

# Ingénierie du WEB Sémantique

## Interroger une ontologie : SPARQL

**Odile PAPINI**

**POLYTECH MARSEILLE. AIX MARSEILLE UNIVERSITE**

**`odile.papini@univ-amu.fr`**

<https://amubox.univ-amu.fr/s/jDtLY9Yj8YNaMNn>

**WORKSHOP SPARQL : être connecté à wikidata**  
**Université de Sfax 25-27 juin 2019**

# Plan du cours

- 1 Introduction
- 2 Le langage SPARQL
  - Structure d'une requête
  - Syntaxe SPARQL
  - Structure du résultat
  - Traitement du résultat

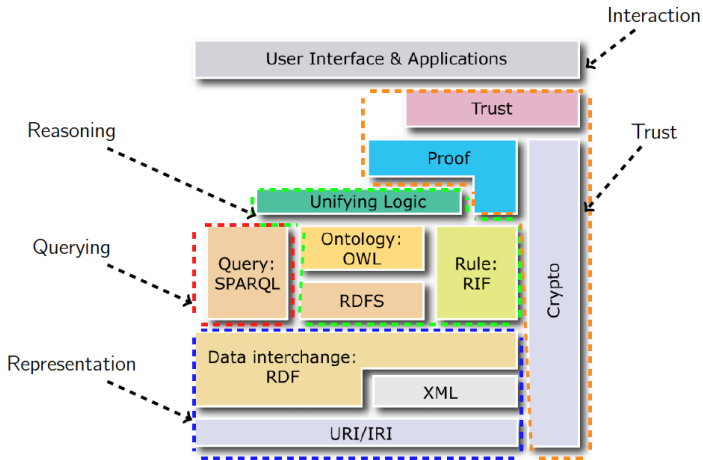
## Bibliographie I

-  Olivier Corby and Fabien Gandon and Catherine Faron-Zucker  
Le Web sémantique : comment lier les données et les schémas sur le web ?  
Dunod, 2012. ISBN : 978-2-10-057294-6.
-  John Hebler and Matthew Fisher and Ryan Blace and Andrew Perez-Lopez and Mike Dean  
Semantic Web Programming  
Wiley, 2009. ISBN : 978-0-470-41801-7
-  Grigoris Antoniou & Frank van Harmelen  
MIT university Press  
<http://www.ics.forth.gr/isl/swprimer/presentation.htm>  
[www.titan.be/common/docs/websemantique2007\\_1.ppt](http://www.titan.be/common/docs/websemantique2007_1.ppt)

## Bibliographie I

-  Pascal Hitzler and Markus Krotzsch and Sebastian Rudolph  
Foundations of Semantic Web Technologies,  
Chapman & Hall/CRC. 2009. ISBN : 9781420090505
-  John Domingue and Dieter Fensel and James A. Hendler  
Handbook of Semantic Web  
Springer. 2011. SBN : 978-3-540-92912-3
-  W3C  
<http://www.w3.org/standards/semanticweb/>
-  W3C  
<http://www.w3.org/TR/sparql11-query/>

# Le Web sémantique : La pile des standards du web sémantique



# La pile des standards du web sémantique

- **Représentation**
  - **URI/IRI** : Universal Resource Identifier/International Resource Identifier
  - **XML** : Extensible Markup Language
  - **RDF** : Resource Description Framework : description des ressources sous forme de graphe à base de triplets
- **Raisonnement**
  - **RDFS** : RDF Schema : langage de description de vocabulaire associé à RDF (description de classes et propriétés)
  - **OWL** : Ontology Web language : langage de représentation des ontologies
  - **RIF** : Rule Interchange Format : échange de systèmes à base de règles
- **Interrogation**
  - **SPARQL** : Simple Protocol And Rdf Query Language : langage d'interrogation de graphe RDF

## Rappels Représentation

### RDF

atome de connaissance en RDF : triplet < sujet, prédicat, objet >

- sujet : ressource (identifiée par un URI)
  - prédicat : propriété (identifiée par un URI)
  - objet : valeur de la propriété (ressource ou littéral )
- 
- représentation sous forme de graphe
    - sujet, objet : sommets
    - prédicat : arc entre sommets

**graphe RDF : combinaison de triplets**

# Interrogation : SPARQL

## SPARQL

- SPARQL : Protocol And RDF Query language
- Recommandation du W3C
- Langage d'interrogation du web sémantique (requêtes pour RDF/RDFS)
- SPARQL Recommandation :  
(<http://www.w3.org/TR/rdf-sparql-query/>)



# Interrogation : SPARQL

- Obtenir des valeurs à partir de données structurées ou semi-structurées
- Explorer des données en interrogeant des relations inconnues
- Réaliser des jointures efficaces de bases de données distribuées avec une seule requête simple
- Transformer des données RDF d'un vocabulaire à un autre

# Interrogation : SPARQL

- SPARQL 1.0 janvier 2008 :
  - SPARQL 1.0 Langage d'interrogation
  - SPARQL 1.0 Protocole
  - SPARQL 1.0 results : format XML
  
- SPARQL 1.1 : mises-à-jour du langage SPARQL et du Protocole SPARQL. Mars 2013 :
  - SPARQL 1.1 mise-à-jour
  - SPARQL 1.1 protocole pour gérer les graphes RDF
  - SPARQL 1.1 service de descriptions
  - SPARQL 1.1 inférences
  - SPARQL 1.1 basic federated query

# Interrogation : SPARQL

## Requête SPARQL

- Une requête est un graphe avec variables
- Recherche de sous-graphes dans un graphe donné (appariement de graphe)
- Recherche des valeurs des variables qui sont des sous-graphes du graphe représentant les données

# Exemple de requête SPARQL

graphe représentant les données à interroger

## RDF: Exemple

bd: ↔ <http://www.collection.com/bd/>

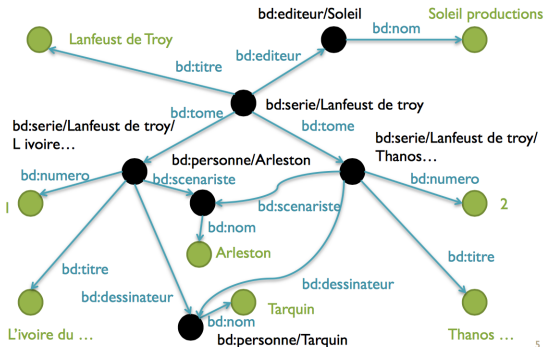


Figure: source : E. Coquery. LIRIS

# Exemple de requête SPARQL

traduction en RDF du graphe représentant les données à interroger

```
<?xml version="1.0" encoding="UTF-8"?>
<rdf:RDF
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:bd="http://www.collection.com/bd#">

  <rdf:Description rdf:about="http://www.collection.com/bd/Lanfeust de Troy">
    <bd:tome rdf:resource="http://www.collection.com/bd/serie/Laufeust de Troy/L
ivoire du Magohamoth"/>
    <bd:tome rdf:resource="http://www.collection.com/bd/serie/Laufeust de Troy/Thanos
1 incongru"/>
    <rdf:type rdf:resource="http://www.collection.com/bd/serie"/> </rdf:Description>

  <rdf:Description rdf:about="http://www.collection.com/bd/editeur/Soleil">
    <bd:nom>Soleil Productions</bd:nom>
  </rdf:Description>

  <rdf:Description
    rdf:about="http://www.collection.com/bd/Laufeust de Troy/L ivoire du
Magohamoth">
    <bd:numero rdf:datatype="http://www.w3.org/2001/XMLSchema#int">1</bd:numero>
  </rdf:Description>
```

# Exemple de requête SPARQL

## graphe requête

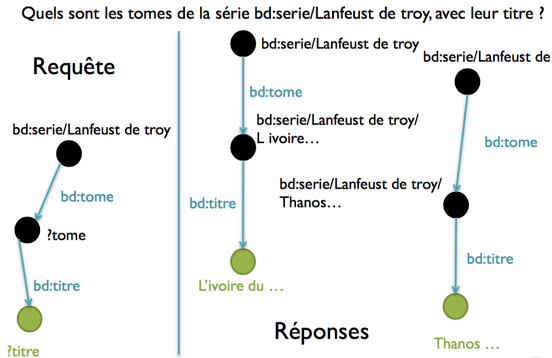


Figure: source : E. Coquery. LIRIS

## Exemple de requête SPARQL

requête

Quels sont les tomes de la série bd :serie/Lanfeust de troy avec leur titre ?

requête SPARQL

prefix bd : <<http://www.collection.com/bd/>>

```
select *  
where {  
  bd :serie/Lanfeust de troy bd :tome ?tome.  
  ?tome bd :titre ?titre  
}
```

# Interrogation

## Syntaxe inspirée de SQL

- Une requête est un graphe avec variables
- Recherche de sous-graphes dans un graphe donné (recherche des valeurs des variables qui sont des sous-graphes du graphe représentant les données)
- **SELECT** : projection : variables et valeurs retournées par la requête ( \* toutes les variables utilisées dans la clause WHERE)
- **[FROM]** : source (graphe de données) interrogée
- **WHERE** : contrainte : définit le "motif" du graphe qui s'apparie avec les données du graphe de donnée



## Exemple d'Interrogation BDpedia

```
PREFIX db-owl : <http://dbpedia.org/ontology/>  
select * where {  
  ?ville db-owl :region <http://fr.dbpedia.org/resource/Île-de-France>.  
  ?ville rdf:type db-owl :Settlement  
}
```

ville
<a href="http://fr.dbpedia.org/resource/Boissy-Saint-Léger">http://fr.dbpedia.org/resource/Boissy-Saint-Léger</a>
<a href="http://fr.dbpedia.org/resource/Boussy-Saint-Antoine">http://fr.dbpedia.org/resource/Boussy-Saint-Antoine</a>
<a href="http://fr.dbpedia.org/resource/Prétoy">http://fr.dbpedia.org/resource/Prétoy</a>
<a href="http://fr.dbpedia.org/resource/Le_Plessis-Robinson">http://fr.dbpedia.org/resource/Le_Plessis-Robinson</a>
<a href="http://fr.dbpedia.org/resource/Rueil-Malmaison">http://fr.dbpedia.org/resource/Rueil-Malmaison</a>
<a href="http://fr.dbpedia.org/resource/Sceaux_(Hauts-de-Seine)">http://fr.dbpedia.org/resource/Sceaux_(Hauts-de-Seine)</a>
<a href="http://fr.dbpedia.org/resource/Bry-sur-Marne">http://fr.dbpedia.org/resource/Bry-sur-Marne</a>
<a href="http://fr.dbpedia.org/resource/Massy_(Essonne)">http://fr.dbpedia.org/resource/Massy_(Essonne)</a>
<a href="http://fr.dbpedia.org/resource/Montrouge">http://fr.dbpedia.org/resource/Montrouge</a>
<a href="http://fr.dbpedia.org/resource/Nanterre">http://fr.dbpedia.org/resource/Nanterre</a>
<a href="http://fr.dbpedia.org/resource/Roinville_(Essonne)">http://fr.dbpedia.org/resource/Roinville_(Essonne)</a>
<a href="http://fr.dbpedia.org/resource/Saint-Germain-Laval_(Seine-et-Marne)">http://fr.dbpedia.org/resource/Saint-Germain-Laval_(Seine-et-Marne)</a>
<a href="http://fr.dbpedia.org/resource/Cergy">http://fr.dbpedia.org/resource/Cergy</a>
<a href="http://fr.dbpedia.org/resource/Chailly-en-Bière">http://fr.dbpedia.org/resource/Chailly-en-Bière</a>
<a href="http://fr.dbpedia.org/resource/Nangis">http://fr.dbpedia.org/resource/Nangis</a>
<a href="http://fr.dbpedia.org/resource/Saint-Michel-sur-Orge">http://fr.dbpedia.org/resource/Saint-Michel-sur-Orge</a>
<a href="http://fr.dbpedia.org/resource/Abbeville-la-Rivière">http://fr.dbpedia.org/resource/Abbeville-la-Rivière</a>
<a href="http://fr.dbpedia.org/resource/Andrésy">http://fr.dbpedia.org/resource/Andrésy</a>

Figure: source : DBpedia

# Interrogation SPARQL : 4 formes de requêtes

- **SELECT**
  - syntaxe comparable à SQL
  - recherche à appairer les termes RDF (noeud anonyme, IRI, littéraux) et les variables du motif du graphe
  - les résultats (appariements) sont retournés sous forme de table
- **CONSTRUCT**
  - reformule des variables sous forme de graphe RDF
  - transforme les données d'un graphe RDF à un autre graphe
  - retourne un graphe (qui peut être ajouté à un réservoir ou combiné à un autre graphe RDF )
- **ASK**
  - teste l'existence d'un résultat non vide
  - retourne une valeur booléenne
- **DESCRIBE**
  - donne des informations sur le graphe RDF
  - retourne un graphe

# SPARQL

- modèle de données : graphe RDF (graphe de triplets : (sujet, prédicat, object))
- ressources : représentées par des URI (ou IRI) abrégées par des préfixes
- objets : littéraux, chaînes de caractères, entiers, booléens , ...

# SPARQL

## Structure d'une requête SPARQL (la plus classique)

- Déclarations des espaces de nommage (préfixes : pour abrégier les URI)
- Choix de l'ensemble de données : le graphe RDF interrogé
- Sélection : identification des variables dont la valeur est retournée par la requête
- Motif de la requête : sous-graphe recherché sous quelles contraintes
- Traitement des résultats de la requête : trier, ordonner , limiter les resultats selon des critères

# SPARQL

## Structure d'une requête SPARQL

```
# prefix declarations
PREFIX foo: <http://example.com/resources/>
...
# dataset definition
FROM ...
# result clause
SELECT ...
# query pattern
WHERE {
    ...
}
# query modifiers
ORDER BY ...
```

Figure: source :

<http://dig.csail.mit.edu/2010/Courses/6.898/resources/sparql-tutorial.pdf>

## SPARQL : Exemple requêtes simples

PREFIX bd : <http://www.collection.com/bd/>

```
select *  
where {  
  ? t bd :dessinateur ?p  
}
```

résultats

t	p
Thanos l'incongru	Tarquin
L'ivoire de Magohamoth	Tarquin

# Exemple d'Interrogation BDpedia

```
PREFIX db-owl : <http://dbpedia.org/ontology/>
select * where {
?ville db-owl :region <http://fr.dbpedia.org/resource/Île-de-France> .
?ville rdf:type db-owl :Settlement .
?ville db-owl :populationTotal ?population .
FILTER ( ?population > 100000)
}
```

ville
<a href="http://fr.dbpedia.org/resource/Boissy-Saint-Léger">http://fr.dbpedia.org/resource/Boissy-Saint-Léger</a>
<a href="http://fr.dbpedia.org/resource/Boussy-Saint-Antoine">http://fr.dbpedia.org/resource/Boussy-Saint-Antoine</a>
<a href="http://fr.dbpedia.org/resource/Prétoy">http://fr.dbpedia.org/resource/Prétoy</a>
<a href="http://fr.dbpedia.org/resource/Le_Plessis-Robinson">http://fr.dbpedia.org/resource/Le_Plessis-Robinson</a>
<a href="http://fr.dbpedia.org/resource/Rueil-Malmaison">http://fr.dbpedia.org/resource/Rueil-Malmaison</a>
<a href="http://fr.dbpedia.org/resource/Sceaux_(Hauts-de-Seine)">http://fr.dbpedia.org/resource/Sceaux_(Hauts-de-Seine)</a>
<a href="http://fr.dbpedia.org/resource/Bry-sur-Marne">http://fr.dbpedia.org/resource/Bry-sur-Marne</a>
<a href="http://fr.dbpedia.org/resource/Massy_(Essonne)">http://fr.dbpedia.org/resource/Massy_(Essonne)</a>
<a href="http://fr.dbpedia.org/resource/Montrouge">http://fr.dbpedia.org/resource/Montrouge</a>
<a href="http://fr.dbpedia.org/resource/Nanterre">http://fr.dbpedia.org/resource/Nanterre</a>
<a href="http://fr.dbpedia.org/resource/Roinville_(Essonne)">http://fr.dbpedia.org/resource/Roinville_(Essonne)</a>
<a href="http://fr.dbpedia.org/resource/Saint-Germain-Laval_(Seine-et-Marne)">http://fr.dbpedia.org/resource/Saint-Germain-Laval_(Seine-et-Marne)</a>
<a href="http://fr.dbpedia.org/resource/Cergy">http://fr.dbpedia.org/resource/Cergy</a>
<a href="http://fr.dbpedia.org/resource/Chailly-en-Bière">http://fr.dbpedia.org/resource/Chailly-en-Bière</a>
<a href="http://fr.dbpedia.org/resource/Nangis">http://fr.dbpedia.org/resource/Nangis</a>
<a href="http://fr.dbpedia.org/resource/Saint-Michel-sur-Orge">http://fr.dbpedia.org/resource/Saint-Michel-sur-Orge</a>
<a href="http://fr.dbpedia.org/resource/Abbéville-la-Rivière">http://fr.dbpedia.org/resource/Abbéville-la-Rivière</a>
<a href="http://fr.dbpedia.org/resource/Andrésy">http://fr.dbpedia.org/resource/Andrésy</a>

# SPARQL : Syntaxe des IRI

## Production d'adresse IRI

**IRIref** : ensemble des adresses IRI

### Règles de grammaire :

[ 67 ]	<u>IRIref</u>	::=	<u>IRI_REF</u>   <u>PrefixedName</u>
[ 68 ]	<u>PrefixedName</u>	::=	<u>PNAME_LN</u>   <u>PNAME_NS</u>
[ 69 ]	<u>BlankNode</u>	::=	<u>BLANK_NODE_LABEL</u>   <u>ANON</u>
[ 70 ]	<u>IRI_REF</u>	::=	'<' ([^<>"{} ^`\ ]-[#x00-#x20])* '>'
[ 71 ]	<u>PNAME_NS</u>	::=	<u>PN_PREFIX?</u> ':'
[ 72 ]	<u>PNAME_LN</u>	::=	<u>PNAME_NS</u> <u>PN_LOCAL</u>



# SPARQL : Syntaxe des IRI

## Nom préfixés

### **PREFIX :**

- associe une étiquette de préfixe à une adresse IRI
- nom préfixé : étiquette\_de\_préfixe : partie\_locale

Exemple : différentes écritures de la même adresse IRI

*< http : //example.org/book/book1 >*

*BASE < http : //example.org/book/ >*  
*< book1 >*

*PREFIX book : < http : //example.org/book/ >*  
*book : book1*

# SPARQL : Syntaxe des littéraux

## littéraux

- " chaîne\_de\_caractères " ou ' chaîne\_de\_caractères '
- chaîne\_de\_caractères @ étiquette\_de\_langue
- chaîne\_de\_caractères ^^ IRI ou chaîne\_de\_caractères ^^ Type
- entiers, décimaux, booléens

- "chat"
- 'chat'@fr avec l'étiquette de langue "fr"
- "xyz"^^<http://example.org/ns/userDatatype>
- "abc"^^appNS:appDataType
- '''The librarian said, "Perhaps you would enjoy 'War and Peace'.'''
- 1, pareil à "1"^^xsd:integer
- 1.3, pareil à "1.3"^^xsd:decimal
- 1.300, pareil à "1.300"^^xsd:decimal
- 1.0e6, pareil à "1.0e6"^^xsd:double
- true, pareil à "true"^^xsd:boolean
- false, pareil à "false"^^xsd:boolean

# SPARQL : Syntaxe des variables d'interrogation

## variables d'interrogation

- ? nom\_de\_variable ou \$ nom\_de\_variable

### Règles de grammaire :

[44] Var ::= VAR1 | VAR2

[74] VAR1 ::= '?' VARNAME

[75] VAR2 ::= '\$' VARNAME

[97] VARNAME ::= ( PN\_CHARS\_U | [0-9] ) ( PN\_CHARS\_U | [0-9] |  
#x00B7 | [#x0300-#x036F] | [#x203F-#x2040] )\*

Figure: source : <http://www.w3.org/TR/rdf-sparql-query/>

# SPARQL : Syntaxe de noeuds anonymes

## noeuds anonymes

### variables indistinctes

- sous forme d'étiquette : `__ : nom_de_variable`
- sous forme agrégée : `[]`

### Règles de grammaire :

[ 39 ]	<u><a href="#">BlankNodePropertyList</a></u>	::=	'[ <u><a href="#">PropertyListNotEmpty</a></u> ]'
[ 69 ]	<u><a href="#">BlankNode</a></u>	::=	<u><a href="#">BLANK_NODE_LABEL</a></u>   <u><a href="#">ANON</a></u>
[ 73 ]	<u><a href="#">BLANK_NODE_LABEL</a></u>	::=	'_:' <u><a href="#">PN_LOCAL</a></u>
[ 94 ]	<u><a href="#">ANON</a></u>	::=	'[ <u><a href="#">WS*</a></u> ]'

Figure: source : <http://www.w3.org/TR/rdf-sparql-query/>

## SPARQL : Syntaxe de noeuds anonymes : Exemple

PREFIX bd : <http://www.collection.com/bd/>

```
select *  
where {  
  ? t bd :dessinateur [ ]  
}
```

# SPARQL : Syntaxe des triplets

## motif de triplet

sujet prédicat objet

- sujet commun, séparateur : ;
- sujet et prédicat communs, séparateur : ,
- plusieurs triplets, séparateur : .

### Règles de grammaire :

[32]	<u><i>TriplesSameSubject</i></u>	::=	<u><i>VarOrTerm</i></u> <u><i>PropertyListNotEmpty</i></u>   <u><i>TriplesNode</i></u> <u><i>PropertyList</i></u>
[33]	<u><i>PropertyListNotEmpty</i></u>	::=	<u><i>Verb</i></u> <u><i>ObjectList</i></u> ( ';' ( <u><i>Verb</i></u> <u><i>ObjectList</i></u> )? )*
[34]	<u><i>PropertyList</i></u>	::=	<u><i>PropertyListNotEmpty</i></u> ?
[35]	<u><i>ObjectList</i></u>	::=	<u><i>Object</i></u> ( ',' <u><i>Object</i></u> )*
[37]	<u><i>Verb</i></u>	::=	<u><i>VarOrIRIref</i></u>   'a'

## SPARQL : Syntaxe des motifs : Exemple

PREFIX bd : <http://www.collection.com/bd/>

```
select *  
where {  
  ? t bd :titre ?ti.  
  ? t bd :dessinateur [ ].  
}
```

```
select *  
where {  
  ? t bd :titre ?ti;  
  bd :dessinateur [ ].  
}
```

## SPARQL : Syntaxe des motifs : Exemple

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

?x foaf :nick "Alice",  
"Alice\_".

?x foaf :nick "Alice".  
?x foaf :nick "Alice\_".



# SPARQL : Motif de graphe

## motif de de graphe

- motif de graphe élémentaire : ensemble de motifs de triplets entre { }
- motif de graphe de groupe : groupe de groupe de motifs entre { }

### Règles de grammaire :

[32]	<i>TriplesSameSubject</i>	::=	<i>VarOrTerm</i> <i>PropertyListNotEmpty</i>   <i>TriplesNode</i> <i>PropertyList</i>
[33]	<i>PropertyListNotEmpty</i>	::=	<i>Verb</i> <i>ObjectList</i> ( ';' ( <i>Verb</i> <i>ObjectList</i> )? )*
[34]	<i>PropertyList</i>	::=	<i>PropertyListNotEmpty</i> ?
[35]	<i>ObjectList</i>	::=	<i>Object</i> ( ',' <i>Object</i> )*
[37]	<i>Verb</i>	::=	<i>VarOrIRIref</i>   'a'

## SPARQL : Syntaxe des motifs de graphe : Exemple

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

```
SELECT ?name, ?mbox  
WHERE {  
  ?x foaf :name ?name.  
  ?x foaf :mbox ?mbox.  
}
```

```
SELECT ?name, ?mbox  
WHERE {  
  { ?x foaf :name ?name. }  
  { ?x foaf :mbox ?mbox. }  
}
```

# SPARQL : définition de nouvelles variables

## définition de nouvelle variable et affectation de valeur

clause **BIND** (expression **AS** nom\_de\_variable) : définition de nouvelles variables et affectation de valeur

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

```
SELECT ?name  
WHERE {  
  ? p foaf :givenname ?g ;  
  foaf :surame ?s  
  BIND(CONCAT( ?g, " ", ?s) AS ?name) }  
}
```

Avec les données :

```
__ :a foaf :givenname "Alice".  
__ :a foaf : surname "Mabelle".
```

Le résultat :

name
"Alice Mabelle"

# SPARQL : collection RDF

## collections

Utilisation de collections : (element1 element2, ...) dans les motifs de triplets

```
(1?x 2 3 4) : p "w"
```

pour

```
_ :b0  rdf :first  1 ;  
      rdf :rest   _ :b1.  
_ :b1  rdf :first  ?x ;  
      rdf :rest   _ :b2.  
_ :b2  rdf :first  3 ;  
      rdf :rest   _ :b3.  
_ :b3  rdf :first  4 ;  
      rdf :rest   rdf :nil.  
_ :b0  :p         "w".
```

# SPARQL : structure du résultat

## forme du résultat

clause **SELECT** : renvoie un ensemble de n-uplets

- **SELECT** ?x?y : couple des valeurs du couple (x,y)
- **SELECT** \* : les valeurs de toutes les variables de la requête
- **SELECT DISTINCT** ?x?y : permet d'enlever les doublons
- SPARQL 1.1 : utilisation d'expressions et renommage des attributs

## SPARQL : structure du résultat : Exemple

```
PREFIX bd : <http://www.collection.com/bd/>
```

```
PREFIX fn : <http://www.w3.org/2005/xpath-functions>
```

```
SELECT (fn:concat(?ti, "par", ?scn, "et", ?den) as ?album
```

```
WHERE {
```

```
?t bd:titre ?ti.
```

```
?t bd:scenariste ?sc.
```

```
?sc bd:nom ?scn.
```

```
?t bd:dessinateur ?de.
```

```
?de bd:nom ?den.
```

```
}
```

# SPARQL : combinaison de motifs

## combinaison de motifs

- Opérateur ET : implicite (suite de triplets)
- Opérateur OU : **UNION**  
motif **UNION** motif

## SPARQL : combinaison de motifs : Exemples

PREFIX bd : <http://www.collection.com/bd/>

```
SELECT *  
WHERE {  
  ?t bd :titre ? ti.  
  ?t bd :dessinateur [ ].  
}
```

```
SELECT *  
WHERE {  
  ?t bd :scenariste ? p.  
UNION  
  ?t bd :dessinateur ?p.  
}
```



# SPARQL : motif optionnel

## filtrage par motif optionnel

- motif **OPTIONAL** { motif }
- { motif **OPTIONAL** { motif } }
- { { } motif **OPTIONAL** { motif } }

## SPARQL : motif optionnel : Exemple

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

```
select ?name, ?mbox
WHERE {
  ? x foaf :name ?name.
  OPTIONAL { ? x foaf :mbox ?mbox }
}
```

Avec les données :

```
_ :a rdf:type foaf:Person.
_ :a foaf:mame "Alice".
_ :a foaf:mbox <mailto:alice@example.com>.
_ :b rdf:type foaf:Person.
_ :b foaf:mame "Bob".
```

Les résultats :

name	mbox
"Alice"	<mailto:alice@example.com>
"Bob"	

# SPARQL : localiser les sources de données

## localiser les données

### GRAPH

- **GRAPH** URI : choix du graphe à interroger localisé avec son URI
- **GRAPH** ?x : recherche de la source des résultats

## SPARQL : localiser les sources de données : Exemple

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

```
SELECT *
```

```
WHERE {
```

```
? x rdf:type foaf:Person.
```

```
GRAPH <http://www.inria.fr/data> { ?x foaf:name ?name }
```

```
GRAPH ?g { ?x foaf:knows ?y }
```

```
}
```

# SPARQL : choisir les sources de données

## choisir les sources de données

### GRAPH

- **FROM** URI : choisir le graphe à interroger, l'identifié par son URI
- **FROM NAMED GRAPH** : choisir le graphe à interroger et possibilité de faire référence à ce graphe dans la clause WHERE avec **GRAPH**

## SPARQL : choisir les sources de données : Exemple

```
PREFIX bd : <http://www.collection.com/bd#>
```

```
SELECT *  
FROM <http://www.collection.com/bd#g1>  
FROM NAMED bd :g2  
WHERE {  
  ?t bd :titre ?ti .  
  GRAPH bd :g2 { ?t bd :titre ?ti2 } .  
}
```

## SPARQL : choisir les sources de données : Exemple

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

```
FROM NAMED <http://www.cnrs.fr/base>
```

```
FROM NAMED <http://www.inria.fr/data>
```

```
SELECT *
```

```
WHERE {
```

```
  GRAPH { ?x foaf :name ?name }
```

```
  GRAPH ?g { ?x foaf :name ?name }
```

```
}
```

## SPARQL filtrage des résultats

restreindre l'ensemble des résultats selon la contrainte

**FILTER( contrainte )** : contrainte construite avec opérateurs et fonctions

Opérateurs

- logiques : **!**, **&&**, **||**
- mathématiques : **+**, **-**, **\***, **/**
- de comparaison : **=**, **! =**, **<**, **>**

Fonctions prédéfinies

- tests : **isURI**, **isBlank**, **isLiteral**, **bound**
- accesseurs : **str**, **lang**, **datatype**
- autre : **sameTerm**, **langMatches**, **regex**



# SPARQL filtrage des résultats

## Fonctions prédéfinies

- **isURI**(term) : renvoie vrai ssi term est un URI
- **isBlank**(term) : renvoie vrai ssi term est un noeud anonyme
- **isLiteral**(term) : renvoie vrai ssi term est un littéral
- **bound**(? var) : renvoie vrai ssi var est associé à une valeur
- **str**(term) : renvoie la chaîne de caractères correspondant à term
- **lang**(lit) : renvoie le code langue de lit
- **datatype**(lit) : renvoie l'URI du type de lit
- **sameTerm**(term1, term2) : renvoie vrai si les termes sont égaux
- **langMatches**(lang, rang) : renvoie vrai si le langage lang appartient au domaine rang
- **regex**(text, motif, options) : renvoie vrai si l'expression donnée par motif correspond à la chaîne text selon l'option spécifiée

## SPARQL : Filtrage des résultats : Exemple

```
PREFIX bd : <http://www.collection.com/bd#>  
PREFIX db-xsd : <http://www.w3.org/2001/XMLSchema#>
```

```
SELECT *  
WHERE {  
  ? t bd :numero ?n .  
  FILTER( ?n > "1" ^^ xsd :integer)  
}
```

## SPARQL : filtrage des résultats : Exemple

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

```
SELECT *
```

```
WHERE {
```

```
?x foaf :name ?name.
```

```
?x foaf :mbox ?mbox.
```

```
FILTER regex ( ?name, "Dupont" )
```

```
}
```

# SPARQL filtrage des résultats

## nouveaux opérateurs et fonctions avec SPARQL 1.1

### Opérateurs

- **FILTER** (IF condition, contrainte )
- **FILTER** ( expression **IN** énumération )
- **FILTER** ( expression **NOT IN** énumération ), ...

### Fonctions prédéfinies

- **IRI**(chaîne)/ **URI**(chaîne) : construit un IRI/URI à partir d'une chaîne
- **BNODE** : construit un noeud anonyme, ...
- des fonctions de manipulation de chaînes de caractères :
  - **CONCAT**, **CONTAINS**, **STRSTART**, **STRENDS**, ...
- des fonctions de manipulation numérique
  - **RAND**, **ABS**, ...
- des fonctions de manipulation de dates
  - **NOW**, **DAY**, **HOURS**, ...

## SPARQL : filtrage des résultats : Exemple

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

```
SELECT *
```

```
WHERE {
```

```
?x foaf :name ?n.
```

```
FILTER ( ?n in ( "Dupont", "Dupuis", "Durand" )
```

```
}
```

# SPARQL trier les résultats

## Ordonner les résultats

- ORDER BY ( expression )
- ORDER BY ( expression ) DESC [ condition ]
- ORDER BY ( expression ) ASC [ condition ]

## SPARQL : filtrage des résultats : Exemple

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

```
SELECT ?name  
WHERE {  
  ?x foaf :name ?name  
}  
ORDER BY ?name
```

```
SELECT *  
WHERE {  
  ?x foaf :name ?name ;  
  foaf :age ?age }  
ORDER BY ?name DESC( ?age)
```

# SPARQL limiter le nombre de résultats

## Limiter le nombre de résultats

- **LIMIT** entier : donne une limite supérieure au nombre de résultats

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

```
SELECT ?name  
WHERE {  
  ?x foaf :name ?name  
}  
LIMIT 5
```



## SPARQL limiter le nombre de résultats

### Limiter le nombre de résultats

- **OFFSET** entier : donne le rang à partir duquel les résultats sont considérés

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

```
SELECT ?name  
WHERE {  
  ?x foaf :name ?name  
}  
ORDER BY ?name  
LIMIT 5  
OFFSET 10
```

## SPARQL negation

tester la non existence d'un motif

- FILTER (NOT EXISTS motif )

PREFIX bd : <http://www.collection.com/bd/>

```
SELECT *  
WHERE {  
  ?serie bd :tome ? to.  
  FILTER( NOT EXISTS { ?serie bd :editor [ ] } ).  
}
```

## SPARQL negation

### tester la non existence d'un motif

- identifier le motif à écarter : **OPTIONAL** motif
- écarter les résultats où le motif existe : **FILTER (! bound (?x)).**

PREFIX bd : <http://www.collection.com/bd/>

```
SELECT *  
WHERE {  
  ?serie bd :tome ? to.  
  OPTIONAL { ?serie bd :editor ?x }.  
  FILTER (! bound(? x)). }  
}
```

## SPARQL sous-requêtes

### requêtes imbriquées

```
SELECT expression1
WHERE {
    SELECT expression2 WHERE { ... }
}
```

exemple

# SPARQL agrégations

## grouper les résultats

- GROUP BY
- fonctions d'agrégation sur les groupes : COUNT, SUM, MIN, MAX, AVG, ...
- contraintes sur les groupes : HAVING

PREFIX bd : <http://www.collection.com/bd/>

```
SELECT ?serie (COUNT( ?to) as ?nbTomes)
WHERE { ?serie bd :tome ? to. }
GROUP BY ?serie
HAVING ( ?nbTomes > 3)
```

# SPARQL chemin de propriétés

## chemin de propriétés

Chercher des ressources reliées par un chemin de longueur variable dans un graphe RDF

Chemin : expression formée à partir de noms de propriétés et de d'opérateurs :

répétition : +, \*

répétition bornée : { n, m }

optionnel : ?

sequence : /

alternative : |

inverse : ^

négation : !

Exemple : liste RDF : chemin de longueur variable

```
SELECT ?value
```

```
WHERE {
```

```
?x ex :values ?list
```

```
? list rdf :rest*/rdf :first ?value
```

```
}
```

# SPARQL Construction de graphe

créer des graphes résultats

CONSTRUCT à la place de WHERE

PREFIX bd : <http://www.collection.com/bd/>

```
CONSTRUCT {  
  ?serie bd :album ?ti.  
  ?serie bd :numero ?n. }  
WHERE {  
  ?serie bd :tome ?to.  
  ?to bd :numero ?n.  
  ?to bd :ti ?ti.  
}
```

# SPARQL 1.1

Pour Plus de détails consulter la documentation W3C SPARQL 1.1 :

<http://www.w3.org/TR/sparql11-query/>