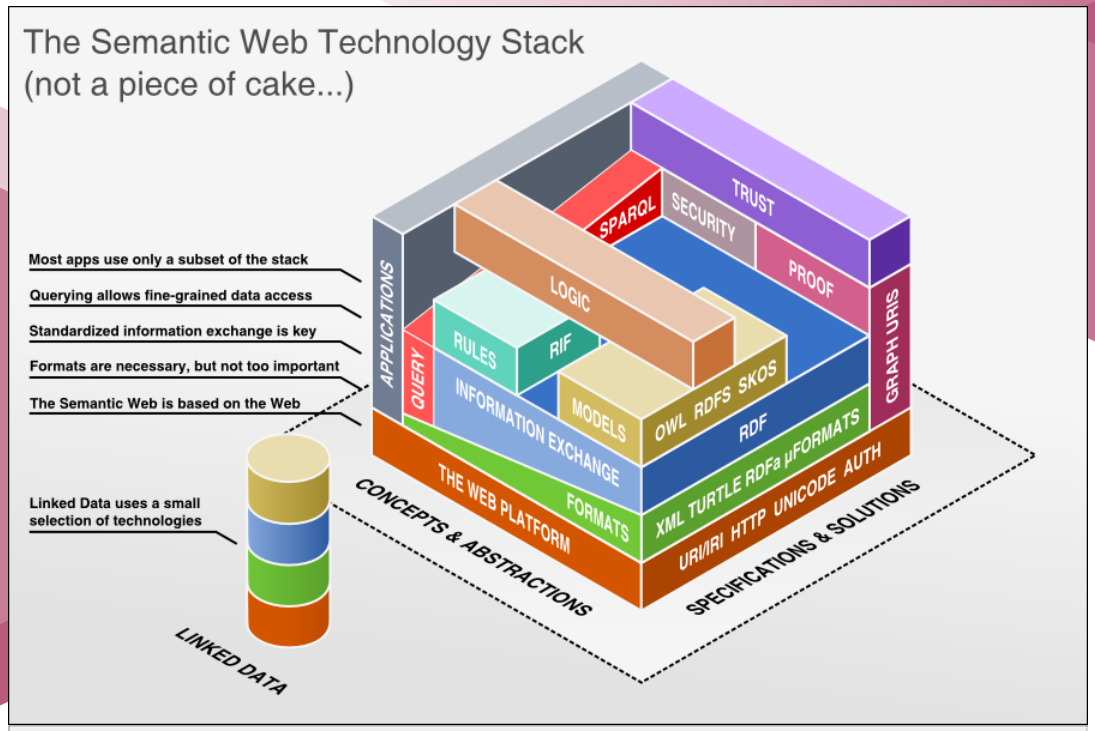


Partie 3 : Interroger le Web

Sylvie Calabretto, Mehdi Kaytoue et Aimene Belfodil

Rappel

- **RDF :** un cadre de modélisation de données dans le **Web 3.0**
- **Triplet RDF :** atome d'information : (**sujet**, **prédicat**, **objet**) avec **sujet** et **prédicat** sont des **ressources identifiées** par un **URI** et **objet** pouvant être une **ressource** ou un **littéral**.
- **Modèle simple :** Triplets/Graphes. On parle de **Triple store**.
- **Infrastructure du Web** : Utilisation des URI et espace de nommage
- **Syntaxes concrètes de RDF** indépendantes du **modèle abstrait**
- **Publication facile de données RDF:** via RDFa par exemple.
- Notion de **déréférencement** de ressources et **négociation de contenu** (Distinction entre **URI (IRI)**, **URL** et **URN**). Les URI sont par **définition sensible à la casse** (i.e. **case-sensitive**).
- **Fusion des données aisée :** agrégation de triplets.
- **Slogan AAA : A**nyone can say **A**nything about **A**nything.
- **Monde ouvert** et **logique monotone**.

Aujourd'hui – Interroger les données du Web 3.0

Interrogation des données avec un standard : SPARQL

- SPARQL - SPARQL Protocol And RDF Query Language
- Les requêtes suivent la syntaxe concrète RDF Turtle
- Il y a eu deux recommandations: SPARQL 1.0 (2008) et SPARQL 1.1 (2013).
- Des liens forts avec SQL et XPATH vus en 3IFA

Mais surtout SPARQL est :

- Un langage de requêtes http://www.w3.org/TR/rdf-sparql-query/
- Un protocole http://www.w3.org/TR/rdf-sparql-protocol/ pour soumettre une requête à un serveur distant et récupérer la réponse
- Un format de résultat (JSON, XML, CSV, etc.)

  JSON http://www.w3.org/TR/rdf-sparql-json-res/

  XML http://www.w3.org/TR/rdf-sparql-XMLres/

Plan du cours

Vers un accès à toutes les données du Web !

1. Rappel Syntaxe Turtle
2. Intuition : Appariement de graphe
3. Éléments de syntaxe de base pour les requêtes SELECT
4. Pré- et post- traitements de requêtes
5. Autres types de requêtes
6. Points d'accès, protocoles et réponses
7. Epilogue

INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES LYON

1. Syntaxe Turtle

Syntaxe Turtle (1)

```
@base <http://example.org/> .
@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/> .
@prefix rel: <http://www.perceive.net/schemas/relationship/> .
```

Définition d'URI de base

Définition de qname

Utilisation de @base, le vrai URI est <http://example.org/#aimene>

```
<#aimene>
     rel:siblingOf <#adnene> ;
     a foaf:Person ; # comment
      foaf:name "Aimene Belfodil" .
<#adnene>
      rel:siblingOf <#aimene> ;
      a foaf:Person ;
      foaf:name "Adnene Belfodil", "بلفضيل عدنان"@ar .
```

Liste de prédicats de même sujet séparés par ;

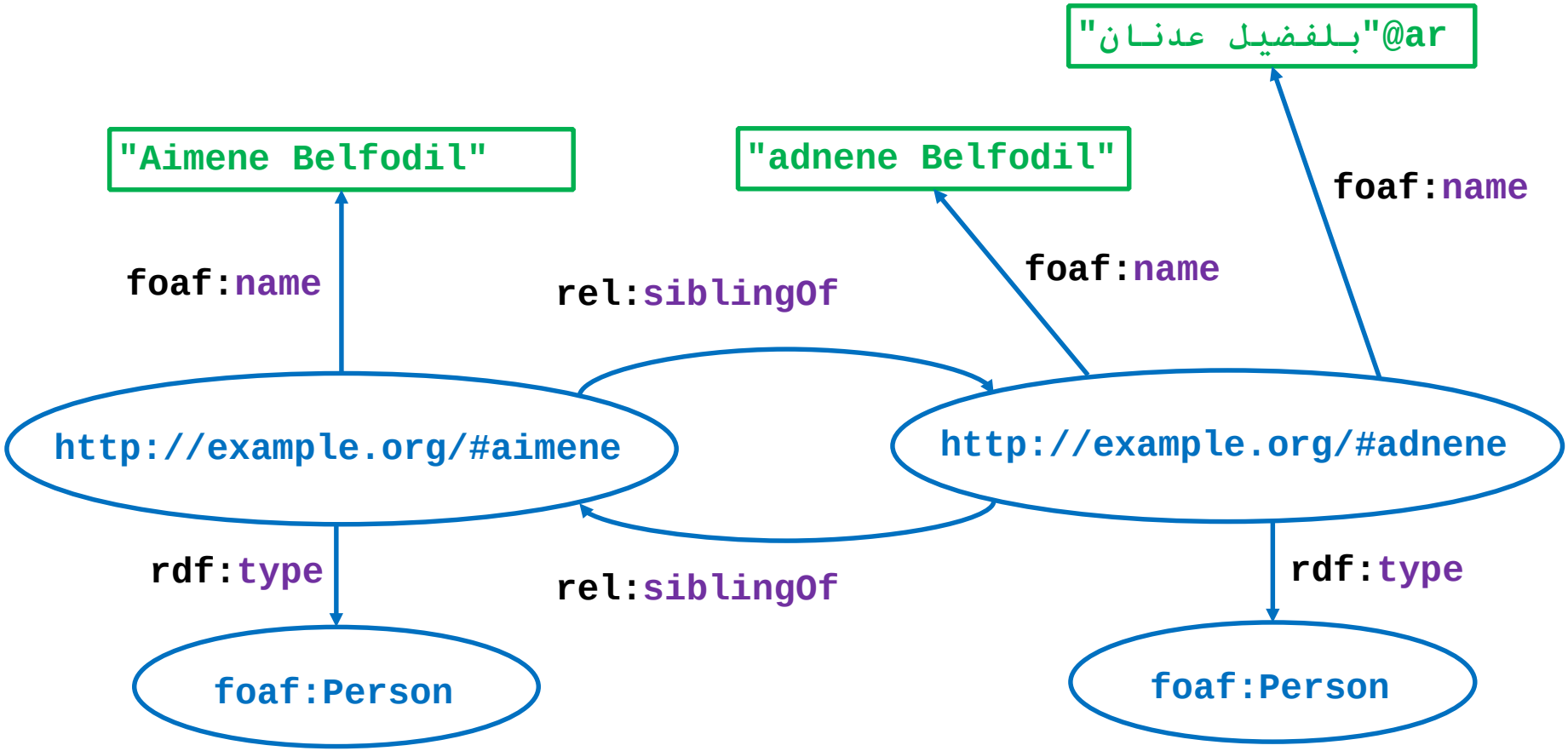
Liste de triplets séparés par .

Liste d'objets de même (sujet, prédicat) séparés par ,

≡ *rdf:type*

Exercice: *Dessiner le graphe RDF associé*

Syntaxe Turtle (2)



Recommandation : https://www.w3.org/TR/turtle/

Transformation entre syntaxes : http://www.easyrdf.org/converter

INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES LYON

2. Intuition

Morphologie d'une requête

Symboles précédés par « ? » sont des variables libres :

- Dis moi tout (extraire tous les triplets de la base) :

```
SELECT * WHERE {
    ?s ?p ?o .
}
```

- Dis moi tout au sujet de Lyon (extraire les propriétés de Lyon et leurs valeurs):

```
PREFIX : <http://dbpedia.org/resource/>
SELECT * WHERE {
    :Lyon ?p ?o .
}
```

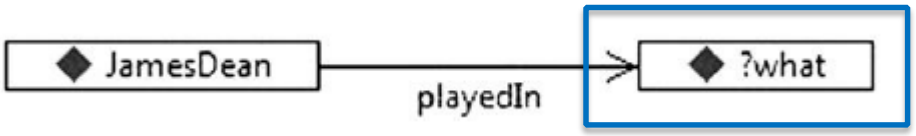
Des appariements de graphes (1)

Ecrire une requête, c'est écrire un graphe requête

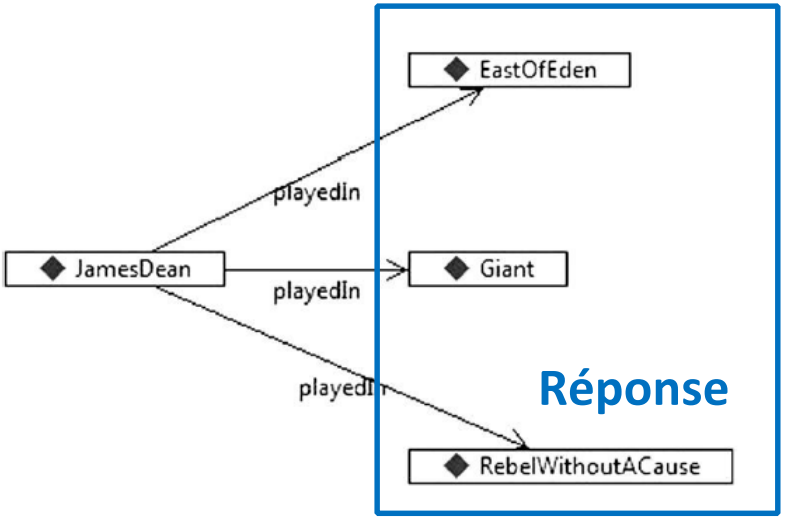
- On recherche ses occurrences dans un **graphe cible (les données)**
- **Appariement de graphes (graph pattern matching) :** on cherche tous les **sous-graphes** qui correspondent au **patron du graphe** donné par la requête

```
Q. SELECT ?what WHERE {:JamesDean :playedIn ?what .}
A. :Giant, :EastOfEden, :RebelWithoutCause .
```

Graphe requête



Variable libre

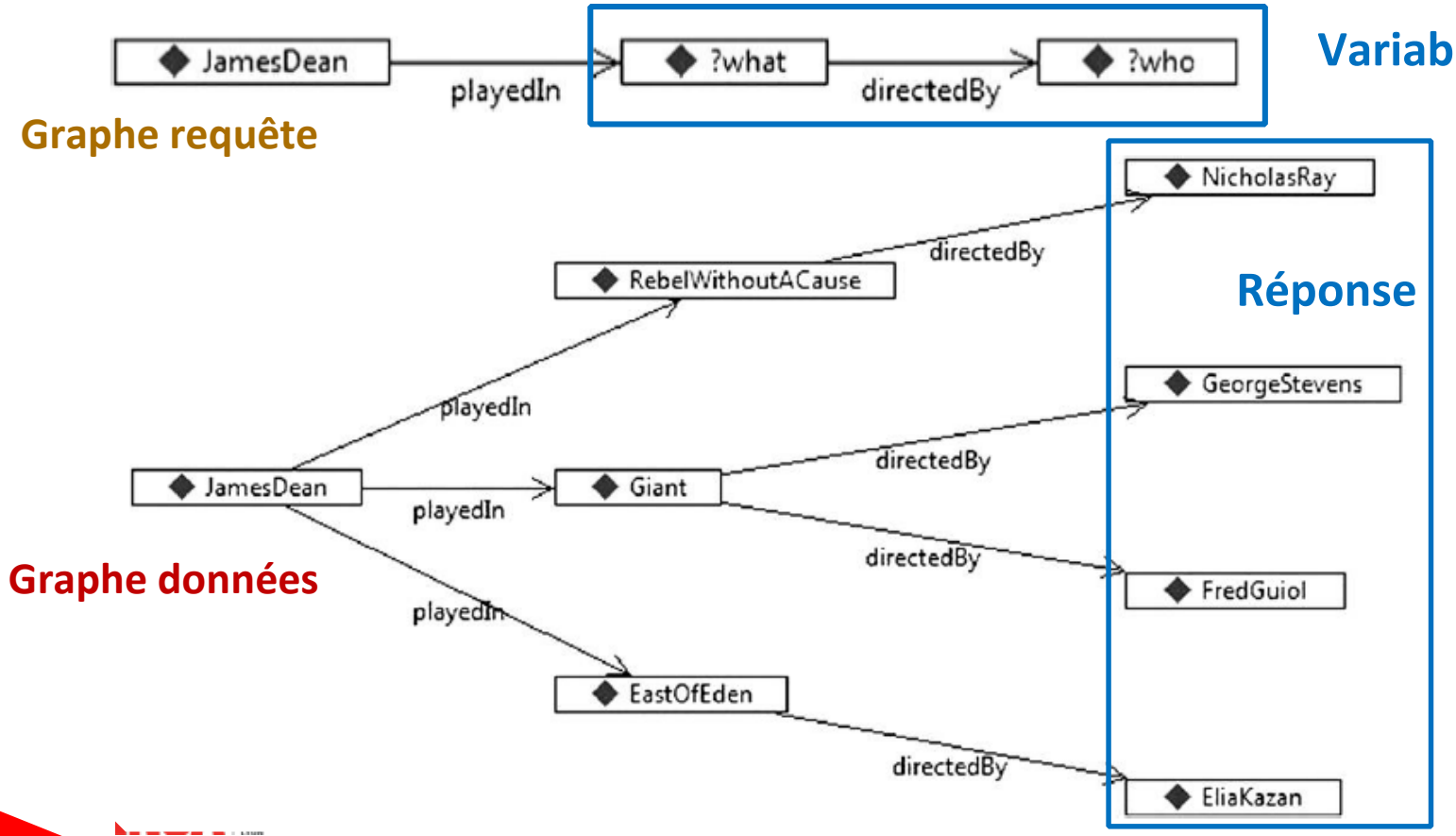


Réponse

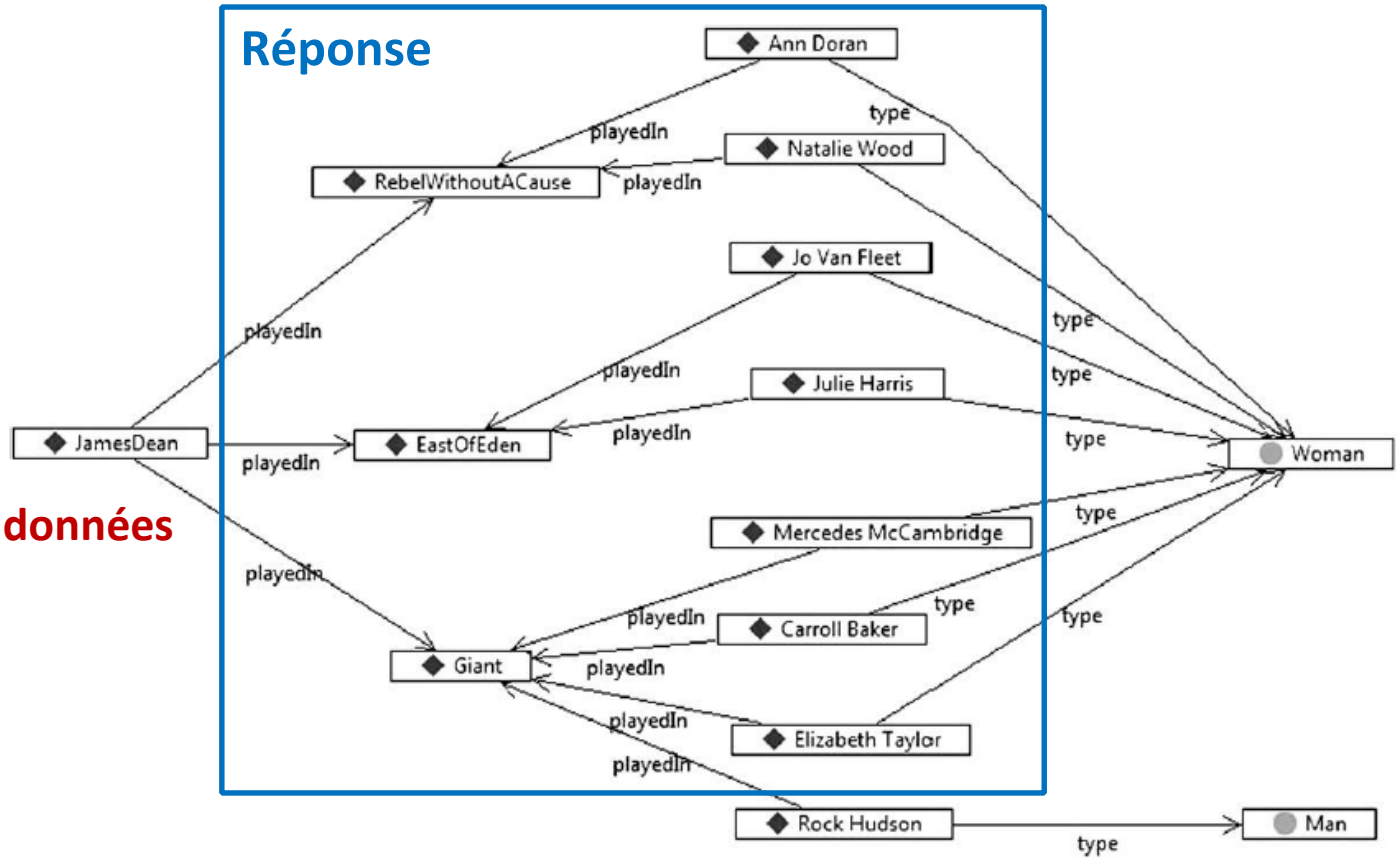
Graphe données

Des appariements de graphes (2)

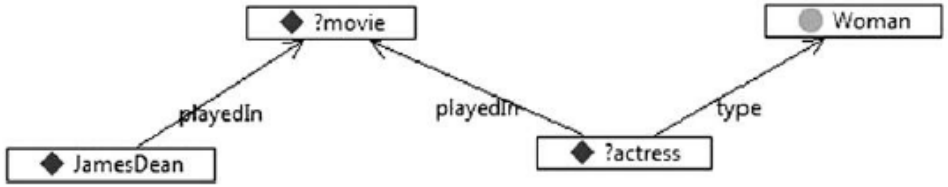
```
Q. SELECT ?who WHERE {:JamesDean :playedIn ?what .
                      ?what :directedBy ?who .}

A. :GeorgeStevens, :EliaKazan, :NicholasRay, :FredGuiol
```



Des appariements de graphes (3)




```
SELECT ?actress ?movie
WHERE {:JamesDean :playedIn ?movie .
       ?actress :playedIn ?movie ;
                rdf:type :Woman.}
```

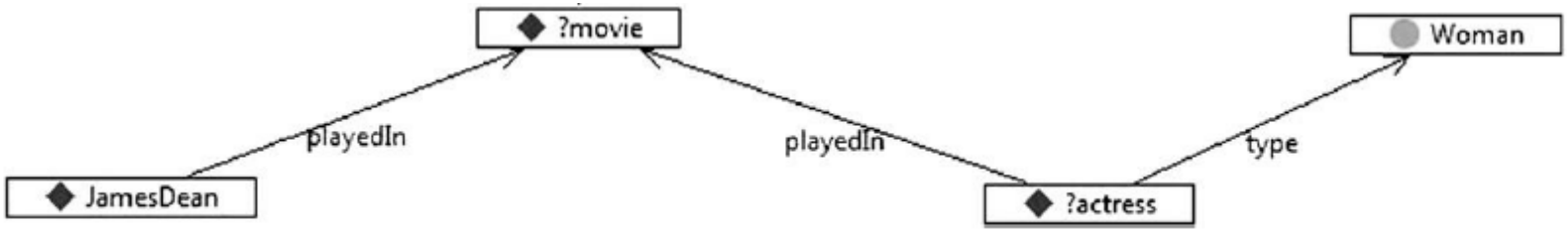


Graphe requête

Des appariements de graphes (3)

- Le nom de variable ?movie est arbitraire

```
SELECT ?director
WHERE {:JamesDean :playedIn ?movie .
       ?movie :director ?director .}
```

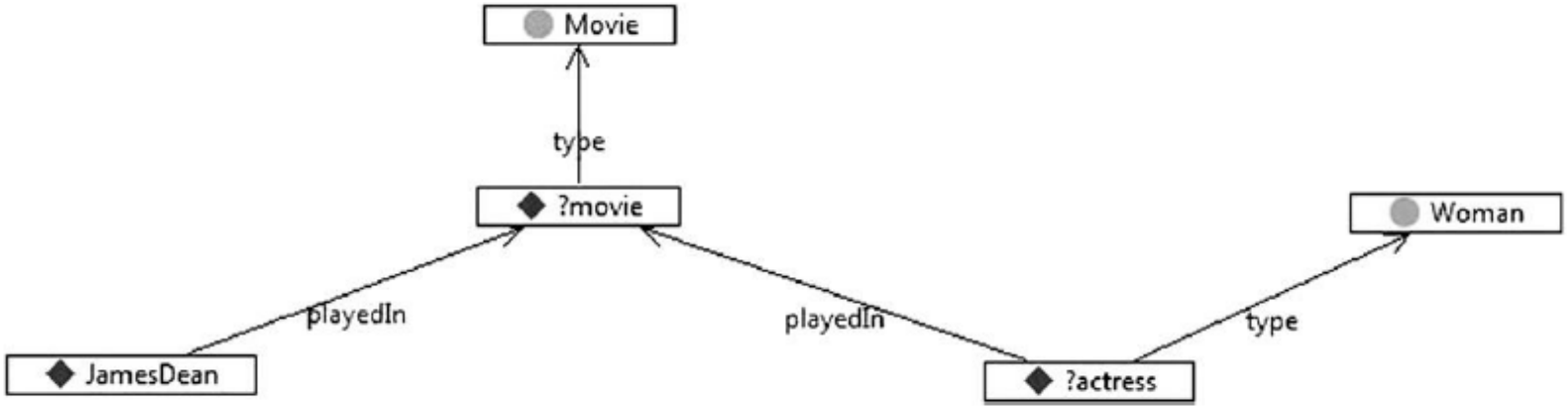


Graphe requête

Des appariements de graphes (3)

- Le nom de variable ?movie est arbitraire
- Il faut contraindre le type de ?movie

```
SELECT ?director
WHERE {:JamesDean :playedIn ?movie .
       ?movie :director ?director ;
              rdf:type :Movie.}
```



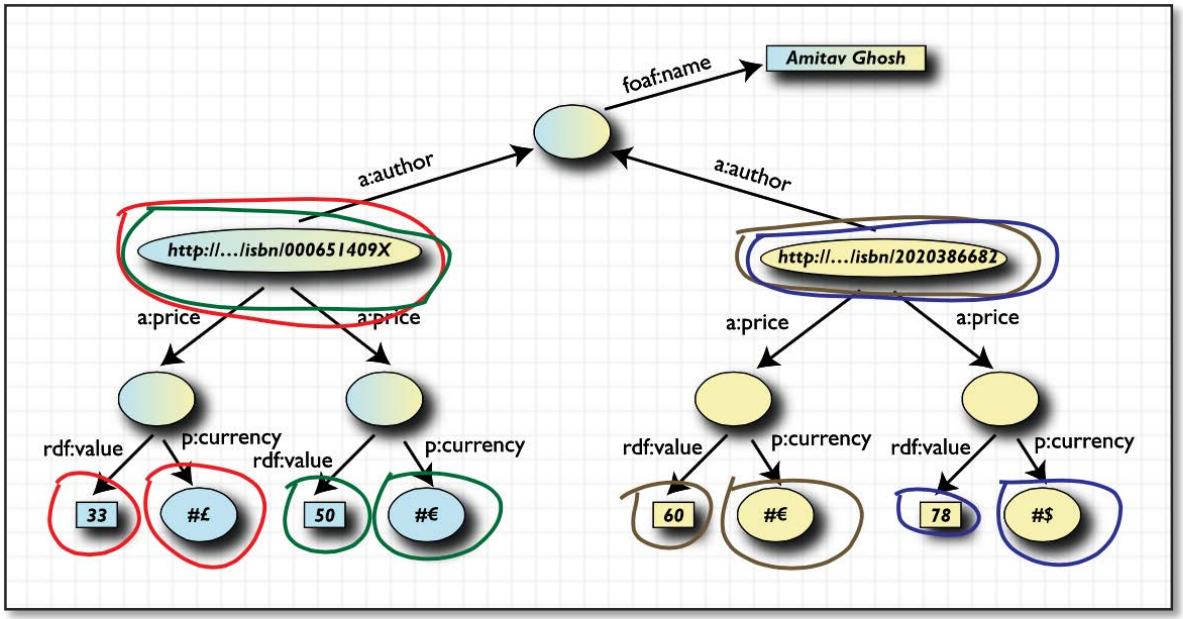
Graphe requête

Des appariements de graphes (4)

Autant de résultats que d'appariements possibles

```
SELECT ?isbn ?price ?currency
WHERE {:isbn a:price ?x .
       ?x rdf:value ?price ;
          p:currency ?currency .}
```

?isbn	?price	?currency
<http://... 409X>	33	#£
<http://... 409X>	50	#€
<http://... 6682>	60	#€
<http://... 6682>	78	#$



INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES LYON

2. Éléments de base

SPARQL – éléments de base

- Les variables libres sont précédés par '?' . Par exemple: ?x, ?y, ..
- Un Motif/patron de graphe est une conjonction de triplets avec « . » :

```
SELECT ?p, ?name
WHERE {?p a foaf:person .
       ?p foaf:name ?name .}
```

- Le keyword a est équivalent à rdf:type
- Factorisation pour un même sujet « ; » et factorisation pour un même couple (propriété, valeurs) avec « , »:

```
SELECT *
WHERE {?p a foaf:person ;
          foaf:name ?name .}
```

- '*' sélectionne toutes les variables libres dans le graphe résultant.
- Précéder * par DISTINCT permet d'enlever les doublons.

SPARQL – Préfixe, base et types

- Déclarer et utiliser le **préfixe** d' un **espace de nom**

```
PREFIX res: <http://dbpedia.org/resource/>
SELECT * WHERE { ?s ?o res:Barack_Obama . }
```

- Déclarer et utiliser un **URI de base**

```
BASE <http://dbpedia.org/resource/>
SELECT * WHERE { ?s ?o <Barack_Obama> . }
```

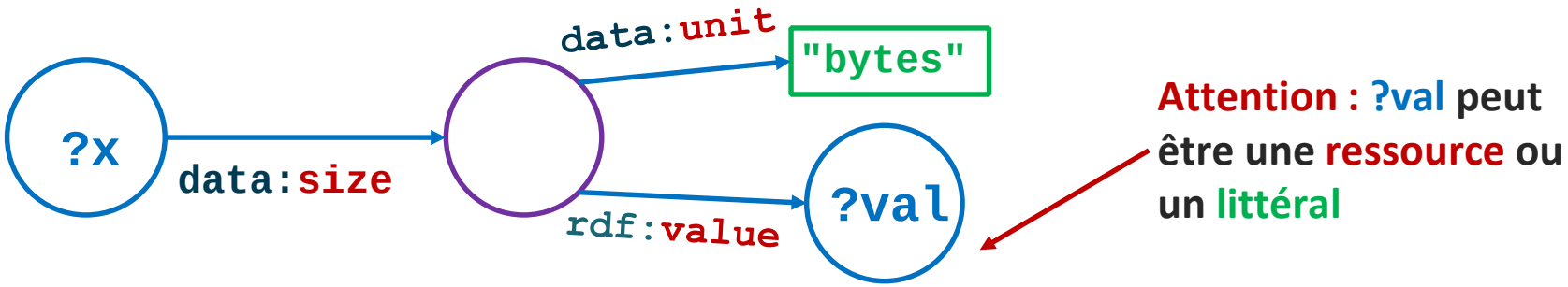
- Spécifier le **type** et la **langue** des **littéraux**

```
SELECT ?s,?p WHERE
{ ?s [] "Frankreich"@de . ?s ?p "50"^^xsd:integer .}
```

SPARQL – Ressources anonymes

- Utilisation de **ressources anonymes** :

```
SELECT * WHERE {
    ?x data:size [ rdf:value ?val; data:unit “bytes"]
}
```



- Cette requête est équivalente à :

```
SELECT * WHERE {
    ?x data:size _:n .
    _:n rdf:value ?val; data:unit “bytes"
}
```

SPARQL – Collections fermées

- **Syntaxe synthétique pour représenter les listes fermées.**
- Exemple : *On cherche les livres qui ont exactement 3 auteurs.*

```
SELECT * WHERE {
    ?x dc:auteur ( ?a ?b ?c ) }
```

- **Exercice: Dessiner le graphe d'appariement**
- Cette requête est équivalente à :

```
SELECT * WHERE {
    ?x    dc:auteur _:b1 .
    _:b1 rdf:first ?a ;
          rdf:rest  _:b2 .
    _:b2 rdf:first ?b ;
          rdf:rest  _:b3 .
    _:b3 rdf:first ?c ;
          rdf:rest  rdf:nil .}
```

Attention !

Le graphe correspondrait plutôt à cette requête

SPARQL – Motifs optionnels

- Un **motif optionnel** est une partie de graphe **non obligatoire:** on autorise le fait qu'une variable ne soit pas liée.

```
BASE <http://dbpedia.org/resource/>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?p, ?homepage
WHERE {
  ?p ?s <France> ;
      a foaf:Person .
  OPTIONAL { ?p foaf:homepage ?homepage .}
}
```

SPARQL – Le OU

Nous disposons du ET, mais qu'en est-il du OU ?

- UNION combine deux motifs graphes
- Les variables de chaque motif sont indépendantes, mais les résultats sont combinés.
- Exemple : *Les « personnes » nées à Brooklyn ou de prénom anglais John:*

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX dbo: <http://dbpedia.org/ontology/>
PREFIX dbr: <http://dbpedia.org/resource/>
SELECT DISTINCT ?p
where {
 { ?p dbo:birthPlace dbr:Brooklyn . }
UNION
 { ?p foaf:givenName "John"@en .}
}
```

SPARQL – chemins

Requêtes transitives

- Répétition : + (1 -) ou * (0 -)
- Répétition bornée : {n,m} ou {n}
- Optionnel : ?

- Séquence : /
- Alternative : |
- inverse : ^ (avant le prédicat)
- négation : ! (avant le prédicat)

- *Exemple : Quelles sont les classes parentes de Hôpital ?*

```
SELECT * WHERE { dbo:Hospital rdfs:subClassOf+ ?type .}
```

- *Exercice : Que fait cette requête ?*

```
SELECT ?value WHERE {
   ex:book dc:auteur ?list .
   ?list rdf:rest*/rdf:first ?value .
}
```

SPARQL – La Négation (1)

- **Deux formes de négation :**
- **MINUS** permet de supprimer des résultats d'une première expression les résultats d'une deuxième expression lorsque toutes les variables communes ont la même valeur dans les deux résultats.
- Exemple : *Les endroits peuplés de France sauf ceux du Grand-Est*

```
SELECT DISTINCT ?p WHERE {
  ?p a dbo:PopulatedPlace;
     dbo:country dbr:France;
     dbo:region [] .
MINUS
  { ?p dbo:region dbr:Grand_Est .}}
```

SPARQL – La Négation (2)

- **Deux formes de négation:**
- **FILTER NOT EXISTS** se présente sous forme d'un filtre qui permet de tester la **non-existence d'un pattern** dans le graphe cible.
- Exemple : *Les endroits peuplés de France **sauf ceux** du Grand-Est*

```
SELECT DISTINCT ?p WHERE {
  ?p a dbo:PopulatedPlace;
     dbo:country dbr:France;
     dbo:region [] .
FILTER NOT EXISTS
  { ?p dbo:region dbr:Grand_Est .}}
```

- Quelques différences subtiles entre les **deux formes de négation** sont détaillées dans la recommandation SPARQL 1.1:

  https://www.w3.org/TR/sparql11-query/#neg-notexists-minus

Aspect déclaratif vs aspect procédural

La clause WHERE fournit un nombre de contraintes sous la forme de triplets avec variables, URI et littéraux :

- **Aspect déclaratif:** Peut importe l'ordre, les résultats sont les mêmes : une conjonction !
- **Aspect Procédural:** Les contraintes sont listées dans un ordre et évaluées dans celui-ci par la plupart des moteurs **(mais pas tous).** Ainsi, veiller à introduire le moins de variables possibles au début de la requête.

```
SELECT ?actor
WHERE {
    ?actor ^dbo:starring ?movie .
    dbr:Uma_Thurman ^dbo:starring ?movie .
    ?movie dbo:director dbr:Quentin_Tarantino .
}
```

Placer la **règle la moins contraignante** à la **fin de la requête**.

INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES LYON

3. Pré- et Post-traitements

SPARQL - Filtres

- **Post-traitement :** contrainte **FILTER** à utiliser dans une clause **WHERE**
- **Exemple :** *Les présidents français nés après 1950*

```
SELECT * WHERE {
  ?p dbp:title dbr:President_of_France;
     dbo:birthDate ?d .
     FILTER (?d >= "1950-01-01"^^xsd:date )
}
```

- *Utilisation des comparateurs* `=, >, >=, <, <=, !=`

SPARQL – Fonctions (1)

Expression régulière

- Utilisation des expressions régulières :

```
SELECT * WHERE {
    ?p dbp:name ?n .
    FILTER ( regex(?n, ".*Jacques.*") ) }
```

Combinaison de tests :

- Connecteurs `&&, ||, !, ()`
- Branchements conditionnels IF/THEN/ELSE (i.e. condition ternaire)

```
FILTER(IF(langMatches(lang(?name),"FR"),?age>=18, ?age>=21))
```

- Appartenance dans une liste

```
FILTER (?n NOT IN ("N", "S", "E","O"))
```

- Absence d'un motif :

```
FILTER NOT EXISTS { ?x foaf:age -1 }
```

SPARQL – Fonctions (2)

- **Tests de types, de langue, etc. :**

```
isURI(?x), isLiteral(?x),  isBlank(?x), lang(?x)='en'
datatype(?x)=xsd:string,  isNumeric(?x), …
```

- **Transformation de types (i.e. cast):**

```
xsd:string(?x) (ou str (?x)),
xsd:integer(?x) (ou int (?x)), …
```

- **D'autres fonctions:**
  - Sur les dates : `YEAR(), NOW()` …
  - Sur les chaînes : `CONCAT(), CONTAINS()` …
  - Sur les nombres : `ABS(), FLOOR(), ROUND(), RAND()` …
  - Autres : `MD5(), URI()` …
- **Consultez https://www.w3.org/TR/sparql11-query/#SparqlOps pour plus de détails sur les fonctions.**

SPARQL - Trier et limiter les résultats

Très similaire à SQL

```
SELECT ?name
WHERE { ?x foaf:name ?name . }
ORDER BY ?name
LIMIT 20
OFFSET 20
```

- On distingue:

```
ORDER BY DESC (?name)
ORDER BY ASC  (?name)
```

SPARQL - Groupes et Agrégats (1)

Table Sales

Company	Amount	Year
ACME	$1250	2010
PRIME	$3000	2009
ABC	$2500	2009
ABC	$2800	2010
PRIME	$1950	2010
ACME	$2500	2009
ACME	$3100	2010
ABC	$1500	2009
ACME	$1250	2009
PRIME	$2350	2009
PRIME	$1850	2010

Base RDF où chaque ligne peut être remplacée par 4 triplets RDF:

```
:row1 a :Sale .
:row1 :company :ACME .
:row1 :amount 1250 .
:row1 :year 2010 .
```

```
SELECT (COUNT (?company) AS ?total)
WHERE  {
  ?x :company ?company.
}
```

?total
11

```
SELECT (COUNT (DISTINCT ?company) AS ?total)
WHERE  {
  ?x :company ?company.
}
```

?total
3

SPARQL - Groupes et Agrégats (2)

Table Sales

Company	Amount	Year
ACME	$1250	2010
PRIME	$3000	2009
ABC	$2500	2009
ABC	$2800	2010
PRIME	$1950	2010
ACME	$2500	2009
ACME	$3100	2010
ABC	$1500	2009
ACME	$1250	2009
PRIME	$2350	2009
PRIME	$1850	2010

Base RDF où chaque ligne peut être remplacée par 4 triplets RDF:

```
:row1 a :Sale .
:row1 :company :ACME .
:row1 :amount 1250 .
:row1 :year 2010 .
```

```
SELECT (SUM (?val) AS ?total)
WHERE  {
  ?s a :Sale; :amount ?val .
}
```

?total
24050.00

```
SELECT ?year (SUM (?val) AS ?total)
WHERE {
   ?s a :Sale;
      :amount ?val ;
      :year   ?year . }
GROUP BY ?year
```

?year	?total
2009	13100.00
2010	10950.00

SPARQL - Groupes et Agrégats (3)

Table Sales

Company	Amount	Year
ACME	$1250	2010
PRIME	$3000	2009
ABC	$2500	2009
ABC	$2800	2010
PRIME	$1950	2010
ACME	$2500	2009
ACME	$3100	2010
ABC	$1500	2009
ACME	$1250	2009
PRIME	$2350	2009
PRIME	$1850	2010

Base RDF où chaque ligne peut être remplacée par 4 triplets RDF:

```
:row1 a :Sale .
:row1 :company :ACME .
:row1 :amount 1250 .
:row1 :year 2010 .
```

```
SELECT ?year ?company (SUM (?val) AS ?total)
WHERE {
       ?s a :Sale ;
          :amount ?val ;
          :year ?year ;
          :company ?company .
}
GROUP BY ?year ?company
```

?year	?company	?total
2009	ACME	3750.00
2009	ABC	4000.00
2009	PRIME	5350.00
2010	ACME	4350.00
2010	PRIME	3800.00
2010	ABC	2800.00

SPARQL - Groupes et Agrégats (4)

Table Sales

Company	Amount	Year
ACME	$1250	2010
PRIME	$3000	2009
ABC	$2500	2009
ABC	$2800	2010
PRIME	$1950	2010
ACME	$2500	2009
ACME	$3100	2010
ABC	$1500	2009
ACME	$1250	2009
PRIME	$2350	2009
PRIME	$1850	2010

Base RDF où chaque ligne peut être remplacée par 4 triplets RDF:

```
:row1 a :Sale .
:row1 :company :ACME .
:row1 :amount 1250 .
:row1 :year 2010 .
```

```
SELECT ?year ?company (SUM (?val) AS ?total)
WHERE {
       ?s a :Sale ;
          :amount ?val ;
          :year ?year ;
          :company ?company .
}
GROUP BY ?year ?company
HAVING (?total > 5000)
```

?year	?company	?total
2009	PRIME	5350.00

HAVING n'est pas FILTER

- **FILTER** concerne les variables liées à un graph-pattern et apparaît dans le **WHERE**
- **HAVING** concerne les variables définies par des **agrégations** dans la clause **SELECT** et apparaît en dehors d'un graph-pattern

SPARQL – Expressions

- **Il est possible de calculer le résultat d'une expression dans une requête.**
- *Exemple1: Calculer le prix total de chaque vente.*

```
SELECT ?sale, (?nb_product*?price as ?total)
WHERE {
    ?sale     ex:product    ?product;
              ex:nbProduct  ?nb_product.
   ?product  ex:price      ?price.
}
```

SPARQL – Sous-requête

- **Il est possible d'emboîter une sous-requête dans une requête.**
- *Exemple1: Quelle est le produit le moins cher dans la base et quelles sont ses propriétés?*

```
SELECT ?product, ?property, ?value WHERE {
    SELECT (MIN(?price) as ?min) WHERE {
        ?item ex:price ?price
    }
    ?product ex:price ?min; ?property ?value .
}
```

SPARQL - Sélectionner le graphe source

- On peut interroger un ou plusieurs (i.e. union des base) graphe(s) de données RDF

```
SELECT *
FROM <http://ns.inria.fr/fabien.gandon/foaf.rdf>
WHERE {?x ?y ?z .}
```

- On peut spécifier dans **FROM** un **graphe nommé**. Dans ce cas on peut **spécifier/contraindre** dans la requête le graphe concerné.

```
SELECT *
FROM NAMED <http://ns.inria.fr/fabien.gandon/foaf.rdf>
WHERE { GRAPH ?g {?x ?y ?z .}}
```

- On peut combiner plusieurs **FROM** et **FROM NAMED**.
- Si **FROM(NAMED)** n'est pas spécifié, le moteur utilise en général une base par défaut.
- Pour plus de détails, veuillez consulter la recommandation:
  https://www.w3.org/TR/sparql11-query/#specifyingDataset

INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES LYON

4. Autres types de requêtes

SPARQL - ASK WHERE

- **ASK WHERE permet de vérifier l'existence d'au moins une réponse à la requête sans pour autant énumérer les résultats. Elle retourne True ou False en résultat.**
- **Exemples :**
  - ```
    ASK WHERE {
        dbr:Edsger_W._Dijkstra dbo:deathDate ?z .
    }
    ```
  - ```
    ASK WHERE {
        ?actor ^dbo:starring dbr:Kill_Bill .
        FILTER NOT EXISTS {?actor dbo:deathDate [].}
    }
    ```

SPARQL - CONSTRUCT

- **CONSTRUCT permet de construire un Graphe RDF à partir d'un résultat de recherche :**
  - Définir un « **template** » pour construire le graphe résultat
  - **Construire** un graphe RDF en guise de réponse
- **Exemple:**

```
CONSTRUCT { ?etudiant a :AvenirDeLaNation . }
WHERE     { ?etudiant a insa:etudiant .     }
```

SPARQL - DESCRIBE

- **Pour en savoir plus au sujet d'une ressource dont on n' peu d'information on utilise DESCRIBE. Le résultat est un graphe RDF sous la syntaxe concrète demandée (ex: Turtle).**
- **Le résultat de la requête dépend du moteur SPARQL.**
- *Exemple 1: tout savoir sur la ressource dbr:Uma_Thruman*

```
DESCRIBE dbr:Uma_Thruman
```

- *Exemple 2: tout savoir sur les ressources dont le label est "Mia Wallace"@en*

```
DESCRIBE * WHERE {
    ?x rdfs:label "Mia Wallace"@en
}
```

SPARQL - CRUD

Langage de type CRUD (create, read, update, delete) introduit dans SPARQL 1.1 pour modifier les bases de données RDF:

- CLEAR
- LOAD
- CREATE
- DROP
- COPY
- MOVE
- INSERT
- DELETE
- …

Exemple

```
PREFIX foaf:<….>
DELETE { ?person foaf:givenName "Amen" }
INSERT { ?person foaf:givenName "Aimene" }
WHERE
{
  ?person a foaf:Person .
  ?person foaf:givenName "Amen" .
  ?person foaf:name "Belfodil" .
}
```

INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES LYON

5. Accès et protocoles

Requêter en SPARQL en pratique (1)

En local ou à distance

- Accéder, parser, interroger un fichier RDF/XML
- Contacter un point d'accès via HTTP
- Le retour est un ensemble de résultats (*bindings*) ou du RDF dans une syntaxe concrète négociée (en retour de `CONSTRUCT` ou `DESCRIBE` par exemple)

Le protocole SPARQL est détaillée dans la recommandation

https://www.w3.org/TR/sparql11-protocol/

Requêter en SPARQL en pratique (2)

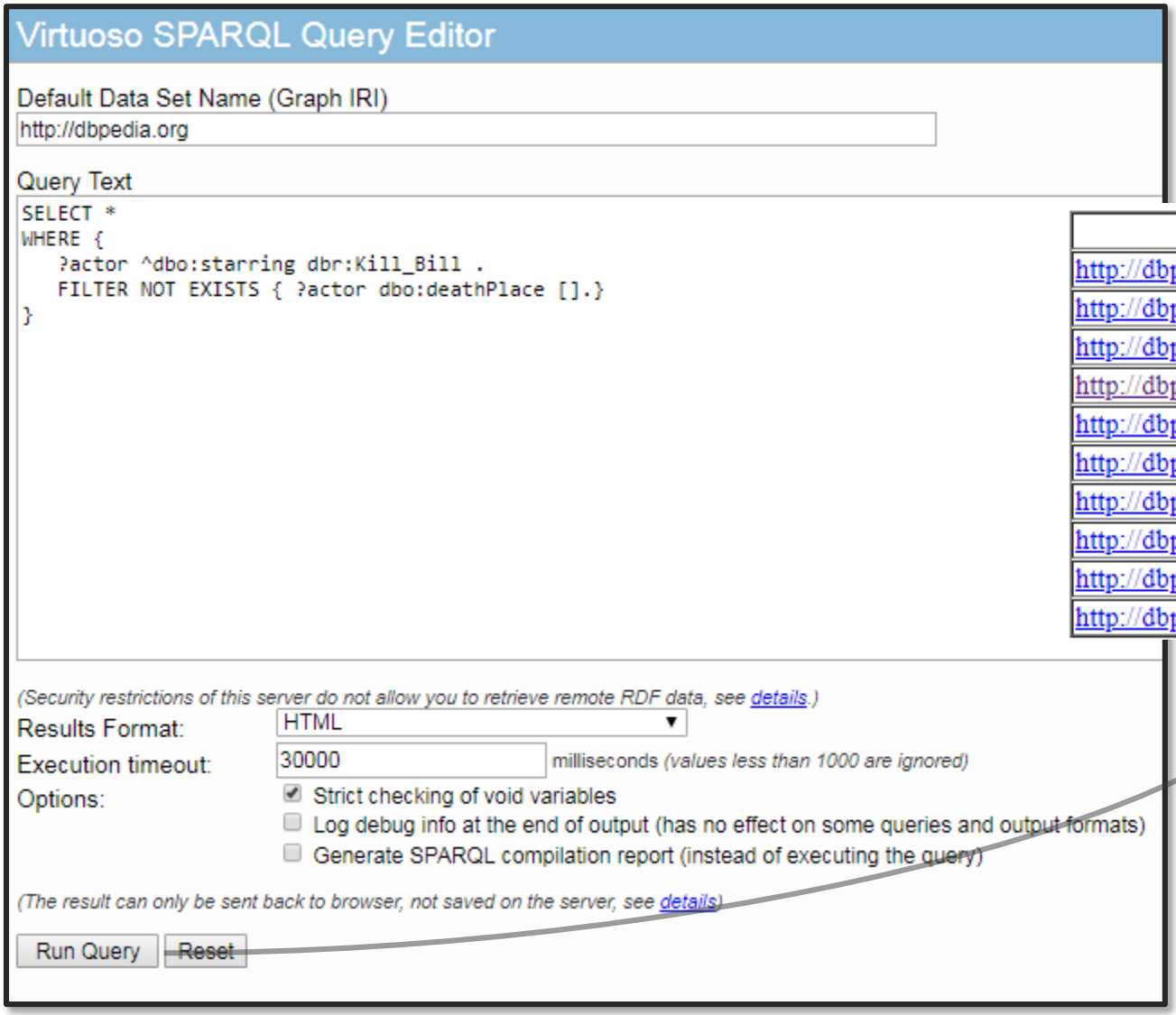
```
PREFIX dc: <http://purl.org/dc/elements/1.1/>
SELECT ?book ?who
WHERE { ?book dc:creator ?who }
```

```
GET /sparql/?query=EncodedQuery HTTP/1.1
Host: www.example
User-agent: my-sparql-client/0.1
```

```
HTTP/1.1 200 OK
Date: Fri, 06 May 2005 20:55:12 GMT
Server: Apache/1.3.29 (Unix) PHP/4.3.4 DAV/1.0.3
Connection: close
Content-Type: application/sparql-results+xml

<?xml version="1.0"?>
<sparql xmlns="http://www.w3.org/2005/sparql-results#">
 <head>
   <variable name="book"/>
   <variable name="who"/>
 </head>
 <results distinct="false" ordered="false">
   <result>
     <binding name="book"><uri>http://www.example/book/book5</uri></binding>
     <binding name="who"><bnode>r29392923r2922</bnode></binding>
   </result>  …
```

DBPEDIA SPARQL Endpoint (1)

https://dbpedia.org/sparql



actor
http://dbpedia.org/resource/Lucy_Liu
http://dbpedia.org/resource/Julie_Dreyfus
http://dbpedia.org/resource/Michael_Parks
http://dbpedia.org/resource/Uma_Thurman
http://dbpedia.org/resource/Daryl_Hannah
http://dbpedia.org/resource/Michael_Madsen
http://dbpedia.org/resource/Sonny_Chiba
http://dbpedia.org/resource/Chiaki_Kuriyama
http://dbpedia.org/resource/Gordon_Liu
http://dbpedia.org/resource/Vivica_A._Fox

Réponse

DBPEDIA SPARQL Endpoint (2)

* Communication HTTP

query.txt

```
SELECT ?actor
WHERE {
   ?actor ^dbo:starring dbr:Kill_Bill .
   FILTER NOT EXISTS {?actor dbo:deathDate [].}
}
```

Commande

- `curl -H "Accept: application/turtle" -g --data-urlencode query@query.txt http://dbpedia.org/sparql` → Retourne le résultat sous format **turtle**

- `curl -H "Accept: application/json" -g --data-urlencode query@query.txt http://dbpedia.org/sparql` → Utiliser la commande **jq** pour post-traiter les résultats

Quelques Outils

Apache Jena sous Java
https://jena.apache.org/

sparqlwrapper sous Python
https://rdflib.github.io/sparqlwrapper/

L'Outil graphique Twinkle
http://www.ldodds.com/projects/twinkle/

INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES LYON

6. Épilogue

A retenir

- **RDF :** un cadre de modélisation de données dans le **Web 3.0**
- **SPARQL :** un langage pour interroger les données du **Web 3.0**
- **Poser une question SPARQL (par exemple SELECT)** consiste à écrire un **graphe requête (un motif)** pour lequel on recherche des occurrences dans le graphe cible.
- **Le calcul des résultats d'une requête se fait par appariement de graphe.**
- **D'autres requêtes que SELECT existent** (ASK WHERE, CONSTRUCT, DESCRIBE, INSERT, DELETE, UPDATE, …).

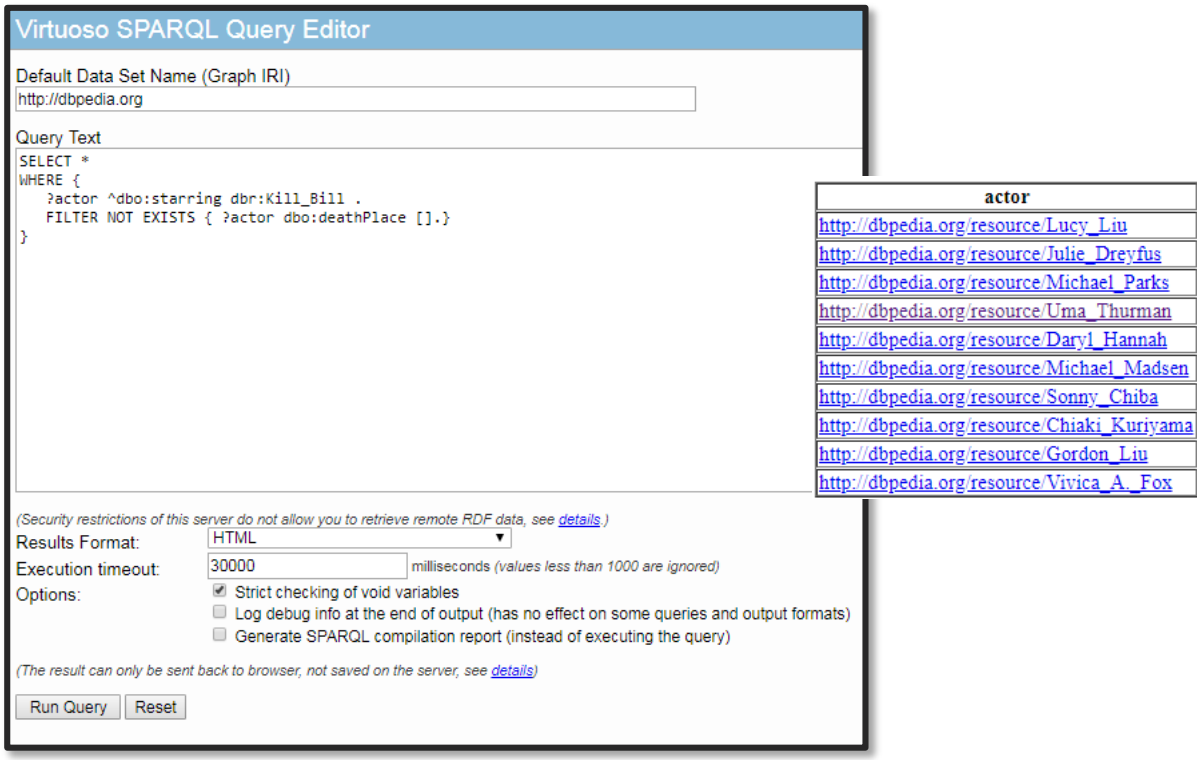
Avant le TD…

Découvrir, naviguer à travers le nuage

- Découvrez les vocabulaires:

  http://xmlns.com/foaf/spec/ , http://dbpedia.org/ontology/

- Jouez avec: http://dbpedia.org/sparql



Des requêtes pour comprendre les schémas (1)

Web sémantique : données distribuées, données en masse

- **Slogan AAA :** Il est ainsi impossible de tout savoir, et **encore moins de savoir comment et par quoi les choses sont décrites**.
- **Exemple:**
  - **Question:** Quand-est-ce que Hayao Miyazaki est né ?
  - **Problème:** je ne connais ni la ressource associé à Hayao Miyazaki, ni le vocabulaire adéquat pour 'la date de naissance' ?
  - **Solution:**
    - Pour les ressources dans DBPEDIA, utiliser Wikipedia: la ressource associée à https://en.wikipedia.org/wiki/Hayao_Miyazaki **est :**

      http://dbpedia.org/resource/Hayao_Miyazaki
    - Pour le vocabulaire, commencez par `DESCRIBE` ou regardez :

      http://dbpedia.org/page/Hayao_Miyazaki

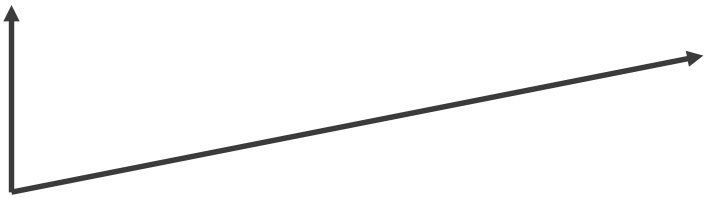
Des requêtes pour comprendre les schémas (2)

Ask:

```
SELECT DISTINCT ?class
WHERE {?class rdfs:subClassOf :Person}
```

Answer:

?class
:Actor
:Actress
:Man
:Woman
:Politician
:Producer



Ask:

```
SELECT DISTINCT ?property
WHERE {?q0 a :Actor .
       ?q0 ?property ?object .}
```

Answer:

?property
bornOn
diedOn
playedIn
rdf:type
rdfs:label
produced
sang
wrote

Requêter les données pour découvrir le vocabulaire.

Biblio-webographie

- Harald Sack. **Semantic Web technologies**. Open course Hasso Plattner Institut (2013) : https://open.hpi.de/course/semanticweb
- Gandon F., Faron-Zucker C., Corby O. **Le Web sémantique. Comment lier les données et les schémas sur le Web ?** Dunod eds. 2012.
- Allemang D., Hendler J. **Semantic Web for the working ontologist : Effective modelling in RDFS and OWL**. Morgan Kaufmann eds. 1st edition (2008)
- http://www.w3.org/People/Ivan/CorePresentations/RDFTutorial/
- Actes annuels des conférences **WWW**, **ISWC**, **ESWC** (voir DBLP)
- http://www.viseo.net/sites/default/files/viseo-group/rdc_cours_5.pdf
- MOOC INRIA Web sémantique par Gandon et al.

Les recommandations et RFC sont toujours d'excellentes sources !