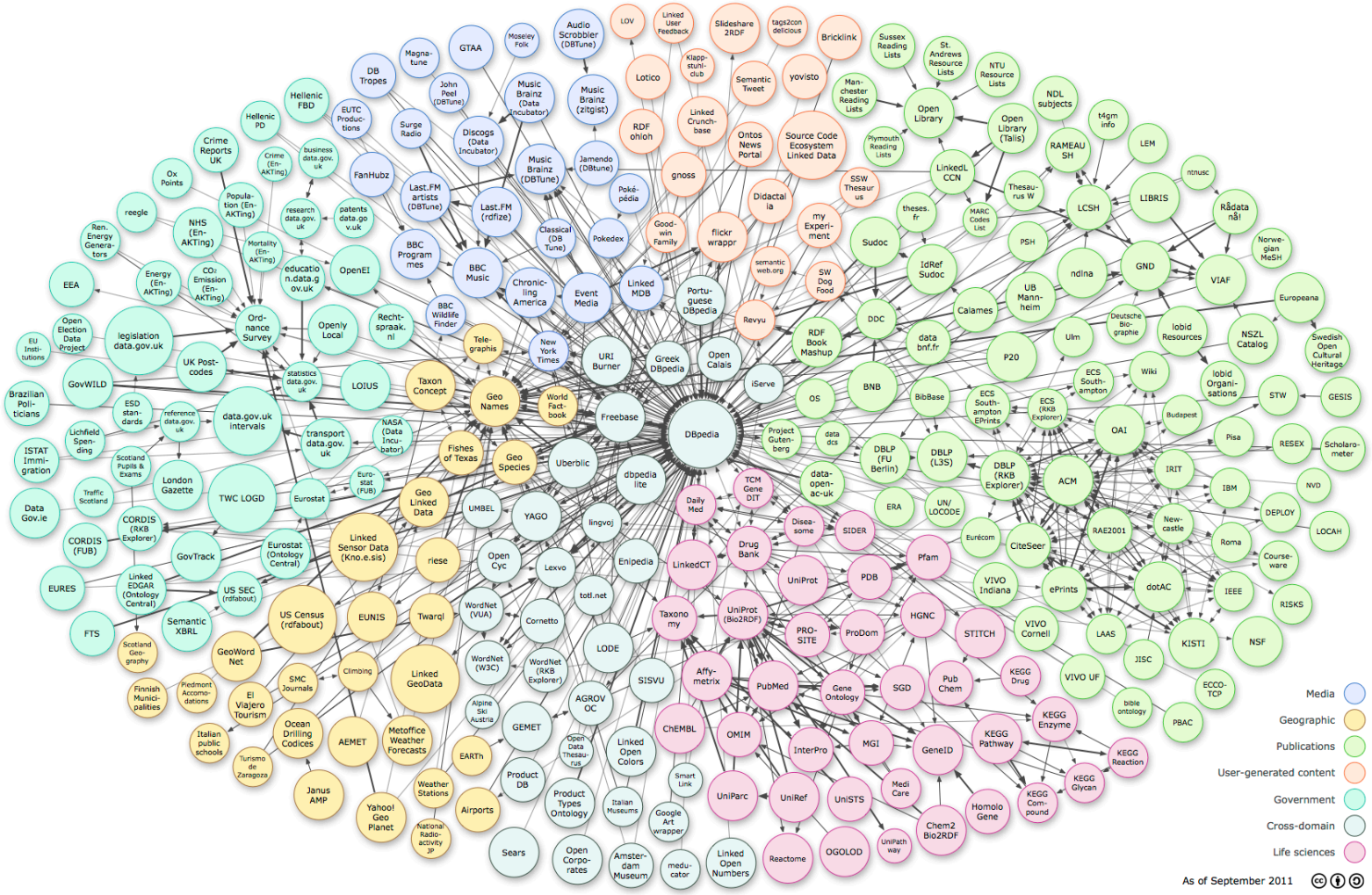
Module MLBDA
Master Informatique
Spécialité DAC

Cours 9 – SPARQL

Linking Open Data



Linking Open Data cloud diagram, by Richard Cyganiak and Anja Jentzsch.
http ://lod-cloud.net/

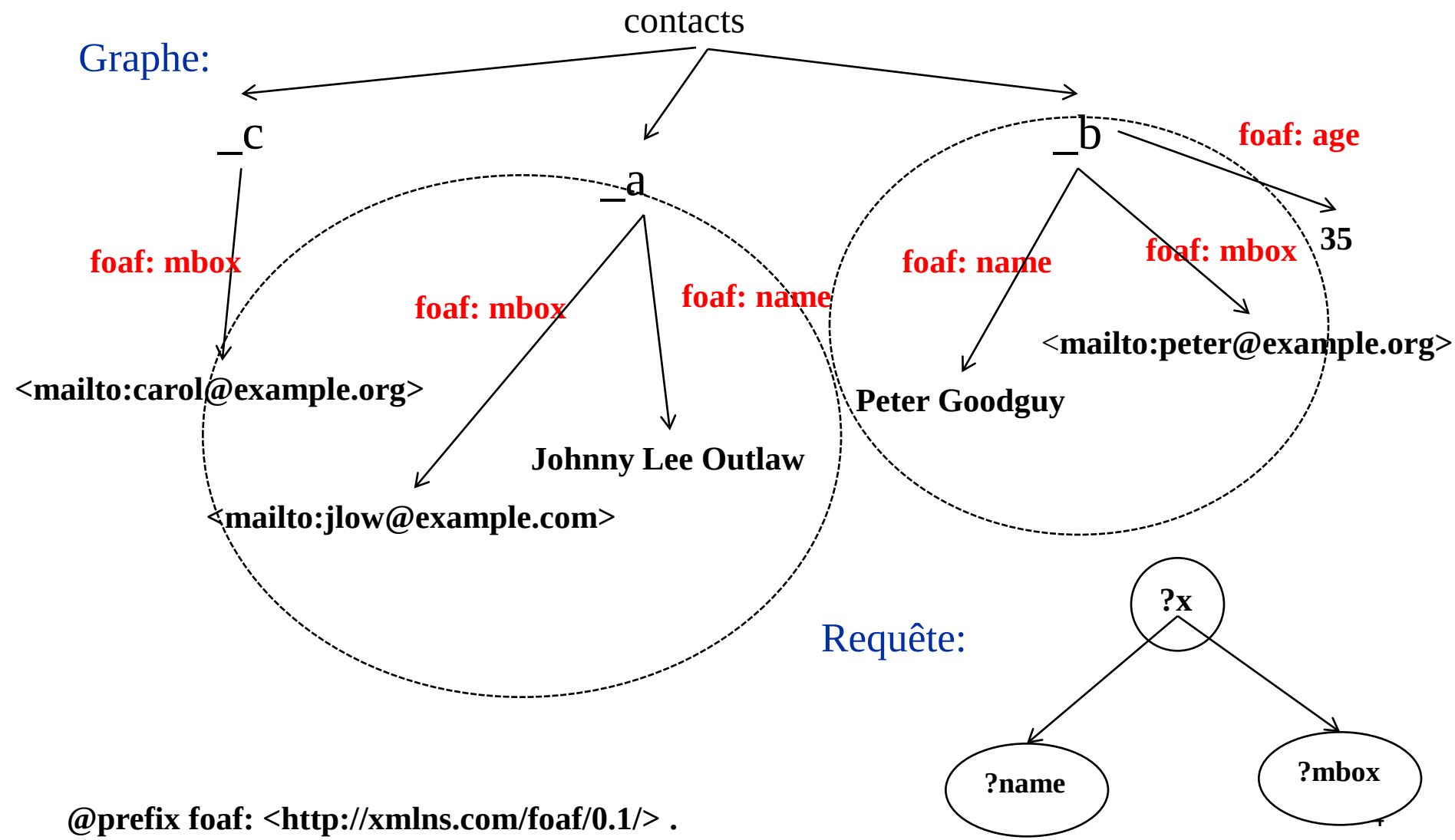
SPARQL

- SPARQL : *Simple Protocol and RDF Query Language*

- Langage de requêtes du W3C pour RDF/RDFS
  - SPARQL 1.0 : recommandation 2008
  - SPARQL 1.1 : recommandation 2013

- Principe : pattern matching sur des graphes
- Plusieurs formes de requêtes
  - Select
  - Construct, Describe
  - Ask

Exemple



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

Exemple

Données (un graphe):

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

_:a  foaf:name   "Johnny Lee Outlaw" .
_:a  foaf:mbox   <mailto:jlow@example.com> .
_:b  foaf:name   "Peter Goodguy" .
_:b  foaf:mbox   <mailto:peter@example.org> .
_:b  foaf:age   35.
_:c  foaf:mbox   <mailto:carol@example.org> .
```

Requête de base

- **PREFIX :** permet de déclarer les espaces de noms.
- **SELECT :** permet d'identifier les variables du résultat (parmi les variables de la clause WHERE).
- **WHERE** définit le motif de graphe recherché à l'aide de triplets (comme en RDF).

Exemple

Requête (motif de graphe) :

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

Réponse :

Name	Mbox
"Johnny Lee Outlaw"	<mailto:jlow@example.com>
"Peter Goodguy"	<mailto:peter@example.org>

Littéraux et nœuds blancs

- Les littéraux : chaînes de caractères entre quotes.

```
"valeur", @motcle, ^^type,
  "false"^^xsd:boolean, "1.3"^^xsd:decimal
```

- On peut simplifier et écrire
  - **false** au lieu de `"false"^^xsd:boolean`
  - **1.3** au lieu de `"1.3"^^xsd:decimal`
- Les nœuds blancs sont désignés par un label (_xx) ou par []

```
[ :a "toto" ] :b "titi" .
```

est équivalent à

```
_xx :a "toto" .
_xx :b "titi" .
```

Espaces de noms

- On peut définir un préfixe associé à un espace de noms, et un espace de nom "base" de la requête.

```
PREFIX dc: <http://purl.org/dc/elements/1.1/>
SELECT ?title
WHERE { <http://example.org/book/book1> dc:title ?title }

PREFIX dc: <http://purl.org/dc/elements/1.1/>
PREFIX : <http://example.org/book/>
SELECT $title
WHERE { :book1 dc:title $title }

BASE <http://example.org/book/>
PREFIX dc: <http://purl.org/dc/elements/1.1/>
SELECT $title
WHERE { <book1> dc:title $title }
```

Motif de triplets

L'ensemble des termes RDF est défini par T = U ∪ L ∪ B où

U : ensemble des URI

`<http://www.w3.org/TR/rdf-syntax-grammar>`

`dc:title`

L : littéraux RDF (valeurs) : `"valeur", @motcle, ^^type`

`"Dave Beckett", @prefix, "false"^^xsd:boolean, "1.3"^^xsd:decimal`

B : nœuds blancs

Un triplet RDF est un élément de l'ensemble (U ∪ B) U T

Un motif de triplets est un élément de l'ensemble (T ∪ V) (U ∪ V) (T ∪ V)
où V est un ensemble de variables tel que V ∩ T = ∅

Les variables sont préfixées par ? ou par $

Exemples

- Motif ∈ U U L

```
<http://example.org/book/book1>
<http://purl.org/dc/elements/1.1/title> "SPARQL" .
```

- Motif ∈ V U V

```
?book dc:title ?title .
```

- Motif ∈ B U L (avec définition de préfixe)

```
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
_:a foaf:name "Bob" .
```

Motifs de triplets

Pour simplifier l'écriture des motifs de triplets, on peut

- Les factoriser

```
?x foaf:name ?name;  foaf:mbox ?mbox .
?x foaf:nick "Robert", "Bob"
```

- Utiliser les collections RDF

```
(1 ?x 3) :a "toto" .
```

équivaut à

```
_:b1 rdf:first 1; rdf:rest _:b2 .
_:b2 rdf:first ?x; rdf:rest _:b3 .
_:b3 ref:first 3; rdf:rest rdf:nil .
_:b1 :a "toto" .
```

- Utiliser des abréviations (`rdf:type`)

`?x a :Class1` équivaut à `?x rdf:type :Class1`

Motifs de graphes

- Un ensemble de *motifs de triplets* est un motif de graphe (élémentaire)
- Si **`GP`** est un motif de graphe, `{GP}` est un motif de graphe de groupe.
- Si **`GP`** et **`GP'`** sont des motifs de graphe :
  - **`GP FILTER (test) GP'`** : sélection/filtrage
  - **`GP OPTIONAL GP'`** : graphes optionnels
  - **`GP UNION GP'`** : union de graphes

  sont des motifs de graphe.

Exemples de motifs de graphe

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

- Motif de graphe élémentaire :

```
_:a foaf:age 26 . _:a foaf:mbox ?m .
```

- Motif de groupe :

```
{ _:a foaf:age 26 . _:a foaf:mbox ?m . }
```

- Motif de groupe avec FILTRE :

```
{ _:a foaf:age ?age .
FILTER (?age < 26)
_:a foaf:mbox ?m . }
```

- Motif de groupe avec OPTIONAL :

```
{ _:a foaf:age 26 . OPTIONAL { _:a foaf:mbox ?m . }}
```

Filtres

- Le mot-clef **FILTER** permet de restreindre les solutions sur tout le groupe où le filtre apparaît. La position du filtre dans le groupe n'a pas d'importance, les motifs suivants sont équivalents :

```
{ ?x ns:price ?price .
FILTER (?price < 15)
?x dc:title ?title . }

{FILTER (?price < 15)
?x ns:price ?price .
?x dc:title ?title . }

{ ?x ns:price ?price .
?x dc:title ?title .
FILTER (?price < 15) }
```

Motif Optionnel

- Un motif de graphe élémentaire permet de rechercher des solutions qui correspondent entièrement au motif d'interrogation.
- Le filtrage optionnel (mot-clé **OPTIONAL**) permet d'obtenir des solutions même si des parties du motif d'interrogation ne correspondent pas.

```
{ _:a foaf:name ?name .
  OPTIONAL { _:a foaf:mbox ?m . }}

{ ?x dc:title ?title .
         OPTIONAL { ?x ns:price ?price .
                    FILTER (?price < 15) }}
{ ?x foaf:name  ?name .
        OPTIONAL { ?x foaf:mbox ?mbox } .
        OPTIONAL { ?x foaf:homepage ?hpage }}
```

Filtrage des alternatives

- Le mot-clef **UNION** permet d'indiquer des alternatives de motifs (un motif OU un autre peut correspondre). Si plusieurs alternatives correspondent, toutes les solutions sont trouvées.

```
{ {?x foaf:name ?name .
   ?x foaf:mbox ?mbox}
UNION
   {?x foaf:name ?name .
   ?x foaf:hpage ?hpage}
}
```

Requête SPARQL

Une requête SPARQL est de la forme

```
<FORMAT>
  FROM source
  WHERE { motif }
<TRANSFORM>
```

<FORMAT> définit le format du résultat : SELECT, CONSTRUCT, DESCRIBE, ASK

FROM définit la source RDF (optionnel si source par défaut)

motif est un motif de graphe

<TRANSFORM> est un transformateur : order, limit, offset

Formats

format	**résultat**
SELECT, SELECT DISTINCT	table de données
CONSTRUCT	graphe RDF
ASK	valeur Booléenne (match non-vide)
DESCRIBE	description des ressources trouvées

Clause FROM

- La clause FROM est utilisée pour indiquer l'IRI d'une ressource sur lequel effectuer la requête.
- En l'absence de clause FROM, l'interrogation s'effectue sur le graphe par défaut.

```
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
_:a foaf:name "Alice" foaf:mbox
   <mailto:alice@work.example> .
_:b foaf:name "Bob" foaf:mbox
   <mailto:bob@work.example> .
```

Requête

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?name
FROM <http://example.org/foaf/aliceFoaf>
WHERE { ?x foaf:name ?name }
```

Sémantique

- Une solution de motif (de graphe ou de triplet) est une fonction de substitution d'un ensemble de variables vers l'ensemble des termes

- Une solution de motif S est une solution pour une requête

  `<format> FROM source WHERE { motif }`

  si S appliqué au motif est un sous-graphe de **`source`**.

- Le résultat d'une requête est l'ensemble de toutes ses solutions de motifs.

Exemples de requêtes

Données :

```
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .

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

Exemples de requêtes

Requête: optional

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?name ?mbox
WHERE  { ?x foaf:name  ?name .
         OPTIONAL { ?x  foaf:mbox  ?mbox }
       }
```

Résultat :

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

Exemples de requêtes

Données :

```
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .

_:a  rdf:type         foaf:Person .
_:a  foaf:name        "Alice" .
_:a  foaf:homepage    <http://work.example.org/alice/> .
_:b  rdf:type         foaf:Person .
_:b  foaf:name        "Bob" .
_:b  foaf:mbox        <mailto:bob@work.example> .
```

Exemples de requêtes

Requête : plusieurs motifs OPTIONAL

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?name ?mbox ?hpage
WHERE  { ?x foaf:name  ?name .
         OPTIONAL { ?x foaf:mbox ?mbox } .
         OPTIONAL { ?x foaf:homepage ?hpage }
       }
```

Résultat :

name	mbox	hpage
"Alice"		<http://work.example.org/alice/>
"Bob"	<mailto:bob@work.example>	

Exemples de requêtes

Requête : plusieurs motifs OPTIONAL (pas équivalent à la requête précédente)

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?name ?mbox ?hpage
WHERE  { ?x foaf:name  ?name .
         OPTIONAL { ?x foaf:mbox ?mbox  .
                    ?x foaf:homepage ?hpage }
       }
```

Résultat :

name	mbox	hpage
"Alice"		
"Bob"		

Exemples de requêtes

Données :

```
@prefix dc: <http://purl.org/dc/elements/1.1/> .
@prefix : <http://example.org/book/> .
@prefix ns: <http://example.org/ns#> .
:book1 dc:title "SPARQL Tutorial" .
:book1 ns:price 42 .
:book2 dc:title "The Semantic Web" .
:book2 ns:price 23 .
```

Exemples de requêtes

Requête : filtrage

```
PREFIX dc: <http://purl.org/dc/elements/1.1/>
PREFIX ns: <http://example.org/ns#>
SELECT ?title ?price
WHERE { ?x ns:price ?price .
FILTER (?price < 30) .
?x dc:title ?title . }
```

Résultat :

Title	Price
"The Semantic Web"	23

Exemples de requêtes

Requête : optional et filtre

```
PREFIX dc: <http://purl.org/dc/elements/1.1/>
PREFIX ns: <http://example.org/ns#>
SELECT ?title ?price
WHERE {?x dc:title ?title .
OPTIONAL {?x ns:price ?price .
          FILTER (?price < 30) .}}
```

Résultat :

Title	Price
"SPARQL Tutorial"	
"The Semantic Web"	23

Exemples de requêtes

Données :

```
@prefix dc10:  <http://purl.org/dc/elements/1.0/> .
@prefix dc11:  <http://purl.org/dc/elements/1.1/> .
_:a  dc10:title   "SPARQL Query Language Tutorial" .
_:a  dc10:creator   "Alice" .
_:b  dc11:title     "SPARQL Protocol Tutorial" .
_:b  dc11:creator   "Bob" .
_:c  dc10:title     "SPARQL" .
_:c  dc11:title     "SPARQL (updated)" .
```

Exemples de requêtes

Requête : avec UNION

```
PREFIX dc10:  <http://purl.org/dc/elements/1.0/>
PREFIX dc11:  <http://purl.org/dc/elements/1.1/>

SELECT ?title
WHERE  { { ?book dc10:title  ?title } UNION { ?book
  dc11:title  ?title } }
```

Résultat :

Title
"SPARQL Protocol Tutorial"
"SPARQL "
"SPARQL (updated)"
"SPARQL Query Language Tutorial"

Exemples de requêtes

Requête : avec UNION (en utilisant des variables différents)

```
PREFIX dc10:  <http://purl.org/dc/elements/1.0/>
PREFIX dc11:  <http://purl.org/dc/elements/1.1/>

SELECT ?x ?y
WHERE  { { ?book dc10:title  ?x } UNION { ?book
  dc11:title  ?y } }
```

Résultat :

x	y
	"SPARQL (updated)"
	"SPARQL Protocol Tutorial"
"SPARQL"	
"SPARQL Query Language Tutorial"	

Séquences de solutions

- Les motifs de solution génèrent une séquence non ordonnée de solutions, chacun étant une fonction partielle de variables présentes dans les motifs vers des termes RDF.
- Ces solutions sont ensuite traitées comme une séquence sur laquelle on peut appliquer des opérateurs (modificateurs)
  - Order by : permet de trier les solutions
  - Projection : choix des variables (SELECT)
  - Distinct : éliminer les doublons parmi les solutions
  - Reduced : peut éliminer des solutions non uniques
  - Offset : indique la position où commencer dans la séquence de solutions
  - Limit : restreint le nombre de solution

ORDER BY

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?name
WHERE { ?x foaf:name ?name }
ORDER BY ?name
```

```
PREFIX foaf:    <http://xmlns.com/foaf/0.1/>
PREFIX     :    <http://example.org/ns#>
SELECT ?name
WHERE { ?x foaf:name ?name ; :empId ?emp }
ORDER BY ?name DESC(?emp)
```

DISTINCT

Données :

```
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
_:a foaf:name "Alice" foaf:mbox <mailto:alice@work.example> .
_:x foaf:name "Alice" foaf:mbox <mailto:smith@work.example> .
```

Requête :

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT DISTINCT ?name
WHERE { ?x foaf:name ?name }
```

Résultat

name
"Alice"

OFFSET et LIMIT

- OFFSET n : commencer à la solution n+1. OFFSET 0 n'a pas d'effet.
- LIMIT n : limite à n le nombre de solutions
- S'utilisent avec la clause ORDER BY

```
PREFIX foaf:    <http://xmlns.com/foaf/0.1/>
SELECT  ?name
WHERE   { ?x foaf:name ?name }
ORDER BY ?name
LIMIT   5
OFFSET  10
```

Le résultat de cette requête aura au maximum 5 solutions, à partir de la 11ème dans la séquence de solutions.

Fonctions

- Logique : !, &&, ||
- Math : +, -, *, /
- Comparaison : =, !=, >, <, …
- Tests (SPARQL) : **`isURI, isBlank, isLiteral, bound`**
- Autres (SPARQL) : **`str, lang, datatype`**
- **`sameTerm, langMatches, regex`**

Exemples de requêtes

Données :

```
@prefix dc: <http://purl.org/dc/elements/1.1/> .
@prefix : <http://example.org/book/> .
@prefix ns: <http://example.org/ns#> .
:book1 dc:title "SPARQL Tutorial" .
:book2 dc:title "The Semantic Web" .
:book2 ns:price 23 .
```

Quantification existentielle

- OPTIONAL + bound() permet d'exprimer la quantification existentielle
- **`Bound()`** renvoie **`true`** si la variable est liée, **`false`** sinon.

```
PREFIX dc: <http://purl.org/dc/elements/1.1/>
PREFIX ns: <http://example.org/ns#>
SELECT ?title ?price
WHERE { ?x dc:title ?title .
        OPTIONAL { ?x ns:price ?price } .
        FILTER ( bound(?price)) }
```

title	price
"The Semantic Web"	23

Quantification existentielle

- OPTIONAL + ! bound() permet de tester qu'un motif de graphe n'est pas exprimé.

```
PREFIX dc: <http://purl.org/dc/elements/1.1/>
PREFIX ns: <http://example.org/ns#>
SELECT ?title ?price
WHERE { ?x dc:title ?title .
        OPTIONAL { ?x ns:price ?price } .
        FILTER ( !bound(?price)) }
```

title	price
"SPARQL Tutorial"	

ASK

- Renvoie un booléen indiquant si un motif d'interrogation correspond ou non.

```
@prefix foaf:       <http://xmlns.com/foaf/0.1/> .
_:a  foaf:name       "Alice" .
_:a  foaf:homepage  <http://work.example.org/alice/> .
_:b  foaf:name       "Bob" .
_:b  foaf:mbox       <mailto:bob@work.example> .
```

Requêtes :

```
PREFIX foaf:    <http://xmlns.com/foaf/0.1/>
ASK  { ?x foaf:name  "Alice" }
```

renvoie `true`

```
PREFIX foaf:    <http://xmlns.com/foaf/0.1/>
ASK  { ?x foaf:name  "Alice" ;
foaf:mbox  <mailto:alice@work.example>}
```

renvoie `false`

CONTRUCT

- CONSTRUCT renvoie un graphe RDF décrit par un gabarit de graphe.

```
@prefix  foaf:  <http://xmlns.com/foaf/0.1/> .
_:a    foaf:name   "Alice" .
_:a    foaf:mbox   <mailto:alice@example.org> .

PREFIX foaf:    <http://xmlns.com/foaf/0.1/>
PREFIX vcard:   <http://www.w3.org/2001/vcard-rdf/3.0#>
CONSTRUCT {<http://example.org/person#Alice> vcard:FN ?name}
WHERE       { ?x foaf:name ?name }
```

Crée des propriétés vcard à partir des informations FOAF:

```
@prefix vcard: <http://www.w3.org/2001/vcard-rdf/3.0#> .
<http://example.org/person#Alice> vcard:FN "Alice" .
```

CONSTRUCT

- Il est possible de créer un graphe avec des nœuds anonymes

```
@prefix  foaf:  <http://xmlns.com/foaf/0.1/> .
_:a    foaf:givenname   "Alice" .
_:a    foaf:family_name "Hacker" .
_:b    foaf:firstname   "Bob" .
_:b    foaf:surname     "Hacker" .
```

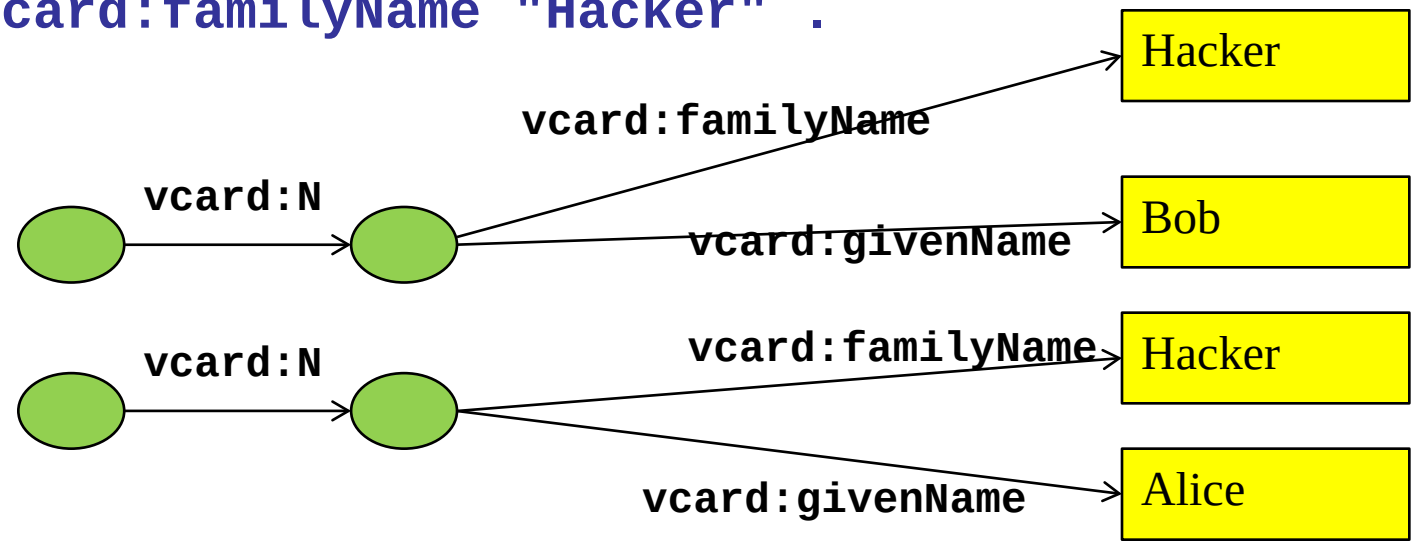
CONSTRUCT

```
PREFIX foaf:   <http://xmlns.com/foaf/0.1/>
PREFIX vcard: <http://www.w3.org/2001/vcard-rdf/3.0#>
CONSTRUCT { ?x  vcard:N _:v .
            _:v vcard:givenName ?gname .
            _:v vcard:familyName ?fname }
WHERE {
{?x foaf:firstname ?gname } UNION {?x foaf:givenname
  ?gname } .
{?x foaf:surname ?fname } UNION  {?x foaf:family_name
  ?fname } .
 }
```

CONSTRUCT

- Résultat:

```
@prefix vcard: <http://www.w3.org/2001/vcard-rdf/3.0#> .
_:v1 vcard:N          _:x .
_:x vcard:givenName   "Alice" .
_:x vcard:familyName "Hacker" .
_:v2 vcard:N          _:z .
_:z vcard:givenName   "Bob" .
_:z vcard:familyName "Hacker" .
```



Conclusion

- Interrogation de graphes sémantiques
  - Définition de motifs
  - Recherche de données correspondant au motif dans le graphe (appariement de graphes)
- Autres fonctionnalités
  - Règles d'inférence : pour déduire des informations
  - Nouveaux opérateurs et fonctions de filtre (négation, agrégats..) (SPARQL1.1)
  - Requêtes imbriquées
  - Chemins de propriétés
  - …

```
@prefix : <http://dbpedia.org/resource/> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
```

```
#actors
:ryan_gosling
   a foaf:actor ;
   rdfs:label "Ryan Gosling";
   :birthDate "1980-11-12" ;
   :countryOfBirth "Canada" .

:Christina_Hendricks
   a foaf:actor ;
   rdfs:label "Christina Hendricks" ;
   :birthDate "1975-05-03" .

:Carey_Mulligan
   a foaf:actor ;
   rdfs:label "Carey_Mulligan" ;
   :birthDate "1985-05-28" ;
   :countryOfBirth "UK".
```

```
#plays
:ryan_gosling
   :plays :drive ,
:gangster_squad .

:Christina_Hendricks
   :plays :drive ,
:madmen .

:Christina_Hendricks
   :plays
:gangster_squad ,
:never_let .
```

```
#movies
 :drive
   a :movie;
   :budget "12";
   :year "2012";
   :rating "7.8";
   :seen "352000".

:madmen
   a :movie;
   :budget "8";
   :year "2007";
   :rating "7.2";
   :seen "96000".

:gangster_squad
   a :movie;
   :budget "15";
   :year "2013";
   :rating "6.8";
   :seen "160000”.
```

Exemple 1

```
select ?actor ?movie where{ ?actor :plays ?movie. ?movie :budget "12"}
```

Exemple 2

```
select ?actor ?birthCount
where
{ ?actor :plays :drive .optional {?actor :countryOfBirth ?birthCount}}
```

Exemple 3

Les acteurs qui ont joué dans le même film.

Le résultat de la requête ne doit pas contenir de doublon :

actor1	actor2
:Christina_Hendricks	:ryan_gosling