

Résultat Sparql

```
<sparql xmlns="http://www.w3.org/2005/sparql-results#">
<head>
<variable name="nom"/>
<variable name="image"/>
<variable name="description"/>
</head>
<results ordered="false" distinct="true">
<result>
<binding name="nom">
<literal>Pierre Dumoulin</literal>
</binding> <binding name="image">
<uri>http://example.net/Pierre\_Dumoulin.jpg</uri> <
/binding> <binding name="description">
<literal>Photo d'identité de Pierre Dumoulin</literal>
</binding> </result> <result> <binding name="nom">
<literal>Paul Dupont</literal> </binding> <binding name="image">
<uri>http://example.net/Paul\_Dupont.jpg</uri>
</binding> <binding name="description">
<literal>Photo d'identité de Paul Dupont</literal>
</binding> </result> </results> </sparql>
```

Cours Web sémantique Langage SPARQL

Syntaxe SPARQL

- Une requête SPARQL est un nuplet (GP,DS,SM,R) où :
 - GP est un motif de graphe (motif de la requête)
 - DS est un ensemble de données RDF (base de données)
 - SM est un “transformateur de solution” : Projection, Distinct, Order, Limit, Offset
 - R est un format de résultat : SELECT, CONSTRUCT, DESCRIBE, ASK

Introduction

- Un langage et protocoles de requêtes pour l'interrogation de métadonnées sous forme de fichiers RDF
- Langage d'interrogations de triples RDF {sujet, objet, predicat}
- Un langage de requête :
 - <http://www.w3.org/TR/rdf-sparql-query/>
- Un protocole :
 - <http://www.w3.org/TR/rdf-sparql-protocol/>
 - Un format de résultat en XML :
- <http://www.w3.org/TR/rdf-sparql-XML-res/>

Syntaxe SPARQL

- Requête : Q = (GP,DS,SM, SELECTVS)

SELECT VS
FROM DS
WHERE { GP }

Requête SPARQL

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

PREFIX foaf: <<http://xmlns.com/foaf/0.1/>>

PREFIX dc: <<http://purl.org/dc/elements/1.1/>>

```
SELECT DISTINCT ?nom ?image ?description
WHERE {
?personne rdf:type foaf:Person .
?personne foaf:name ?nom .
?image rdf:type foaf:Image .
?personne foaf:img ?image .
?image dc:description ?description }
```

Requête simple

- PREFIX vcard:
<http://www.w3.org/2001/vcard-rdf/3.0#>
- SELECT ?y ?givenName
- WHERE { ?y vcard:Family "Smith" . ?y vcard:Given ?givenName . }

Requête simple

- SELECT ?x WHERE { ?x <http://www.w3.org/2001/vcard-rdf/3.0#FN> "John Smith" }

| x |

=====

<http://somewhere/JohnSmith/>

Filtres

- Opérations pour tester les chaînes de caractères, expressions régulières
 - SQL « LIKE »
- FILTER regex(?x, "pattern" [, "flags"])

```
SELECT ?g
WHERE { ?y vcard:Given ?g . FILTER regex(?g, "r", "i") }
• Réponse
----- g -----
=====
"Rebecca"
"Sarah "
```

Requête simple

- SELECT ?x, ?fname
- WHERE { ?x <http://www.w3.org/2001/vcard-rdf/3.0#FN> ?fname }

```
• | x          | name |
=====|
<http://somewhere/RebeccaSmith/> | "Becky Smith" |
| <http://somewhere/SarahJones/> | "Sarah Jones" | |
<http://somewhere/JohnSmith/> | "John Smith" | |
<http://somewhere/MattJones/> | "Matt Jones" |
• -----
```

Test

- Le mot clé FILTER impose des contraintes sur les variables

```
SELECT ?resource
WHERE { ?resource info:age ?age
FILTER (?age >= 24) }
```

```
resource
=====
<http://somewhere/JohnSmith/>
```

Requête simple

- SELECT ?givenName
- WHERE { ?y <http://www.w3.org/2001/vcard-rdf/3.0#Family> "Smith" . }
- ?y <http://www.w3.org/2001/vcard-rdf/3.0#Given> ?givenName . }

```
| givenName |
=====
| "John" |
| "Rebecca" |
```

OPTIONAL+FILTER

```
SELECT ?name ?age
WHERE { ?person vcard:FN ?name .
OPTIONAL { ?person info:age ?age . }
FILTER ( !bound(?age) || ?age > 24 ) }
```

name	age	
"Becky Smith "		
"Sarah Jones"		
"John Smith"	25	
"Matt Jones"		

- OPTIONAL + bound() permet d'exprimer la quantification existentielle.

OPTIONAL

- OPTIONAL permet de dire qu'un *pattern* peut être optionnel

```
SELECT ?name ?age
WHERE { ?person vcard:FN ?name
OPTIONAL { ?person info:age ?age } }
```

name	age	
"Becky Smith"	23	
"Sarah Jones"		
"John Smith"	25	
"Matt Jones"		

UNION

```
_:a foaf:name "Matt Jones" .
_:b foaf:name "Sarah Jones" .
_:c vcard:FN "Becky Smith" .
_:d vcard:FN "John Smith" .
```

```
SELECT ?name
WHERE { { [] foaf:name ?name } UNION { [] vCard:FN ?name } }
```

```
SELECT ?name1 ?name2
WHERE { { [] foaf:name ?name1 } UNION { [] vCard:FN ?name2 } }
```

OPTIONAL

```
SELECT ?name ?age
WHERE { ?person vcard:FN ?name
?person info:age ?age
}
```

name	age	
"Becky Smith"	23	
"John Smith"	25	

OPTIONAL+UNION

```
SELECT ?name1 ?name2
WHERE { ?x a foaf:Person
OPTIONAL { ?x foaf:name ?name1 }
OPTIONAL { ?x vCard:FN ?name2 } }
```

- OPTIONAL : pour augmenter le nombre de solutions trouvées
- UNION : pour concaténer

OPTIONAL+FILTER

```
SELECT ?name ?age
WHERE { ?person vcard:FN ?name .
OPTIONAL { ?person info:age ?age .
FILTER ( ?age > 24 ) } }
```

name	age	
"Becky Smith "		
"Sarah Jones"		
"John Smith"	25	
"Matt Jones"		

LIMIT/OFFSET

```
select * where {  
  PATTERN}  
LIMIT 20  
OFFSET 10
```

- Au plus 20 résultats
- A partir du 11ème

RDF Dataset

```
SELECT ?bobAge WHERE {  
  GRAPH data:graph1 {  
    ?x foaf:mbox <mailto:bob@work.example> .  
    ?x foaf:age ?bobAge  
  }  
}
```

CONSTRUCT

- Construire un graphe résultat :

```
construct {?boy c:hasSister ?girl}  
where {  
  ?girl c:hasBrother ?boy  
}
```

DISTINCT

```
select distinct ?x ?z where {  
  ?x c:friend ?y  
  ?y c:friend ?z  
}
```

- Ne retourne pas deux fois les mêmes valeurs de x et z

ASK / DESCRIBE

- ASK va répondre par OUI ou NON à une question
- **ask** {?x p ?y}
- DESCRIBE permet d'obtenir des informations détaillées

ORDER BY

```
select ?pers ?date where {  
  ?pers c:author ?doc  
  ?doc c:date ?date  
  order by ?date
```

Tests

- isURI(?x)
- isLiteral(?y)
- isBlank(?z)
- bound(?t)
- lang(?x) = 'en'
- datatype(?y) = xsd:string