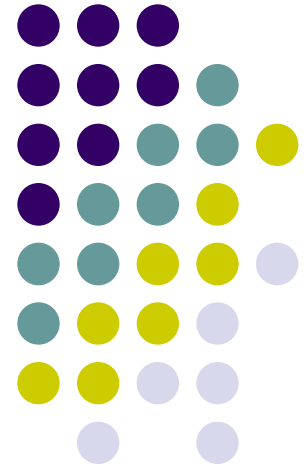
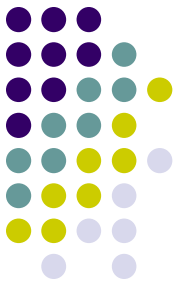


Chapitre III: Langages du Web Sémantique

III.3 SPARQL

SPARQL Protocol And RDF Query Language

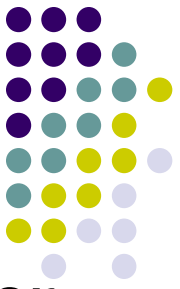




Plan du cours

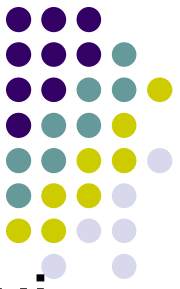
- Introduction
- Structure d'une requête SPARQL
- Requête simple
 - Préfix
 - Variables
 - Clause where(pattern)
- Filter
- Optional
- Union
- Séquence des solutions

Introduction



- SPARQL est le langage standard pour interroger les données de graphes représentés par des triplets RDF. ([rdf/rdfs](#))
- C'est devenu un standard du W3C en Janvier 2008.
- SPARQL 1.1 est pour l'instant une version de travail et donc au statut de brouillon.
- Pour ce cours :SPARQL 1.0 (2008)

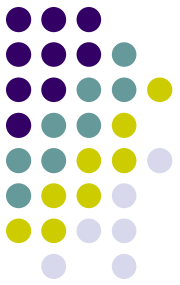
Introduction



- C'est un langage de requête et un protocole qui permettra de rechercher, d'ajouter, de modifier ou de supprimer des données [RDF/RDFS](#) disponible à travers Internet.
- SPARQL s'apparente à SQL qui comme SQL accède aux base de données via une langage de requête, mais SPARQL accède lui aux ressources présentes sur le Web.

Introduction

- Exemple

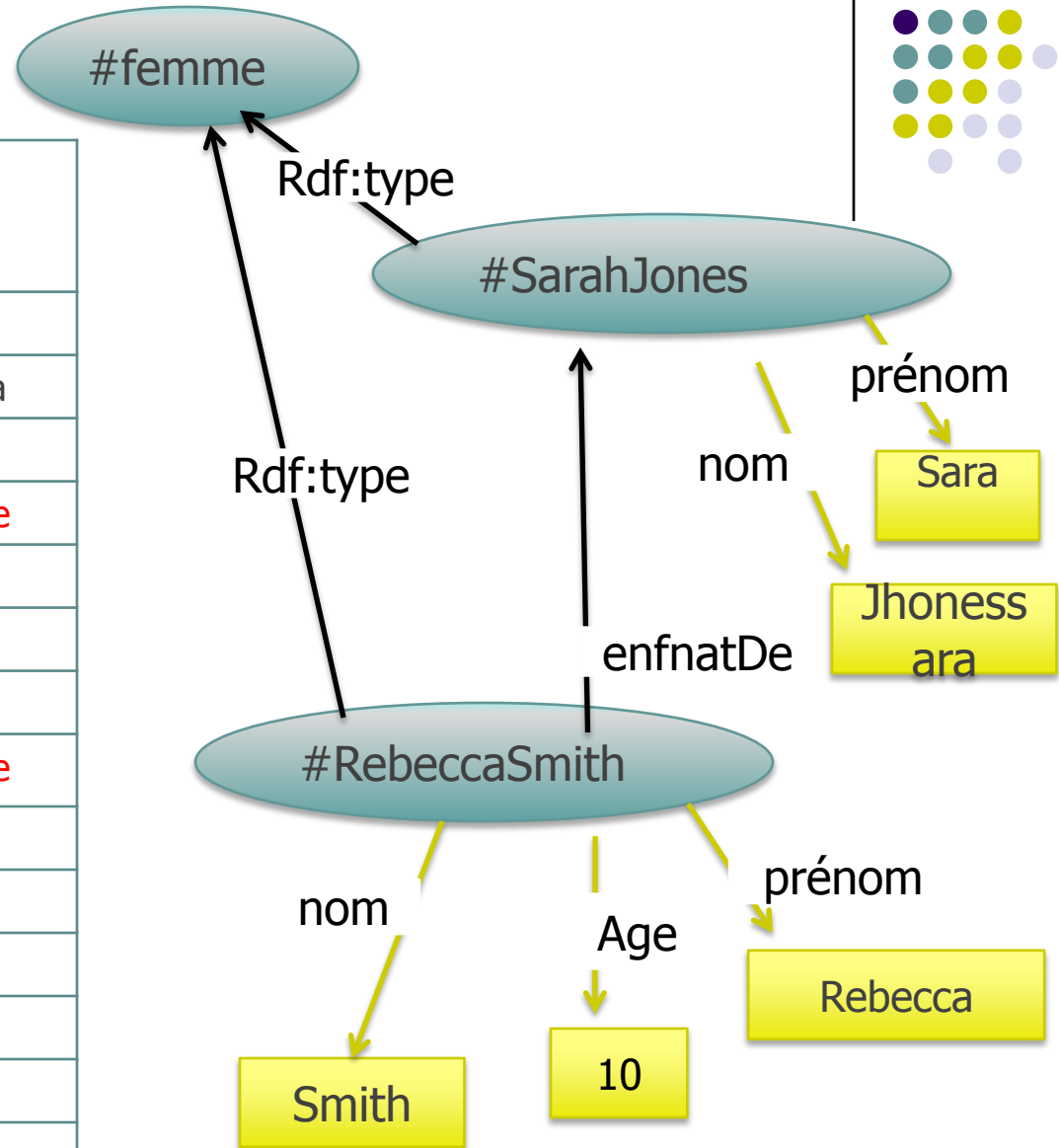


Uri	nom	Prénom	age	enfantDe
#RebeccaSmith	Smith	Rebecca	10	#JohnSmith
#SarahJones	Jones	Sarah		
#JohnSmith	Smith	John	25	
#MattJones	Jones	Matt	75	
#AnnaJones		anna		

Introduction

BT(base de triplets)

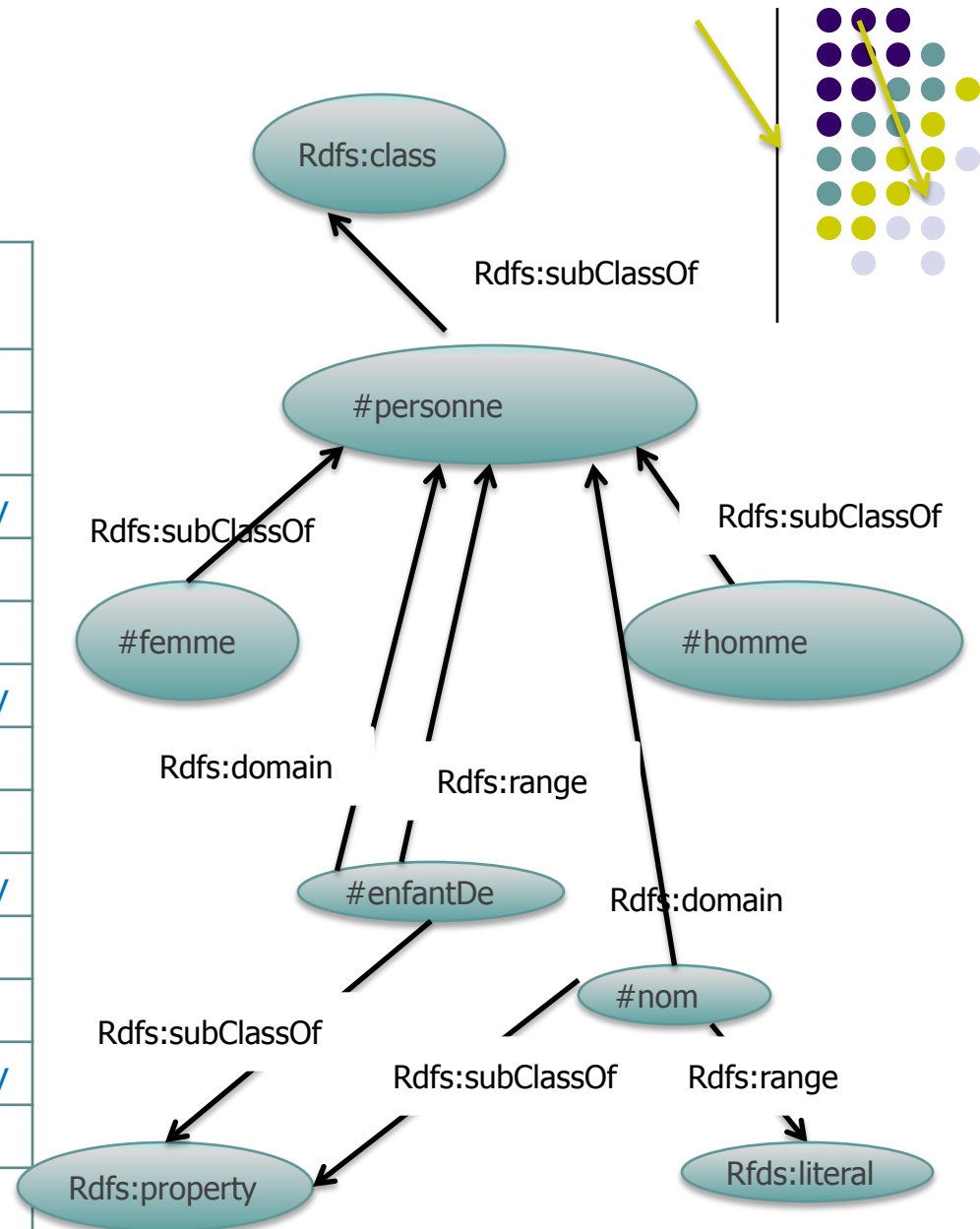
Sujet (uri)	Predicat (Propriété, Relation)	objet
#RebeccaSmith	nom	Smith
#RebeccaSmith	Prénom	Rebecca
#RebeccaSmith	age	10
#RebeccaSmith	Rdf:type	#femme
#SarahJones	nom	Jones
#SarahJones	Prénom	Sarah
#SarahJones	age	50
#SarahJones	Rdf:type	#femme
.....		



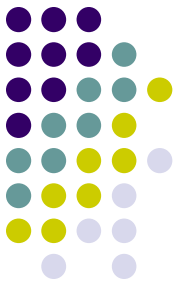
Introduction

BT(base de triplets)

Sujet (uri)	Predicat (Propriété,Relation)	objet
#personne	rdfs:subClassOf	rdfs:class
#femme	rdfs:subClassOf	#personne
#enfantDe	rdfs:subClassOf	rdfs:property
#enfantDe	Rdfs:domain	#personne
#enfantDe	Rdfs:range	#personne
#nom	rdfs:subClassOf	rdfs:property
#nom	Rdfs:domain	#personne
#nom	Rdfs:range	#personne
#prenom	rdfs:subClassOf	rdfs:property
#prenom	Rdfs:domain	#personne
#prenom	Rdfs:range	#personne
#age	rdfs:subClassOf	rdfs:property
#age	Rdfs:domain	#personne
#age	Rdfs:range	#personne



formes de requetes



- **select** :d'extraire des données spécifiques (valeurs, ressources, littéraux) à partir de jeux de données RDF, en les présentant sous forme de tableau de résultats
- **Construct**:générer un nouveau graphe RDF (un ensemble de triplets) à partir des données existantes, plutôt que de retourner une simple table de résultats comme SELECT
- **describe** a pour objectif principal d'extraire un sous-graphe RDF décrivant une ou plusieurs ressources (IRI) spécifiques sans nécessiter la connaissance précise de leur structure
- **Ask**: vérifier l'existence de données spécifiques dans un graphe RDF, en retournant une valeur booléenne simple : true (vrai)

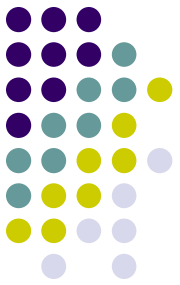
Structure d'une requête SPARQL

Requête simple

PREFIX :<[préfixes]>

SELECT [variables]

WHERE {[petterne]}



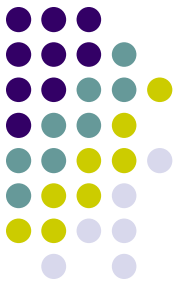
Préfixe : pour déclarer les préfixes utilisés dans la requête

SELECT [variables] : liste des variables du résultat

WHERE {[petterne]} : le (les paternes) Modèles de triplets RDF, c-a-d les conditions qui doivent être remplies (les triplets concernés)

Structure d'une requête SPARQL

Requête simple



Exemple: toutes les personnes(URIs) et leurs noms:
les ressources liées par **dc:nom** et **dc:prenom**

PREFIX dc:<http://www.univ-msila.dz/mi/inf/vocab#/>

SELECT **?prs ?nm**

Where { **?prs dc:nom ?nm.**
?prs dc:prenom ?prn.}

?prs	?nm
#RebeccaSmith	Smith
#SarahJones	Jones
#JohnSmith	Smith
#MattJones	Jones

Structure d'une requête SPARQL

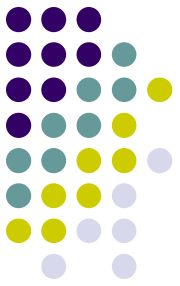
Préfix

- PREFIX permet de déclarer les espaces de nom utilisés dans la requête

PREFIX dc:<http://www.univ-msila.dz/mi/inf/vocab#/>

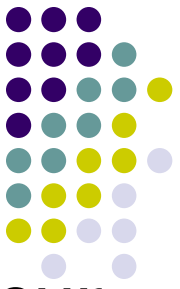
SELECT ?prs ?nm ?pren

Where { ?prs dc:nom ?nm.
 ?prs dc:prenom ?pren. }



Structure d'une requête SPARQL

Variables



- Les variables sont introduites par un identificateur préfixé par '?'
- Les variables ne sont pas des noms d'attributs
- Résultat: table de valeurs correspondant à (?var1 , ..., ?var_m)

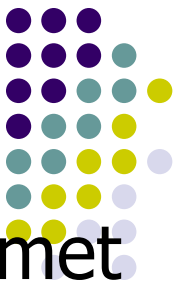
PREFIX dc:<http://www.univ-msila.dz/mi/inf/vocab#/>

SELECT ?prs ?nm ?prn

Where { ?prs dc:nom ?nm.
 ?prs dc:prenom ?prn. }

Structure d'une requête SPARQL

Clause WHERE



- constitue le cœur de la requête puisque elle permet de spécifier les conditions
- une condition prend la forme d'un pattern(triplets avec variables (sujet prédicat objet) pour permettre le filtrage des résultats par « pattern matching »
- La clause WHERE peut avoir plusieurs patterns dont il faut considérer leur conjonction.
- Un pattern doit se terminer par un point

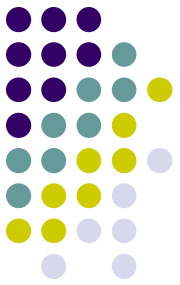
PREFIX dc:<http://www.univ-msila.dz/mi/inf/vocab#/>

SELECT ?prs ?prn

Where { ?prs dc:nom ?nm.
 ?prs dc:prenom ?prn. }

Structure d'une requête SPARQL

Clause WHERE



Pattern

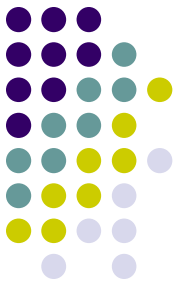
- Un pattern est un triplet ou plusieurs triplets de plusieurs variables afin de matérialiser plusieurs triplets (*.pdf, *.doc, k*a.pdf, tp.*)
- Exemple: le pattern : **?prs** **s:nom** "Smith"

Triplets correspondantes :

Uri	nom	Prenom	age
#RebeccaSmith	Smith	Rebecca	10
#JohnSmith	Smith	John	25

Structure d'une requête SPARQL

Clause WHERE

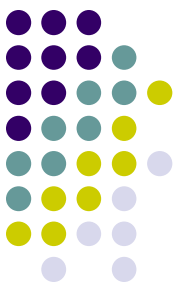


Pattern

Pattern	Triplets designés
?prs s:nom "Smith"	Les triplets ayant valeur nom=Smith
?prs s:prenom "Anna"	Les triplets ayant valeur prenom=Anna
?prs ?r "Smith"	Les triplets ayant valeur d'une propriété Smith
?prs s:nom ?n	Les triplets ayant une valeur pour nom
?prs s:nom ?n ?prs s:prenom ?pren	Les triplets ayant une valeur pour nom et prenom

Structure d'une requête SPARQL

Clause WHERE



Pattern matching

Sujet (uri)	Predicat (Propriété, Relation)	objet
#RebeccaSmith	nom	Smith
#RebeccaSmith	Prénom	Rebecca
#RebeccaSmith	age	10
#SarahJones	nom	Jones
#SarahJones	Prénom	Sarah
#JohnSmith	nom	Smith
#JohnSmith	Prénom	John
#JohnSmith	age	25
#MattJones	nom	Jones
#MattJones	Prénom	Matt
#MattJones	age	75

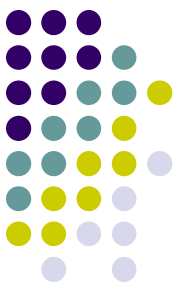
?prs

s:nom

"Smith"

Structure d'une requête SPARQL

Clause WHERE



Pattern matching

Sujet (uri)	Predicat (Propriété, Relation)	objet
#RebeccaSmith	nom	Smith
#RebeccaSmith	Prénom	Rebecca
#RebeccaSmith	age	10
#SarahJones	nom	Jones
#SarahJones	Prénom	Sarah
#JohnSmith	nom	Smith
#JohnSmith	Prénom	John
#JohnSmith	age	25
#MattJones	nom	Jones
#MattJones	Prénom	Matt
#MattJones	age	75

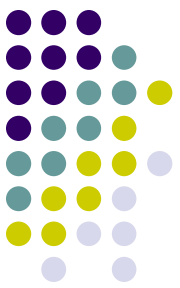
?prs

s:nom

"Smith"

Structure d'une requête SPARQL

Clause WHERE



Pattern matching

Sujet (uri)	Predicat (Propriété, Relation)	objet
#RebeccaSmith	nom	Smith
#RebeccaSmith	Prénom	Rebecca
#RebeccaSmith	age	10
#SarahJones	nom	Jones
#SarahJones	Prénom	Sarah
#JohnSmith	nom	Smith
#JohnSmith	Prénom	John
#JohnSmith	age	25
#MattJones	nom	Jones
#MattJones	Prénom	Matt
#MattJones	age	75

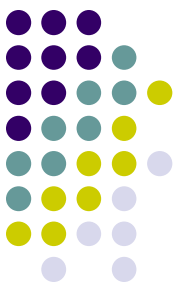
?prs

s:nom

"Smith"

Structure d'une requête SPARQL

Clause WHERE



Pattern matching

Sujet (uri)	Predicat (Propriété, Relation)	objet
#RebeccaSmith	nom	Smith
#RebeccaSmith	Prénom	Rebecca
#RebeccaSmith	age	10
#SarahJones	nom	Jones
#SarahJones	Prénom	Sarah
#JohnSmith	nom	Smith
#JohnSmith	Prénom	John
#JohnSmith	age	25
#MattJones	nom	Jones
#MattJones	Prénom	Matt
#MattJones	age	75

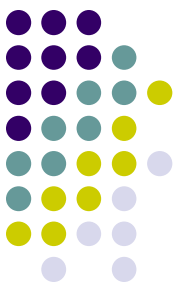
?prs

s:nom

"Smith"

Structure d'une requête SPARQL

Clause WHERE



Pattern matching

Sujet (uri)	Predicat (Propriété, Relation)	objet
#RebeccaSmith	nom	Smith
#RebeccaSmith	Prénom	Rebecca
#RebeccaSmith	age	10
#SarahJones	nom	Jones
#SarahJones	Prénom	Sarah
#JohnSmith	nom	Smith
#JohnSmith	Prénom	John
#JohnSmith	age	25
#MattJones	nom	Jones
#MattJones	Prénom	Matt
#MattJones	age	75

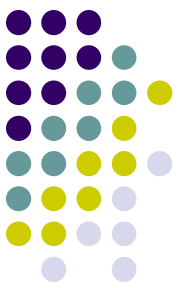
?prs

s:nom

"Smith"

Structure d'une requête SPARQL

Clause WHERE



Pattern matching

Sujet (uri)	Predicat (Propriété, Relation)	objet
#RebeccaSmith	nom	Smith
#RebeccaSmith	Prénom	Rebecca
#RebeccaSmith	age	10
#SarahJones	nom	Jones
#SarahJones	Prénom	Sarah
#JohnSmith	nom	Smith
#JohnSmith	Prénom	John
#JohnSmith	age	25
#MattJones	nom	Jones
#MattJones	Prénom	Matt
#MattJones	age	75

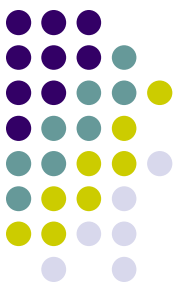
?prs

s:nom

"Smith"

Structure d'une requête SPARQL

Clause WHERE



Pattern matching

Sujet (uri)	Predicat (Propriété, Relation)	objet
#RebeccaSmith	nom	Smith
#RebeccaSmith	Prénom	Rebecca
#RebeccaSmith	age	10
#SarahJones	nom	Jones
#SarahJones	Prénom	Sarah
#JohnSmith	nom	Smith
#JohnSmith	Prénom	John
#JohnSmith	age	25
#MattJones	nom	Jones
#MattJones	Prénom	Matt
#MattJones	age	75

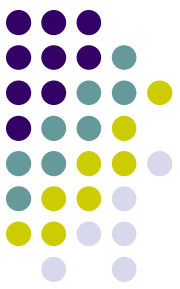
?prs

s:nom

"Smith"

Structure d'une requête SPARQL

Clause WHERE (éléments optionnels)



FILTER

- Pour ajouter des conditions à la requête
- Un filtre s'applique à un motif de graphe
- La position du filtre dans partie Where n'a pas d'importance

- Exemple :

```
SELECT ?prs ?nm ?prn ?a
```

```
Where { ? prs dc:prenom ?prn.
```

```
        ? prs dc:nom ?nm.
```

```
        ? prs dc:age ?a.
```

```
FILTER (?a>18) }
```

Structure d'une requête SPARQL

Clause WHERE (éléments optionnels)

FILTER

- Exemple :

SELECT ?prs ?nm ?prn ?a

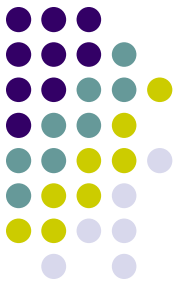
Where { ?prs dc:prenom ?prn.

?prs dc:nom ?nm.

?prs dc:age ?a.

FILTER (?a>18) }

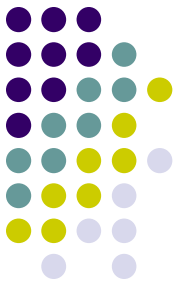
?pers	?nm	?prn	?a
#JohnSmith	Smith	John	25
#MattJones	Jones	Matt	27



Structure d'une requête SPARQL

Clause WHERE (éléments optionnels)

OPTIONAL



```
SELECT ?pers ?prn ?a
WHERE { ?pers s:prenom ?prn .
        ?pers s:age ?a. }
```

Cette requête retourne les prenom des personnes ayant un prenom et un age (celles qui n'ont pas d'age ne sont prises en compte)

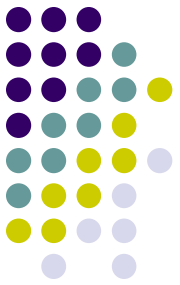
?prs	? prn	?a
#RebeccaSmith	Rebecca	10
#JohnSmith	John	25
#MattJones	Matt	27

#SaraJones	age	?
------------	-----	---

Structure d'une requête SPARQL

Clause WHERE (éléments optionnels)

OPTIONAL

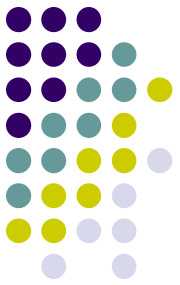


- Permet de spécifier un deuxième pattern optionnel.
- Permet d'éviter problème de conjonction dans les patterns de la cause where et permet de prendre en compte les solutions n'ayant pas de valuation (valeur) pour certains patterns
- **SELECT ?var1,?var 2.....**
WHERE { [patterns basiques].
OPTIONAL { [patterns optionels]} }

Structure d'une requête SPARQL

Clause WHERE (éléments optionnels)

OPTIONAL



SELECT ?prn ?a

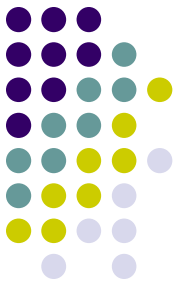
WHERE { ?pers s:prenom ?prn.

OPTIONAL { ?pers s:age ?a } }

Donne toutes les personnes ayant un prenom (age facultatif)

?pers	? prn	?a
#RebeccaSmith	Rebecca	10
#SarahJones	Sarah	
#JohnSmith	John	25
#MattJones	Matt	27

Structure d'une requête SPARQL



Clause WHERE

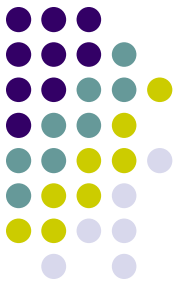
La fonction BOUND()

- Cette fonction vérifie si une variable a déjà été assignée (liée) à une valeur dans une partie précédente de la requête
- Elle renvoie *true* si la variable est liée, sinon *false*. Elle est essentielle pour gérer les données optionnelles.
- Role
 - Vérifie l'existence d'une valeur pour une variable (?var).
 - Utilisée principalement dans la clause FILTER.
 - Permet de différencier les résultats où une donnée optionnelle est présente de ceux où elle est absente.

Structure d'une requête SPARQL

Clause WHERE

La fonction BOUND()



```
SELECT ?prn ?a
```

```
WHERE { ?pers s:prenom ?prn.
```

```
OPTIONAL { ?pers s:age ?a }
```

```
FILTER (bound(?a) }
```

Donne toutes les personnes ayant un prenom et age)

```
SELECT ?prn ?a
```

```
WHERE { ?pers s:prenom ?prn.
```

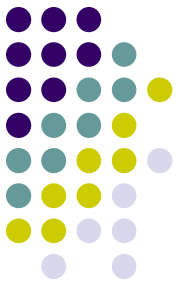
```
OPTIONAL { ?pers s:age ?a }
```

```
FILTER (!bound(?a) }
```

Donne toutes les personnes ayant un prenom et age)

Structure d'une requête SPARQL

Clause WHERE

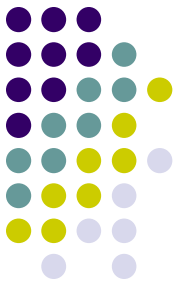


Union

- Permet de satisfaire un motif OU un autre.
- On peut différencier les résultats des deux réponses, en utilisant des variables différentes dans les motifs.
- L'opérateur d'union n'enlève pas les doublons

Structure d'une requête SPARQL

Clause WHERE



Union

SELECT ?nm

WHERE {

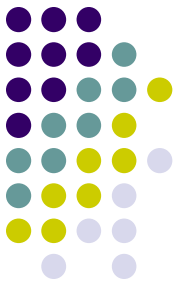
{ ?prs dc:nom ?nm.
?prs dc:age ?a.
Filter {?a<15} }

UNION {

?prs dc:nom ?nm.
?prs dc:Age ?a.
Filter {?a>25}}

Structure d'une requête SPARQL

Interrogation RDFS



- Toutes les classes

```
SELECT ?cls
```

```
WHERE{?cls rdfs:subclassOf rdfs:class.}
```

- Toutes les propriétés

```
SELECT ?prop
```

```
WHERE{?prop rdfs:subpropertyOf rdfs:property.}
```


Structure d'une requête SPARQL

Interrogation RDFS

- Les classes meres de la classe 'femme'

```
SELECT ?classMere
```

```
WHERE{ :femme rdfs:subclassOf ? classMere.}
```

- Les classes filles de la classe 'personne'

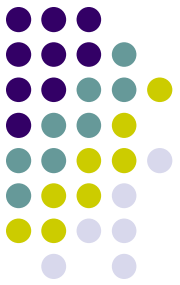
```
SELECT ?sousClasses
```

```
WHERE{?sousClasses rdfs:subclassOf :personne.}
```

- Le domaine et le rang de la propriété 'enfntatDe'

```
SELECT ?rng ?dm
```

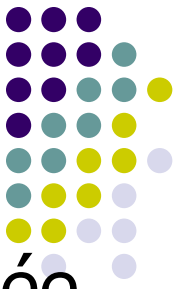
```
WHERE{ :enfntatDe rdfs:range ?rng.  
       :enfntatDe rdfs:domain ?dmn.}
```



Structure d'une requête SPARQL

Modifier la séquence des solutions

- les motifs génèrent une collection non-ordonnée de solutions. Ces solutions sont ensuite traitées comme une séquence, sur laquelle on peut appliquer un opérateur (séquence modifier).
- Il existe 4 opérateurs permettant de modifier la séquence des solutions :
 - 1. **order by** : trier les solutions
 - 3. **distinct** : éliminer les doublons parmi les solutions
 - 5. **offset** : indiquer la position de démarrage.
 - 6. **limit** : borner la taille de la séquence des solutions



Structure d'une requête SPARQL

Modifier la séquence des solutions

ORDER BY

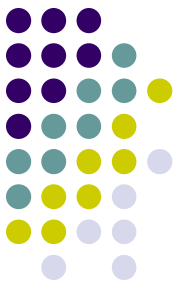
- pour trier les solutions
- il faut qu'une relation d'ordre existe pour ces solutions.
- 'DESC' et 'ASC' pour tri descendant et ascendant.

- Exemples

```
SELECT ?nm
```

```
WHERE { ?x dc:nom ?nm }
```

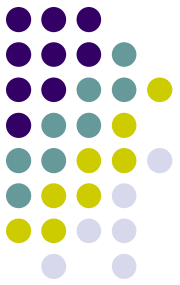
```
ORDER BY DESC(?nm)
```



Structure d'une requête SPARQL

Modifier la séquence des solutions

OFFSET et LIMIT



- **OFFSET n** : signifie qu'on "passe" les n premières solutions. Un offset de 0 n'a pas d'effet.
- **LIMIT n** : signifie qu'on donne au maximum n solutions.
- Exemples: au maximum 5 solutions, à partir de la 11eme dans la séquence des solutions.

```
SELECT ?nm
```

```
WHERE { ?x dc:nom ?nm }
```

```
LIMIT 5
```

```
OFFSET 10
```



Structure d'une requête SPARQL

Modifier la séquence des solutions

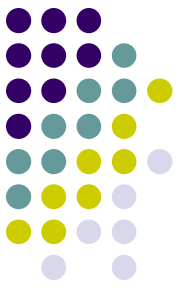
DISTINCT

- pour éliminer les doublons parmi les solutions

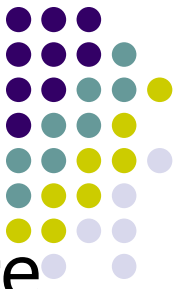
- Exemple

```
SELECT DISTINCT ?nm
```

```
WHERE { ?x dc:nom ?nm }
```



La requête SPARQL ASK



- ASK : retourne un booléen indiquant l'existence d'une solution qui satisfait le motif recherché

- Exemple :

ASK

WHERE { ?x dc:nom ?nm }

- Resultat :True