIRI()`, `isBlank()`, `isLiteral()`, `isNumeric()`, `str()`, `lang()`, `datatype()`, `strdt()`, `strlang()`...
- Opérations sur les chaînes de caractères: `strlen()`, `substr()`, `ucase()`, `lcase()`, `strstarts()`, `strends()`, `contains()`, `regex()`, `encode_for_uri()`, `concat()`, `langMatches()`...
- Opérations sur les littéraux numériques: `abs`, `round`, `ceil`, `floor`, `rand`
 - Opérateurs arithmétiques classiques: `+`, `-`, `*`, `/`
- Opérations sur les dates: `year()`, `month()`, `day()`, `hours()`, `minutes()`, `seconds()`, `now()`, `timezone()`, `tz()`
- Fonctions de haschage: `MD5()`, `SHA1()`...
- Autres: `bound()`, `if()`, `coalesce()`, `not exists()`, `sameTerm()`, `in()`

Opérateurs sur des variables

- `isIRI` ou `isURI`
 - retourne true si le paramètre est de type IRI ; retourne false sinon
- `isBlank`
 - retourne true si le paramètre est un *nœud vide* ;
retourne false sinon
- `isLiteral`
 - retourne true si le paramètre est un *littéral* ;
retourne false sinon

Opérateurs sur des littéraux

- **lang**
 - retourne l'étiquette de langue du paramètre, le cas échéant ; retourne "" si le paramètre n'a pas d'étiquette de langue
- **datatype**
 - retourne l'URI du type de données du paramètre; retourne un type xsd:string si le paramètre est un littéral simple
- **langMatches**
 - retourne true si le premier argument correspond au deuxième argument selon le schéma de filtrage de base défini dans [RFC4647]
 - langMatches(lang(?label), “*”) retourne true ssi label a un tag de langue

Opérations sur les motifs de graphe

Motifs de graphe

- Motifs de graphe élémentaires
 - Ensemble de triplets
 - Délimité par des accolades
- Motifs de graphe de groupe (complexes)
 - Formés de plusieurs motifs
 - Permettent de réduire la portée de certaines conditions
 - Exemple: la période de révolution des satellites naturels

```
SELECT * WHERE
{
  { ?1 :hasNaturalSatellite ?sat .
    ?sat :hasOrbitalPeriod ?period .}
  {}
  ?sat rdf:type :Planet .
}
```

Ici sans intérêt; les motifs de graphe de groupe deviennent intéressants lorsque couplés avec d'autres fonctionnalités:
FILTER, OPTIONAL, UNION...

Filtrage de motifs

- Mot clef: **FILTER**
- Permet d'ajouter des contraintes sur des données
- Agit sur le motif dans lequel il se trouve
- Exemple: les planètes dont le rayon excède 6000km

```
SELECT * WHERE
```

```
{
```

```
  ?planet a :Planet .
```

```
  ?planet :hasRadius ?radius.
```

```
}
```

Raccourci pour `rdf:type`



| ?planet | ?radius |
|---------|---------|
| :venus | 6051.9 |
| _:b0 | 2439.7 |
| _:b1 | 3402.5 |
| :earth | 6372.8 |

Filtrage de motifs

- Mot clef: **FILTER**
- Permet d'ajouter des contraintes sur des données
- Agit sur le motif dans lequel il se trouve
- Exemple: les planètes dont le rayon excède 6000km

```
SELECT * WHERE
```

```
{
```

```
  ?planet a :Planet .
```

```
  ?planet :hasRadius ?radius.
```

```
  FILTER (?radius > 6000)
```

```
}
```

Raccourci pour `rdf:type`



| ?sat | ?rad |
|--------|--------|
| :venus | 6051.9 |
| :earth | 6372.8 |

Filtrage de motifs

- `isIRI` ou `isURI`
 - retourne true si le paramètre est de type IRI ; retourne false sinon

```
@prefix foaf:      <http://xmlns.com/foaf/0.1/> .

_:a  foaf:name      "Alice".
_:a  foaf:mbox       <mailto:alice@work.example> .

_:b  foaf:name      "Bob" .
_:b  foaf:mbox       "bob@work.example" .
```

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?name ?mbox
WHERE { ?x foaf:name ?name ;
         foaf:mbox ?mbox .
        FILTER isIRI(?mbox) }
```

| ?name | ?mbox |
|-------|-----------------------------|
| Alice | <mailto:alice@work.example> |

Filtrage de motifs

- lang
 - retourne l'étiquette de langue du paramètre, le cas échéant ;
retourne "" si le paramètre n'a pas d'étiquette de langue

```
@prefix foaf:      <http://xmlns.com/foaf/0.1/> .

_:a  foaf:name      "Robert"@EN.
_:a  foaf:name      "Roberto"@ES.
_:a  foaf:mbox       <mailto:bob@work.example> .
```

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?name ?mbox
WHERE { ?x foaf:name ?name ;
        foaf:mbox ?mbox .
        FILTER ( lang(?name) = "ES" ) }
```

| ?name | ?mbox |
|--------------|---------------------------|
| "Roberto"@ES | <mailto:bob@work.example> |

Filtrage de motifs

- **datatype**
 - retourne l'URI du type de données du paramètre; retourne un type `xsd:string` si le paramètre est un littéral simple

```
@prefix foaf:      <http://xmlns.com/foaf/0.1/> .
@prefix eg:        <http://biometrics.example/ns#> .
@prefix xsd:       <http://www.w3.org/2001/XMLSchema#> .

_:a  foaf:name      "Alice".
_:a  eg:shoeSize    "9.5"^^xsd:float .

_:b  foaf:name      "Bob".
_:b  eg:shoeSize    "42"^^xsd:integer .
```

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX xsd:  <http://www.w3.org/2001/XMLSchema#>
PREFIX eg:   <http://biometrics.example/ns#>
SELECT ?name ?shoeSize
  WHERE { ?x foaf:name ?name ; eg:shoeSize ?shoeSize .
         FILTER ( datatype(?shoeSize) = xsd:integer ) }
```

?name

"Bob"

?shoeSize

42

Filtrage de motifs

- langMatches
 - retourne true si le premier argument correspond au deuxième argument selon le schéma de filtrage de base défini dans [RFC4647]

```
@prefix dc:      <http://purl.org/dc/elements/1.1/> .

_:a  dc:title    "That Seventies Show"@en .
_:a  dc:title    "Cette série des années soixante-dix"@fr .
_:a  dc:title    "Cette série des années septante"@fr-BE .
_:b  dc:title    "Il Buono, il Bruto, il Cattivo" .
```

```
PREFIX dc: <http://purl.org/dc/elements/1.1/>
SELECT ?title
WHERE { ?x dc:title "That Seventies Show"@en ;
        dc:title ?title .
        FILTER langMatches( lang(?title), "FR" ) }
```

?title

"Cette série des années soixante-dix"@fr

"Cette série des années septante"@fr-BE

Filtrage de motifs

- **bound**
 - permet de vérifier si une variable est liée
 - **!bound** permet de vérifier si une variable n'est pas liée (utilisé pour la négation par défaut)

Filtrage de motifs

- **in**
- Permet de déterminer si une ressource appartient à une liste donnée
- Exemple: les satellites de la terre **et** du soleil

```
SELECT ?sat WHERE
{
  ?body :hasNaturalSatellite ?sat
  FILTER (?body in (:Earth, :Sun)).
}
```

- Existe aussi: **not in**

| sat |
|--|
| <http://ontologies.alwaysdata.net/space#Moon> |
| <http://ontologies.alwaysdata.net/space#Venus> |
| <http://ontologies.alwaysdata.net/space#Earth> |
| _.b0 |
| <http://ontologies.alwaysdata.net/space#Jupiter> |
| _.b1 |

Motifs de graphe optionnels

- Mot clef: **OPTIONAL**
- Permet de spécifier des motifs qui ne doivent pas obligatoirement apparaître dans le résultat
- Exemple: les satellites du soleil avec leur rayon

```
SELECT * WHERE
{
  :Sun :hasNaturalSatellite ?sat.
  ?sat :hasRadius ?rad.
}
```

| ?sat | ?rad |
|--------|--------|
| :venus | 6051.9 |
| _:2 | 2439.7 |
| :mars | 3402.5 |
| :earth | 6372.8 |

Motifs de graphe optionnels

- Mot clef: **OPTIONAL**
- Permet de spécifier des motifs qui ne doivent pas obligatoirement apparaître dans le résultat
- Exemple: les satellites du soleil avec leur rayon

```
SELECT * WHERE
```

```
{  
  :Sun :hasNaturalSatellite ?sat.  
  OPTIONAL {?sat :hasRadius ?rad. }  
}
```

| ?sat | ?rad |
|----------|--------|
| :venus | 6051.9 |
| _:2 | 2439.7 |
| :mars | 3402.5 |
| :earth | 6372.8 |
| :jupiter | |

Négation par défaut

- Utilise **OPTIONAL**, **FILTER** et **!bound**
- Exemple: les corps célestes qui n'ont pas de satellite (référéncé dans la base de connaissances)

```
SELECT ?body WHERE
{
  ?body a :CelestialBody.
  OPTIONAL { ?body :hasSatellite ?sat }
  FILTER ( !bound(?sat) )
}
```

```

SELECT * WHERE
{
  ?body a :CelestialBody.
  OPTIONAL { ?body :hasSatellite ?sat }
}

```



```

SELECT * WHERE
{
  ?body a :CelestialBody.
  OPTIONAL { ?body :hasSatellite ?sat }
  FILTER ( !bound(?sat) )
}

```



| body | sat |
|--|---|
| <http://ontologies.alwaysdata.net/space#Moon> | |
| <http://ontologies.alwaysdata.net/space#Deimos> | |
| <http://ontologies.alwaysdata.net/space#Phobos> | |
| <http://ontologies.alwaysdata.net/space#Earth> | <http://ontologies.alwaysdata.net/space#Moon> |
| <http://ontologies.alwaysdata.net/space#Earth> | <http://ontologies.alwaysdata.net/space#InternationalSpaceStation> |
| <http://ontologies.alwaysdata.net/space#Jupiter> | |
| <http://ontologies.alwaysdata.net/space#Venus> | |
| _ b0 | |
| _b1 | <http://ontologies.alwaysdata.net/space#Deimos> |
| _b1 | <http://ontologies.alwaysdata.net/space#Phobos> |
| <http://ontologies.alwaysdata.net/space#Sun> | <http://ontologies.alwaysdata.net/space#Earth> |
| <http://ontologies.alwaysdata.net/space#Sun> | <http://ontologies.alwaysdata.net/space#Jupiter> |
| <http://ontologies.alwaysdata.net/space#Sun> | <http://ontologies.alwaysdata.net/space#Venus> |
| <http://ontologies.alwaysdata.net/space#Sun> | _ b0 |
| <http://ontologies.alwaysdata.net/space#Sun> | _ b1 |
| <http://ontologies.alwaysdata.net/space#InternationalSpaceStation> | |

Négation par défaut

- Utile pour identifier des données manquantes.
- Exemple: les corps célestes dont le rayon n'est pas renseigné:

```
SELECT ?body WHERE
```

```
{
```

```
  ?body a :CelestialBody.
```

```
  OPTIONAL { ?body :hasRadius ?rad }
```

```
  FILTER ( !bound(?rad) )
```

```
}
```

| body |
|--|
| <http://ontologies.alwaysdata.net/space#Deimos> |
| <http://ontologies.alwaysdata.net/space#Phobos> |
| <http://ontologies.alwaysdata.net/space#Jupiter> |
| <http://ontologies.alwaysdata.net/space#InternationalSpaceStation> |

Négation par défaut

- Attention: l'utilisation n'est pas très naturelle
- Exemple: les corps célestes qui n'ont pas la lune comme satellite

```
SELECT ?body WHERE
{
  ?body a :CelestialBody.
  OPTIONAL {
    ?body :hasSatellite ?sat .
    FILTER ( sameterm(?sat, :Moon) )
  }
  FILTER ( !bound(?sat) )
}
```

```
SELECT * WHERE
{
  ?body :hasSatellite ?sat.
}
```



```
SELECT * WHERE
{
  ?body :hasSatellite ?sat.
  FILTER ( sameterm(?sat, :Moon) ).
}
```



| body | sat |
|---|---|
| <http://ontologies.alwaysdata.net/space#Earth> | <http://ontologies.alwaysdata.net/space#Moon> |
| _b0 | <http://ontologies.alwaysdata.net/space#Deimos> |
| _b0 | <http://ontologies.alwaysdata.net/space#Phobos> |
| <http://ontologies.alwaysdata.net/space#Sun> | <http://ontologies.alwaysdata.net/space#Earth> |
| <http://ontologies.alwaysdata.net/space#Sun> | <http://ontologies.alwaysdata.net/space#Jupiter> |
| <http://ontologies.alwaysdata.net/space#Sun> | <http://ontologies.alwaysdata.net/space#Venus> |
| <http://ontologies.alwaysdata.net/space#Sun> | _b1 |
| <http://ontologies.alwaysdata.net/space#Sun> | _b0 |
| <http://ontologies.alwaysdata.net/space#Earth> | <http://ontologies.alwaysdata.net/space#InternationalSpaceStation> |

```
SELECT * WHERE
{
  ?body :hasSatellite ?sat.
  FILTER ( sameterm(?sat, :Moon) ).
}
```



| body | sat |
|--|---|
| <http://ontologies.alwaysdata.net/space#Earth> | <http://ontologies.alwaysdata.net/space#Moon> |

```
SELECT * WHERE
{
  ?body a :CelestialBody.
}
```



| body |
|--|
| <http://ontologies.alwaysdata.net/space#Moon> |
| <http://ontologies.alwaysdata.net/space#Deimos> |
| <http://ontologies.alwaysdata.net/space#Phobos> |
| <http://ontologies.alwaysdata.net/space#Earth> |
| <http://ontologies.alwaysdata.net/space#Jupiter> |
| <http://ontologies.alwaysdata.net/space#Venus> |
| _.b0 |
| _.b1 |
| <http://ontologies.alwaysdata.net/space#Sun> |
| <http://ontologies.alwaysdata.net/space#InternationalSpaceStation> |

```
SELECT * WHERE
{
  ?body a :CelestialBody.
  OPTIONAL {
    ?body :hasSatellite ?sat .
    FILTER ( sameterm(?sat, :Moon) )
  }
}
```



| body | sat |
|--|---|
| <http://ontologies.alwaysdata.net/space#Moon> | |
| <http://ontologies.alwaysdata.net/space#Deimos> | |
| <http://ontologies.alwaysdata.net/space#Phobos> | |
| <http://ontologies.alwaysdata.net/space#Earth> | <http://ontologies.alwaysdata.net/space#Moon> |
| <http://ontologies.alwaysdata.net/space#Jupiter> | |
| <http://ontologies.alwaysdata.net/space#Venus> | |
| _.b0 | |
| _.b1 | |
| <http://ontologies.alwaysdata.net/space#Sun> | |
| <http://ontologies.alwaysdata.net/space#InternationalSpaceStation> | |

```

SELECT * WHERE
{
  ?body a :CelestialBody.
  OPTIONAL {
    ?body :hasSatellite ?sat .
    FILTER ( sameterm(?sat, :Moon) )
  }
}

```



```

SELECT * WHERE
{
  ?body a :CelestialBody.
  OPTIONAL {
    ?body :hasSatellite ?sat .
    FILTER ( sameterm(?sat, :Moon) )
  }
  FILTER ( !bound(?sat) )
}

```



| body | sat |
|--|--|
| <http://ontologies.alwaysdata.net/space#Moon> | |
| <http://ontologies.alwaysdata.net/space#Deimos> | |
| <http://ontologies.alwaysdata.net/space#Phobos> | |
| <http://ontologies.alwaysdata.net/space#Earth> | <http://ontologies.alwaysdata.net/space#Moon> |
| <http://ontologies.alwaysdata.net/space#Jupiter> | |
| <http://ontologies.alwaysdata.net/space#Venus> | |
| _:b0 | |
| _:b1 | |
| <http://ontologies.alwaysdata.net/space#Sun> | |
| <http://ontologies.alwaysdata.net/space#InternationalSpaceStation> | |

Négation

- Mot clef: **NOT EXISTS**
- Permet de spécifier un motif de graphe qui ne doit pas pouvoir être apparié au résultat
- Exemple: les corps célestes qui n'ont pas la lune comme satellite:

```
SELECT ?body WHERE
{
  ?body a :CelestialBody.
  NOT EXISTS { ?body :hasSatellite :Moon }
}
```


Motifs de graphe alternatifs

- Mot clef: **UNION**
- Permet de spécifier des motifs alternatifs
- Equivalent à la **disjonction** en logique
- Exemple 1: les satellites de la terre **ou** du soleil

```
SELECT ?sat WHERE
{
    { :Earth :hasSatellite ?sat }
    UNION { :Sun :hasSatellite ?sat }
}
```

Motifs de graphe alternatifs

- Mot clef: **UNION**
- Permet de spécifier des motifs alternatifs
- Equivalent à la **disjonction** en logique
- Exemple 2: les satellites de la terre **ou** du soleil avec deux variables:

```
SELECT * WHERE
{
    { :Earth :hasSatellite ?satE }
    UNION { :Sun :hasSatellite ?satS }
}
```

```

SELECT * WHERE
{
    { :Earth :hasSatellite ?satE }
    UNION { :Sun :hasSatellite ?satS }
}

```

| satE | satS |
|--|--|
| <http://ontologies.alwaysdata.net/space#Moon> | |
| <http://ontologies.alwaysdata.net/space#InternationalSpaceStation> | |
| | <http://ontologies.alwaysdata.net/space#Earth> |
| | <http://ontologies.alwaysdata.net/space#Jupiter> |
| | <http://ontologies.alwaysdata.net/space#Venus> |
| | _.b0 |
| | _.b1 |

Alternatives

- Simplification de l'union

```
SELECT ?name
WHERE {
    {?book rdfs:label ?name}
    UNION
    {?book dc:title ?name}
}
```



```
SELECT ?name
WHERE {
    ?book (dc:title|rdfs:label) ?name
}
```

UNION et OPTIONAL

- **UNION** et **OPTIONAL** sont associatifs à gauche
- Exemple: les satellites de la Terre **ou** de Mars et **éventuellement** leur rayon

```
SELECT * WHERE
{
  { :Earth :hasSatellite ?sat } UNION
  { :Sun :hasSatellite ?sat }
  OPTIONAL { ?sat :hasRadius ?rad }
}
```

UNION et OPTIONAL

- **UNION** et **OPTIONAL** sont associatifs à gauche
- Exemple: les satellites de la Terre **ou** de Mars et **éventuellement** leur rayon

```
SELECT * WHERE
{
  {[:Earth :hasSatellite ?sat] UNION
   [:Sun :hasSatellite ?sat]}
  OPTIONAL {?sat :hasRadius ?rad}
}
```

Projection

- Mot clef: **AS** ou **BIND AS**
- Permet d'enregistrer le résultat d'une expression dans une variable
- Exemple: les satellites du soleil avec leur périmètre

```
SELECT ?sat (?rad * 3.14 AS ?perim) WHERE
```

```
{
```

```
  :Sun :hasSatellite ?sat.
```

```
  ?sat :hasRadius ?rad.
```

```
}
```



```
SELECT ?sat ?perim WHERE
```

```
{
```

```
  :Sun :hasSatellite ?sat.
```

```
  ?sat :hasRadius ?rad.
```

```
  BIND (?rad * 3.14 AS ?perim)
```

```
}
```

```

SELECT ?sat (?rad * 3.14 AS ?perim)
WHERE
{
  :Sun :hasSatellite ?sat.
  ?sat :hasRadius ?rad.
}

```

≡

```

SELECT ?sat ?perim WHERE
{
  :Sun :hasSatellite ?sat.
  ?sat :hasRadius ?rad.
  BIND (?rad * 3.14 AS ?perim)
}

```



| sat | perim |
|--|---|
| <http://ontologies.alwaysdata.net/space#Earth> | "20010.592e0" ^^<http://www.w3.org/2001/XMLSchema#double> |
| <http://ontologies.alwaysdata.net/space#Venus> | "19002.966e0" ^^<http://www.w3.org/2001/XMLSchema#double> |
| _b0 | "10683.85e0" ^^<http://www.w3.org/2001/XMLSchema#double> |
| _b1 | "7660.657999999999e0" ^^<http://www.w3.org/2001/XMLSchema#double> |

Groupes et agrégats

Groupes et agrégats

- **GROUP BY** permet de regrouper des résultats
- **COUNT, SUM, MIN, MAX, AVG, SAMPLE, GROUP_CONCAT** permettent d'effectuer des opérations d'agrégats sur les groupes
- **HAVING** permet de filtrer les groupes

Les opérateurs d'agrégats

- **COUNT (?v)** permet d'obtenir le nombre fois que v est affectée dans l'ensemble
- **SUM (?v)** retourne la somme des valeurs prises par v
- **MIN (?v)** retourne la plus petite des valeurs prises par v
- **MAX (?v)** retourne la plus grande des valeurs prises par v
- **AVG (?v)** retourne la moyenne des valeurs prises par v
- **SAMPLE (?v)** retourne au hasard une des valeurs prises par la variable v
- **GROUP_CONCAT (?v, separator = " | ")** retourne les chaînes de caractères représentant l'ensemble des valeurs prises par v, concaténées et séparées par le séparateur spécifié

Groupes et agrégats

- Exemple 1: le nombre de satellites pour chaque corps céleste

```
SELECT ?planet COUNT(?sat)
WHERE {
    ?planet a :CelestialBody.
    ?planet :hasSatellite ?sat.
}
GROUP BY ?planet
```

| planet | .1 |
|--|--|
| <http://ontologies.alwaysdata.net/space#Sun> | "5" ^^<http://www.w3.org/2001/XMLSchema#integer> |
| _:b0 | "2" ^^<http://www.w3.org/2001/XMLSchema#integer> |
| <http://ontologies.alwaysdata.net/space#Earth> | "2" ^^<http://www.w3.org/2001/XMLSchema#integer> |

Groupes et agrégats

- Possibilité d'utiliser l'opérateur de projection **AS**
→ Plus grande lisibilité du résultat
- Exemple 2 : le nombre de satellites pour chaque corps céleste

```
SELECT ?planet (COUNT(?sat) AS ?nbSat)
WHERE {
    ?planet a :CelestialBody.
    ?planet :hasSatellite ?sat.
}
GROUP BY ?planet
```

| planet | nbSat |
|--|--|
| <http://ontologies.alwaysdata.net/space#Sun> | "5" ^^<http://www.w3.org/2001/XMLSchema#integer> |
| _:b0 | "2" ^^<http://www.w3.org/2001/XMLSchema#integer> |
| <http://ontologies.alwaysdata.net/space#Earth> | "2" ^^<http://www.w3.org/2001/XMLSchema#integer> |

Groupes et agrégats

- Remarque: l'ensemble des résultats non regroupés constitue également un groupe; il est donc possible de calculer des agrégats sur la requête entière sans GROUP BY
- Exemple 3 : compter les planètes répertoriées dans la base

```
SELECT COUNT(?planet)
WHERE {
    ?planet a :Planet.
}
```

| |
|--|
| .1 |
| "5" ^^<http://www.w3.org/2001/XMLSchema#integer> |

Groupes et agrégats

- Exemple 4 : le plus grand rayon pour une planète

```
SELECT (MAX(?rad) as ?maxRad)
WHERE {
  ?planet a :Planet.
  ?planet :hasRadius ?rad.
}
```

| maxRad |
|--|
| "6372.8" ^{^^} <http://www.w3.org/2001/XMLSchema#double> |

Groupes et agrégats

- Possibilité d'utiliser des sous-requêtes
- Exemple 5 : la plus grosse planète

```
SELECT ?planet
WHERE {
  {
    # obtenir le plus grand rayon
    SELECT (MAX(?rad) as ?maxRad)
    WHERE {
      ?planet a :Planet.
      ?planet :hasRadius ?rad.
    }
  }
  ?planet a :Planet.
  ?planet :hasRadius ?maxRad.
}
```

| planet |
|---|
| <http://ontologies.alwaysdata.net/space#Earth> |

Groupes et agrégats

- Exemple++ : le corps céleste ayant le plus de satellites

```
SELECT ?body
WHERE {
```

```
  #obtain max number of satellites for one body
```

```
  { SELECT (MAX(?nbSat) AS ?nbSatMax)
    WHERE
    {
      { SELECT (COUNT(?sat) as ?nbSat)
        WHERE
        {
          ?body a :CelestialBody.
          ?body :hasSatellite ?sat.
        } GROUP BY ?body
      }
    }
  }
```

```
  #obtain number of satellites for each body
```

```
  { SELECT ?body (COUNT(?sat) as ?nbSat)
    WHERE
    {
      ?body a :CelestialBody.
      ?body :hasSatellite ?sat.
    } GROUP BY ?body
  }
```

```
  #keep only the body/ies with maximum number of satellites
```

```
  FILTER (?nbSat = ?nbSatMax)
```

```
}
```

| body |
|---|
| <http://ontologies.alwaysdata.net/space#Sun> |

Groupes et agrégats

- **HAVING** agit sur les ensembles de solutions groupées de la même façon que **FILTER** agit sur les solutions individuelles
- Exemple : liste des corps célestes ayant plus de deux satellites

PREFIX : `<http://ontologies.alwaysdata.net/space#>`

```
SELECT ?body
WHERE {
    ?body a :CelestialBody.
    ?body :hasSatellite ?sat.
    ?body :hasRadius ?rad.
}
```

```
GROUP BY ?body
```


```
HAVING (COUNT(?sat) > 2)
```

| body |
|---|
| <code><http://ontologies.alwaysdata.net/space#Sun></code> |

Modificateurs

Modificateurs

- ORDER BY
- DISTINCT
- LIMIT
- OFFSET



Fonctionnent comme en SQL

ORDER BY

- Etablit l'ordre d'une séquence de solutions

- Exemple 1: les satellites du soleil par ordre lexic

```
SELECT ?sat  
WHERE { :sun :satellite ?sat }  
ORDER BY ?sat
```

?sat

:earth

:jupiter

:mars

:venus

_:2

- Exemple 2: les satellites du soleil par

```
SELECT ?sat  
WHERE { :sun :satellite ?sat }  
ORDER BY DESC(?sat)
```

?sat

_:2

:venus

:mars

:jupiter

:earth

DISTINCT

- Permet d'éliminer les solutions en double
- Exemple: les corps célestes ayant au moins un satellite

```
SELECT ?body WHERE  
{ ?body a :celestial_body  
  ?body :satellite ?sat }
```

| ?sat |
|--------|
| :earth |
| :sun |
| :sun |
| :sun |
| :sun |
| :sun |
| _:b0 |
| _:b0 |
| :earth |

DISTINCT

- Permet d'éliminer les solutions en double
- Exemple: les corps célestes ayant au moins un satellite

```
SELECT DISTINCT ?body WHERE  
{ ?body a :celestial_body  
  ?body :satellite ?sat }
```

?sat

:earth

:sun

:mars

LIMIT

- Place une limite supérieure au nombre de solutions retournées
- Exemple: les deux premiers satellites du soleil dans l'ordre lexicographique

```
SELECT ?sat  
WHERE { :sun :satellite ?sat }  
ORDER BY ?sat  
LIMIT 2
```

?sat

:earth

:jupiter

OFFSET

- Fait commencer les solutions générées après le nombre de solutions indiqué
- Exemple: les troisième et quatrième satellites du soleil dans l'ordre lexicographique

```
SELECT ?sat  
WHERE { :sun :satellite ?sat }  
ORDER BY ?sat  
LIMIT 2 OFFSET 2
```

?sat

:mars

:venus

- **Attention!** dangereux si l'on ne rend pas l'ordre prévisible avec **ORDER BY**

Formats de résultats

Formats de résultat

- SELECT
 - Liste des éléments correspondants au motif de requête
- CONSTRUCT
 - Construction d'un nouveau graphe RDF
- ASK
 - Recherche si la requête a un résultat ou non
- DESCRIBE
 - Décrit une ressource

CONSTRUCT

- **SELECT** → traitement séquentiel des résultats
- **CONSTRUCT** → exploitation de la structure des résultats et des relations entre les éléments
- Syntaxe:

PREFIX ...

FROM ...

CONSTRUCT {...}

← Template utilisé pour créer le document RDF du résultat

WHERE

{...}

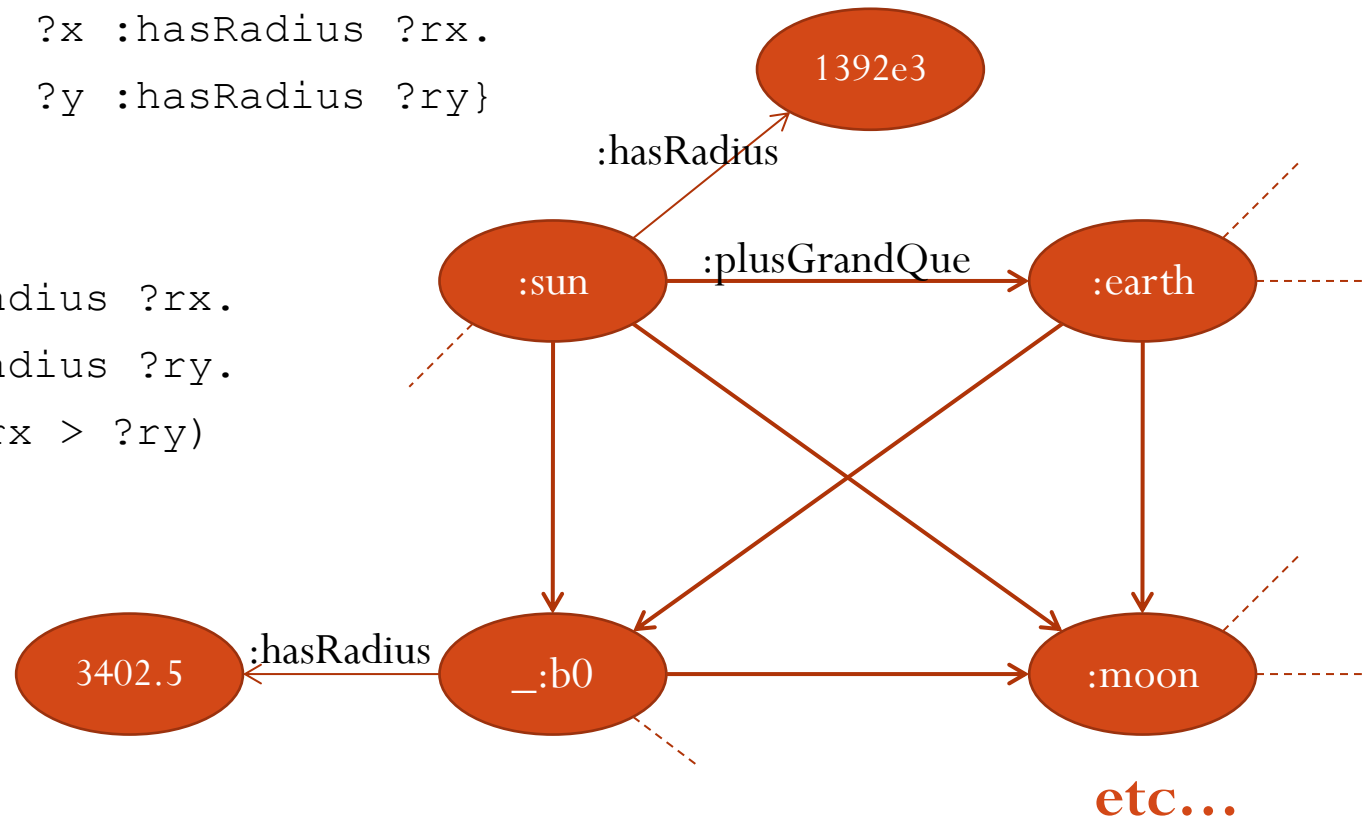
CONSTRUCT

- Exemple 1: construction d'un graphe ordonnant les corps célestes par leur rayon

CONSTRUCT {
 ?x :plusGrandQue ?y.
 ?x :hasRadius ?rx.
 ?y :hasRadius ?ry}

WHERE

```
{  
  ?x :hasRadius ?rx.  
  ?y :hasRadius ?ry.  
  FILTER(?rx > ?ry)  
}
```



CONSTRUCT

- Exemple 2: la terre possède-t-elle un satellite?

CONSTRUCT { :maRequete :reponse "Vrai". }

WHERE

```
{  
  :earth :satellite ?s  
}
```

- Résultat:



- Equivalent à **ASK**

ASK

- Cherche si la requête a un résultat ou non
- Le motif de requête peut-il être apparié au moins une fois dans la base de faits?
- Résultat: Vrai ou Faux
- Exemple: la terre possède-t-elle un satellite?

ASK

```
{  
  :earth :satellite ?s  
}
```

Résultat:

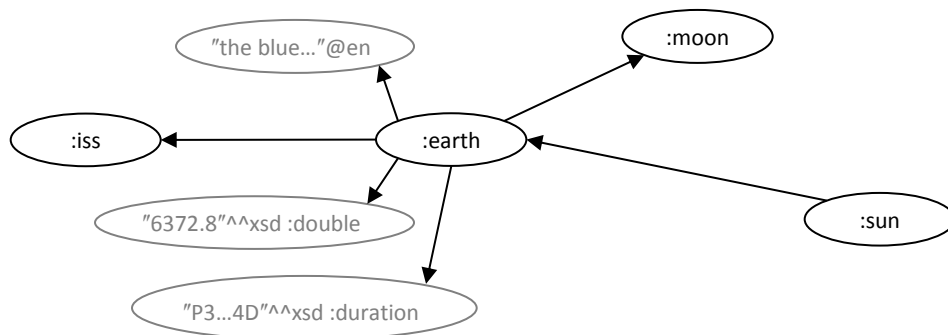
<boolean>true</boolean>

DESCRIBE

- Utilisé pour décrire une ressource
- Résultat non spécifié par la norme
- Dépend de l'application

- Exemple: décrire la terre

DESCRIBE :Earth



Equivalent à:

```
SELECT ?property ?hasValue ?isValueOf
WHERE {
  { :earth ?property ?hasValue }
  UNION
  { ?isValueOf ?property :earth }
}
```


DESCRIBE

- Utilisé pour décrire une ressource
- Résultat non spécifié par la norme
- Dépend de l'application
- Peut également inclure une clause WHERE
- Exemple: décrire toutes les planètes

```
DESCRIBE ?planet
WHERE
{
    ?planet a :Planet.
}
```

SPARQL Update

SPARQL Update

- Répond à plusieurs besoins:
 - Ajouter des données
 - Effacer des données
 - Modifier des données

Ajouter des données

INSERT DATA {...}

- Pour ajouter des donnée « brutes »
- Exemple: ajouter les rayons manquant

PREFIX : <http://ontologies.alwaysdata.net/space#>

PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>

INSERT DATA

```
{  
  :Jupiter :hasRadius "71492"^^xsd:double.  
}
```

Ajouter des données

INSERT {...} WHERE {...}

- Pour ajouter des connaissances reposant sur celles déjà présentes dans la base
- Fonctionne comme **CONSTRUCT**
- Exemple 1: introduire la relation :plusGrandQue

INSERT {?x :plusGrandQue ?y.}

WHERE

```
{  
  ?x :hasRadius ?rx.  
  ?y :hasRadius ?ry.  
  FILTER(?rx > ?ry)  
}
```

Ajouter des données

INSERT {...} WHERE {...}

- Exemple 2: créer et peupler une classe :BigPlanet

PREFIX : <http://ontologies.alwaysdata.net/space#>

PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>

PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>

PREFIX owl: <http://www.w3.org/2002/07/owl#>

```
INSERT { :BigPlanet rdf:type owl:Class ;  
          rdfs:label "big planet"@en ;  
          rdfs:comment "set of planets whose radius is over 20000km"@en .  
          ?bigPlanet a :BigPlanet.  
}  
WHERE { ?bigPlanet a :Planet ;  
          :hasRadius ?rx.  
        FILTER(?rx > 20000)  
}
```

Ajouter des données

`INSERT {...} WHERE {}` équivalent à `INSERT DATA {...}`

Effacer des données

DELETE DATA {...}

- Pour effacer des donnée « brutes »
- Exemple: enlever Pluton de la classe :Planet

DELETE DATA

```
{  
  :Pluton a :Planet.  
}
```


Effacer des données

DELETE {...} WHERE {...}

- Pour effacer des données en fonction de motifs présents dans la base de connaissances
- Exemple: enlever la classe **:BigPlanet** et les déclarations d'instances créées précédemment.

PREFIX : <http://ontologies.alwaysdata.net/space#>

```
DELETE {  
    ?i a :BigPlanet.  
    :BigPlanet ?p ?o.  
}  
WHERE {  
    OPTIONAL {?i a :BigPlanet.}  
    OPTIONAL {:BigPlanet ?p ?o.}  
}
```

Effacer des données

`DELETE {...} WHERE {}` équivalent à `DELETE DATA {...}`

Modifier des données

DELETE {...} INSERT {...} WHERE {...}

- Exemple 1: Changer les tags de langue des labels de planète

PREFIX : <http://ontologies.alwaysdata.net/space#>

PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>

```
DELETE {  
    ?planet rdfs:label ?lab.  
}  
INSERT {  
    ?planet rdfs:label ?frLab.  
}  
WHERE {  
    ?planet a :Planet.  
    ?planet rdfs:label ?lab.  
    FILTER (lang(?lab) = "fr")  
    BIND (strLang(str(?lab), "fr-FR") AS ?frLab)  
}
```

Modifier des données

DELETE {...} INSERT {...} WHERE {...}

- Exemple 2: Mettre la première lettre de chaque label de planète en majuscule

PREFIX : <http://ontologies.alwaysdata.net/space#>

PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>

```
DELETE {  
    ?planet rdfs:label ?lab.  
}  
INSERT {  
    ?planet rdfs:label ?newLab.  
}  
WHERE {  
    ?planet a :Planet.  
    ?planet rdfs:label ?lab.  
    BIND (str(?lab) AS ?strLab)  
    BIND (lang(?lab) AS ?langLab)  
    BIND (strLang( concat(ucase(substr(?strLab, 1, 1)),  
                           substr(?strLab, 2)),  
           ?langLab)  
          AS ?newLab)  
}
```

Modifier des données

DELETE {...} INSERT {...} WHERE {...}

- Exemple 2: remplacer les nœuds vides de planètes en créant dynamiquement des URI à partir de labels

Première approche – test avec CONSTRUCT:

PREFIX : <http://ontologies.alwaysdata.net/space#>

PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>

```
CONSTRUCT {  
    ?newUri ?p2 ?o.  
    ?s ?p1 ?newUri.  
}  
WHERE {  
    ?blankPlanet a :Planet.  
    ?blankPlanet rdfs:label ?lab.  
    OPTIONAL { ?s ?p1 ?blankPlanet. }  
    ?blankPlanet ?p2 ?o.  
    FILTER (isBlank(?blankPlanet))  
    BIND ( URI(ENCODE_FOR_URI(?lab)) AS ?newUri )  
}
```

→ problème dans le cas où un nœud vide a plusieurs labels

Deuxième approche – test avec CONSTRUCT:

PREFIX : <http://ontologies.alwaysdata.net/space#>

PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>

```
CONSTRUCT {  
    ?newUri ?p2 ?o.  
    ?s ?p1 ?newUri.  
}  
WHERE {  
    ?blankPlanet a :Planet.  
    { SELECT ?blankPlanet (SAMPLE(?lab) AS ?oneLabel)  
      WHERE  
      {  
          ?blankPlanet rdfs:label ?lab.  
      } GROUP BY ?blankPlanet  
    }  
    OPTIONAL { ?s ?p1 ?blankPlanet. }  
    ?blankPlanet ?p2 ?o.  
    FILTER (isBlank(?blankPlanet))  
    BIND ( URI(ENCODE_FOR_URI(?oneLabel)) AS ?newUri )  
}
```

→ OK, on peut passer au UPDATE

PREFIX : <http://ontologies.alwaysdata.net/space#>

PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>

DELETE {

 ?blankPlanet ?p2 ?o.

 ?s ?p1 ?blankPlanet.

}

INSERT{

 ?newUri ?p2 ?o.

 ?s ?p1 ?newUri.

}

WHERE {

 ?blankPlanet a :Planet.

 { SELECT ?blankPlanet (SAMPLE(?lab) AS ?oneLabel)

 WHERE

 {

 ?blankPlanet rdfs:label ?lab.

 } GROUP BY ?blankPlanet

 }

 OPTIONAL { ?s ?p1 ?blankPlanet. }

 ?blankPlanet ?p2 ?o.

 FILTER (isBlank(?blankPlanet))

 BIND (URI(ENCODE_FOR_URI(?oneLabel)) AS ?newUri)

}

Pour finir

Des exemples en vrai

- SPARQL endpoints sur le Web:
 - DbPedia: <http://fr.dbpedia.org/sparql>
 - ... liste ici: <http://www.w3.org/wiki/SparqlEndpoints>

OpenLink Virtuoso SPARQL Query

This query page is designed to help you test OpenLink Virtuoso SPARQL protocol endpoint.

Consult the [Virtuoso Wiki page](#) describing the service or the [Online Virtuoso Documentation](#) section [RDF Database and SPARQL](#).

There is also a rich Web based user interface with sample queries. You can access it at: [/isparql](#).

Query

Default Graph URI

<http://dbpedia.org>

Use only local data (including

Les pays sans littoral de plus de
15 millions d'habitants

Query text

```
PREFIX type: <http://dbpedia.org/class/yago/>
PREFIX prop: <http://dbpedia.org/property/>
SELECT ?country_name ?population
WHERE {
    ?country a type:LandlockedCountries ;
             rdfs:label ?country_name ;
             prop:populationEstimate ?population .
    FILTER (?population > 15000000 && langMatches(lang(?country_name), "EN")) .
} ORDER BY DESC(?population)
```

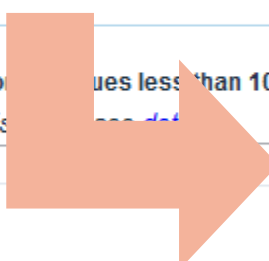
Rigorous check of the query: ☒ Execution timeout, in milliseconds: values less than 1000 are

Format Results As: **HTML** (The CXXML output is displayed as default)

Display the result and not save

Run Query

Reset



| country_name | population |
|--------------|------------|
| Ethiopia | 79221000 |
| Uganda | 32369558 |
| Nepal | 29331000 |
| Afghanistan | 28150000 |
| Niger | 15306252 |

Des exemples en vrai

- Implémentations de moteurs SPARQL
 - ARQ – <http://openjena.org> (serveur Fuseki)
 - OpenLink Virtuoso - <http://virtuoso.openlinksw.com/>
 - OpenAnzo - <http://www.openanzo.org/>
 - Corese - <http://www-sop.inria.fr/teams/edelweiss/wiki/wakka.php?>

Ressources

- Spécifications du W3C: <http://www.w3.org/>
 - Traductions françaises: <http://www.yoyodesign.org/>
- Learning SPARQL
 - O'REILLY
 - Bob Ducharme
- Foundations of Semantic Web Technologies
 - CRC Press
 - Pascal Hitzler, Markus Krötzsch, Sebastian Rudolph
- SPARQL in a nutshell
 - http://www.slideshare.net/fabien_gandon/sparql-in-a-nutshell
 - Présentation par Fabien Gandon
- SPARQL by example
 - <http://www.cambridgesemantics.com/2008/09/sparql-by-example/>
 - Présentation par Lee Feigenbaum

Contact

camillepradel@gmail.com

