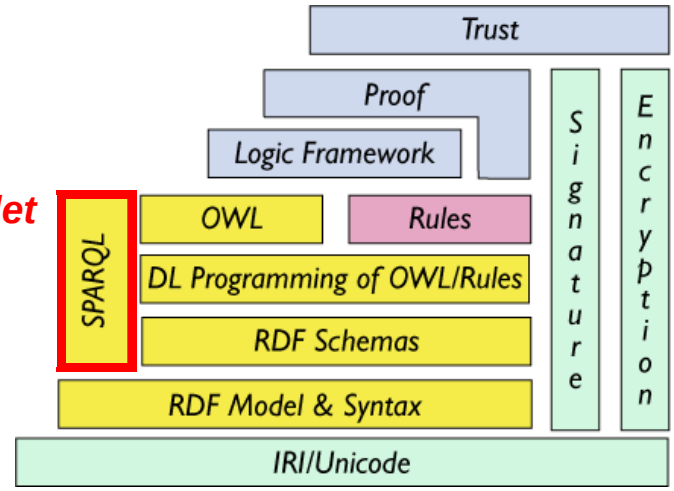
Requêtes sur RDF

Le langage de requêtes SPARQL
SPARQL Protocol And RDF Query Language

Tout est triplet



W3C, T Berners-Lee, Ivan Herman

Le gâteau du web sémantique…

- SPARQL Query Language for RDF
  W3C Working Draft 4 October 2006
  - Langage de requêtes

- SPARQL Protocol for RDF
  W3C Candidate Recommendation 6 April 2006
  - Description (WSDL 2.0) pour soumettre une requête à une serveur distant et récupérer la réponse (binding SOAP)

- SPARQL Query Results XML Format
  W3C Candidate Recommendation 6 April 2006
  - Langage de résultat

- Forme principale en 3 clauses (SQL):

```
PREFIX ugb: <http://www.ugb.sn/dess#>
SELECT ?etudiant ?nom
FROM http//www.ugb.sn/data.rdf
WHERE {
 ?etudiant ugb:inscrit ?x .
 ?x ugb:siteweb http//www.ugb.sn .
 ?etudiant ugb:nom ?nom .
 ?etudiant ugb:age ?age .
 FILTER ( ?age > 20 )
}
ORDER BY ?nom
LIMIT 20
OFFSET 20
```

```
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>  …
```


- Exemple : noms des personnes ayant un email

```
SELECT ?nom WHERE {
  ?x nom ?nom .
  ?x email ?email
}
```

- Résultat : tous les patterns solutions sur lesquels le pattern query peut être projeté (une variable peut avoir plusieurs bindings)
- Exemple :

```
_:a nom "Fabien"
_:b nom "Thomas"
_:c nom "Louis XIV"
_:d nom "Aline"
_:b email <mailto:thom@chaka.sn>
_:a email <mailto:Fabien.Gandon@inria.fr>
_:d email <mailto:avalandre@pacinco.com>
_:a email <mailto:zinzin@free.fr>
```

x2

- Noms et prénoms des auteurs:

```
SELECT ?nom ?prenom
WHERE {
 ?x nom ?nom .
 ?x prenom ?prenom .
 ?x auteur ?y .
}
```

- Pour utiliser des namespaces:

```
PREFIX fac: <http://www.u-picardie.fr#>
SELECT ?etudiant
WHERE {
 ?etudiant fac:inscrit ?x .
 ?x fac:siteweb http://www.u-picardie.fr .
}
```

- Namespace de base : BASE <>

- Le résultat de la requête précédente en XML

```
<?xml version="1.0"?>
<sparql xmlns="http://www.w3.org/2005/sparql-results#">
 <head>
  <variable name="etudiant"/>
 </head>
 <results ordered="false" distinct="false">
  <result>
   <binding name="etudiant">
<uri>http//www.ugb.sn/data.rdf#ndieng</uri></binding>
  </result>
  <result>
   <binding name="etudiant">
<uri>http//www.ugb.sn/data.rdf#ndouf</uri></binding>
  </result>
</sparql>
```

- Les triplets ayant une racine commune peuvent être simplifiés ainsi que la relation de typage:

```
SELECT ?nom ?prenom
WHERE {
 ?x a Person;
    nom ?nom ;
    prenom ?prenom ;
    auteur ?y . }
```

```
SELECT ?nom ?prenom
WHERE {
 ?x rdf:type Person .
 ?x nom ?nom .
 ?x prenom ?prenom .
 ?x auteur ?y .
}
```

- Une liste de valeurs

```
?x prenom "Fabien", "Lucien" .
```

- Ressource anonyme dans pattern requête

`[prenom "Fabien"]` ou `[] prenom "Fabien"`

- Question: `?x a Document; auteur [nom "Lo] .`
- Réponse: les documents ?x écrits par un auteur ayant pour nom "Lo"

- Sélectionner les sources utilisables :

```
PREFIX fac: <http://www.u-picardie.fr#>
SELECT ?etudiant
FROM http//www.u-picardie.fr/data.rdf
WHERE {
 ?etudiant fac:inscrit ?x .
 ?x fac:siteweb http//www.u-picardie.fr .
}
```

- Parties optionnelles :

```
PREFIX fac: <http://www.u-picardie.fr#>
SELECT ?etudiant ?nom
WHERE {
 ?etudiant fac:inscrit ?x .
 ?x fac:siteweb http//www.u-picardie.fr .
 OPTIONAL {?etudiant fac:nom ?nom . }
 }
```

- Donner des patterns alternatifs :

```
PREFIX fac: <http://www.u-picardie.fr#>
SELECT ?etudiant
WHERE {
 ?etudiant fac:inscrit ?x .
 {
  {
   ?x fac:siteweb http//www.u-picardie.fr .
  }
  UNION
  {
   ?x fac:siteweb http//www.univ-
lille1.fr .
  }
 }
}
```



```
PREFIX iut: <http://www.iutnice.fr#>
SELECT ?etudiant ?nom
WHERE {
 ?etudiant iut:inscrit ?x .
 ?x iut:siteweb http://www.iutnice.fr .
 ?etudiant iut:nom ?nom .
 ?etudiant iut:age ?age .
 FILTER ?age > 22
}
ORDER BY ?nom
LIMIT 20
OFFSET 20
```

- Étudiants de plus de 22 ans triés par nom, les réponses de #21 à #40

- Dans la clause FILTER:
  - Comparateurs: <, >, =, <=, >=, !=
  - Tests sur les binding des variables:
    isURI(?x), isBlank(?x), isLiteral(?x), bound(?x)
  - Filtres à base d'expressions régulières regex(?x, "A.*")
  - Accès aux attributs/valeur lang(), datatype(), str()
  - Fonctions de (re-)typage (casting) xsd:integer(?x)
  - Fonctions externes / extensions
  - Combinaisons &&, ||
- Dans la clause WHERE: @fr , ^^xsd:integer
- Dans la clause SELECT: distinct

- Tester si un pattern est introuvable :

```
PREFIX fac: <http://www.u-picardie.fr#>
SELECT ?etudiant
WHERE {
 ?etudiant rdf:type fac:Etudiant .
 OPTIONAL
  {
    ?etudiant fac:auteur ?x .
    ?x rdf:type fac:Programme .
    ?x fac:langage fac:Java .
  }
 FILTER ! bound(?x)
}
```

- Vérifier qu'il existe au moins une réponse :

```
PREFIX iut: <http://www.u-picardie.fr#>
ASK {
 ?etudiant fac:inscrit ?x .
 ?x fac:siteweb www.u-picardie.fr .
 ?etudiant fac:age ?age .
 FILTER ?age > 30
}
```

- Existe-t-il un étudiant de plus de 30 ans?
- Résultat booléen :

```
<sparql xmlns="http://www.w3.org/2005/sparql-results#">
 <head> … </head>
 <boolean> … </boolean>
</sparql>
```


Services SPARQL

• Plusieurs moteurs SPARQL existent (JENA, KAON, Virtuoso, …)

• Des sites proposent des services SPARQL, éventuellement avec une interface

•Liste de services SPARQL : *http://esw.w3.org/topic/SparqlEndpoints*

Exercice

• Le site *http://dbpedia.org/* permet de fouiller Wikipedia en utilisant RDF et SPARQL. Un point d'accès SPARQL avec formulaire est disponible à l'adresse *http://dbpedia.org/sparql/* ou *http://dbpedia.org/snorql/*

Ontologie : *http://wiki.dbpedia.org/Ontology* et *http://mappings.dbpedia.org/server/ontology/classes*

Question 1 : Retrouver tous les livres (type *http://dbpedia.org/ontology/Book*) dont l'auteur (propriété *author*) est *Jules_Verne* (ressource décrite dans *http://dbpedia.org/page/Jules_Verne*)

- Question 2 : Retrouver toutes les informations liées au livre Voyage_au_centre de la terre

- Question 3 : Retrouver tous les écrivains nés à Nantes et éventuellement ceux qu'ils ont influencés.