RDF

Dan VODISLAV

Université de Cergy-Pontoise

Master Informatique M2

Plan

- RDF
  - Modèle de données
  - Graphe RDF
  - RDF/XML
- RDFS : schémas pour RDF
- OWL: schémas avancés
- SPARQL: interrogation de données RDF

RDF: Ressource Description Framework

- Langage de base du web sémantique
  - Description de ressources web: pages web, images, vidéos, …
  - Décrit les propriétés des ressources ou les relations entre ressources
  - Plusieurs syntaxes possibles (y compris XML)
  - RDF Schema (RDFS): concepts, classes, schémas → ontologies
- Niveaux du modèle RDF
  - Niveau physique : triplets / déclarations
    - Types de base : ressources, propriétés, déclarations
    - Types complexes: collections, listes
  - Schémas (RDFS): classes, types de propriétés
    - OWL: éléments plus avancés

Triplets RDF

- Déclaration : triplet (S, P, V)
  - « Atome » de connaissance
  - Signification: le sujet **S** a pour la propriété **P** la valeur **V**
    - On note parfois les triplets (**Sujet**, **Prédicat**, **Objet**)
- Exemple
  - `(PageETIS, auteur, Michel)`
  - `(ETIS, pageWeb, PageETIS)`
  - `(Michel, pageWeb, PageMichel)`
  - `(ETIS, directeur, Inbar)`
  - `(Michel, nom, "Michel Jordan")`
- Comparaison avec le modèle relationnel : `(ETIS, directeur, Inbar)`

Laboratoire

identifiant	directeur	pageWeb	…
…	…	…	…
ETIS	Inbar	PageETIS	…
…	…	…	

Personne

identifiant	nom	…
Michel	Michel Jordan	…
Inbar	Inbar Fijalkow	…
…		

Ressources et URI

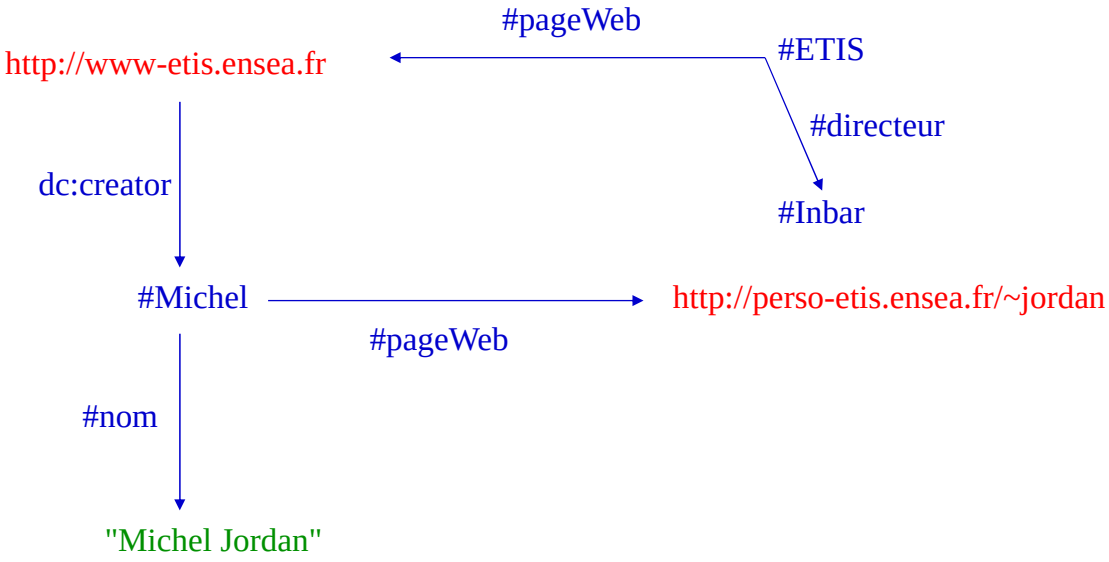
- Les ressources et les propriétés sont identifiées par des URI
  - **S**, **P** et **V** sont donnés par des URI
  - **V** peut être aussi une valeur littérale
- Remarque: URI ≠ URL, URI pas forcément une adresse réelle sur le web
- Exemple (diverses notations possibles)
  - `(http://www-etis.ensea.fr, dc:creator, #Michel)`
  - `(#ETIS, #pageWeb, http://www-etis.ensea.fr)`
  - `(#Michel, #pageWeb, http://perso-etis.ensea.fr/~jordan)`
  - `(#ETIS, #directeur, #Inbar)`
  - `(#Michel, #nom, "Michel Jordan")`
- URI locales : `#Michel, #ETIS, #pageWeb, #directeur, #Inbar, #nom`
- URI externes : `http://www-etis.ensea.fr, dc:creator, http://perso-etis.ensea.fr/~jordan`
- Valeurs littérales : `"Michel Jordan"`
  - On peur spécifier un type, ex. `"32"^^xsd:integer`
  - On peur spécifier une langue, ex. `"Eiffel Tower"@en`

Utilisation des espaces de noms

- Pour les ressources locales: espace de noms propre
  - Regroupe et identifie les noms des ressources locales (`#ETIS, #PageWeb, …`)
  - Ex. `xmlns:moi="http://monappli.monorg.com"`
  - `#ETIS` signifiera `http://monappli.monorg.com#ETIS`
  - Notations alternatives:
    `moi:ETIS` ou `http://monappli.monorg.com/ETIS`
- Pour ressources externes: référence aux espaces de noms spécifiques
  - Objectif: utiliser des ressources/propriétés « standard »
  - Ex. Dublin Core: standardisation des concepts concernant les documents
    `xmlns:dc="http://purl.org/dc/elements/1.1"`
    `dc:creator` = le créateur d'un document/ressource
- Pour les types de données: espace de noms XML Schema
  - `xmlns:xsd="http://www.w3.org/2001/XMLSchema"`

Graphe RDF

- Triplet = deux nœuds (S, V) + l'arc orienté (P) qui les relie
- Ensemble de triplets → graphe orienté



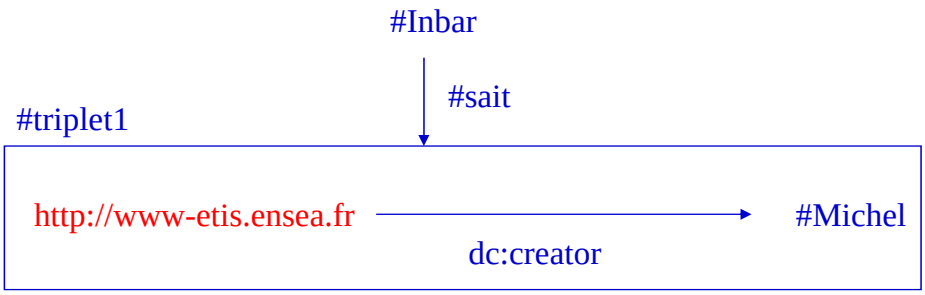
Éléments prédéfinis

- Espaces de noms `rdf` ou `rdfs`
  - `xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"`
  - `xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"`
- Pour les types
  - Propriété `rdf:type`
  - Types de base: `rdf:Resource`, `rdf:Property`, `rdf:Statement`
- Pour une déclaration (triplet)
  - `rdf:subject`, `rdf:predicate`, `rdf:object` désignent les trois composantes
- D'autres exemples plus loin

Réification

- Une déclaration peut devenir une ressource
- Exemple

```
(#triplet1, rdf:subject, http://www-etis.ensea.fr)
(#triplet1, rdf:predicate, dc:creator)
(#triplet1, rdf:object, #Michel)
(#Inbar, #sait, #triplet1)
```




```
(#triplet1, rdf:type, rdf:Statement)
(#triplet1, rdf:type, rdf:Resource)
```


Types complexes

- *Container*: ressource de la classe *rdfs:Container*
  - Sous-classes: *rdf:Bag*(multi-ensemble), *rdf:Seq*(séquence), *rdf:Alt*(alternative)
  - Appartenance au Container: propriétés *rdf:_1*, *rdf:_2*, …
- *Liste*: ressource de type *rdf:List*
  - Constructeurs : *rdf:first*, *rdf:rest*, *rdf:nil*
- Exemple

```
(#doctorants, rdf:type, rdf:Bag)
(#doctorants, rdf:_1, #Mehdi)
(#doctorants, rdf:_2, #Maria)

(#membresETIS, rdf:type, rdf:Bag)
(#membresETIS, rdf:_1, #Inbar
(#membresETIS, rdf:_2, #Michel)
(#membresETIS, rdf:_3, #doctorants)
```

Ressources anonymes (blank nodes)

- Ressource non identifiée par une URI
  - Utilisée quand on n'a pas besoin d'identifier une ressource
  - Notation: `_:x`
  - *x* est un identifiant local, pas une URI locale
- Exemples
  - Condition d'existence : il existe une page web créée par Michel

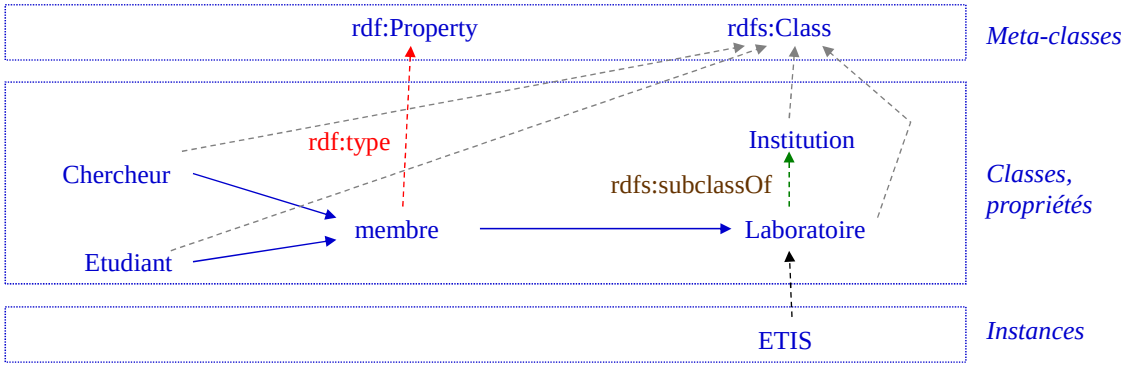
```
(_:page, rdf:type, #PageWeb)
(_:page, dc:creator, #Michel)
```

  - Construire des valeurs structurées: une adresse

```
(#Jean, #habite, _:adr)
(_:adr, #ville, "Paris")
(_:adr, #rue, "rue Saint Dominique")
(_:adr, #num, "10"^^xsd:integer)
```


RDF Schema

- Description de *classes* et de *types de propriétés*
  - Classes: *rdfs:Class, rdfs:subclassOf*
  - Propriétés: *rdfs:subpropertyOf , rdfs:domain, rdfs:range*




```
(#Institution, rdf:type, rdfs:Class)
(#Laboratoire, rdf:type, rdfs:Class)
(#Laboratoire, rdfs:subclassOf, #Institution)
(#membre, rdf:type, rdf:Property)
(#membre, rdfs:domain, #Etudiant)
(#membre, rdfs:domain, #Chercheur)
(#membre, rdfs:range, #Institution)
(#ETIS, rdf:type, #Laboratoire)
```

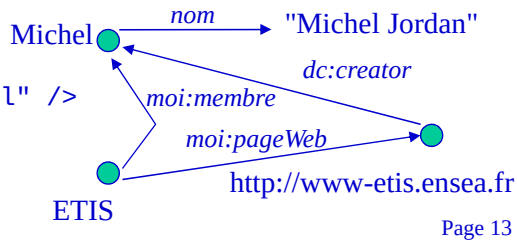
RDF/XML

- Syntaxe la plus utilisée pour RDF/RDFS (d'autres existent)
  - Document RDF/RDFS : éléments de type Description
  - Avantages: outils XML, utilisation des espaces de noms
  - Problème : RDF=graphe, XML=arbre → plusieurs représentations possibles

```
<rdf:RDF xmlns:rdf="http://wwww.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:dc="http://purl.org/dc/elements/1.1/"
         xmlns:moi="http://monappli.monorg.com"
         xml:base="http://monappli.monorg.com">
    <rdf:Description rdf:ID="Michel"/>
        <nom>Michel Jordan</nom>
    </rdf:Description>
    <rdf:Description rdf:ID="ETIS">
        <moi:pageWeb>
            <rdf:Description rdf:about='http://www-etis.ensea.fr'>
                <dc:creator>
                    <rdf:Description rdf:about="#Michel"/>
                </dc:creator>
            </rdf:Description>
        </moi:pageWeb>
        <moi:membre>
            <rdf:Description rdf:about="#Michel" />
        </moi:membre>
    </rdf:Description>
</rdf:RDF>
```

Ressource locale (rdf:ID)

Ressource globale (rdf:about)



Abréviations

- `rdf:ID` et `rdf:about`
  - Dans l'exemple précédent (`rdf:ID="Michel"`) ≡ (`rdf:about="moi:Michel"`) à cause de `xml:base` et `xmlns:moi` qui ont la même valeur
- Propriétés ayant pour valeur une ressource

```
<rdf:Description rdf:about='http://www-etis.ensea.fr'>
    <dc:creator>
        <rdf:Description rdf:about="#Michel"/>
    </dc:creator>
</rdf:Description>
```

peut s'écrire

```
<rdf:Description rdf:about='http://www-etis.ensea.fr'>
    <dc:creator resource="#Michel"/>
</rdf:Description>
```

- Utilisation des noms de classe

```
<rdf:Description rdf:about='moi:ETIS'>
    <rdf:type resource="moi:Laboratoire"/>
    <moi:membre resource="#Michel"/>
</rdf:Description>
```

peut s'écrire

```
<moi:Laboratoire rdf:about='moi:ETIS'>
    <moi:membre resource="#Michel"/>
</moi:Laboratoire>
```

Collections et blank nodes en RDF/XML

- Valeurs structurées avec blank node

```
<rdf:Description rdf:ID="Jean"/>
  <habite>
    <rdf:Description>
      <ville>Paris</ville>
      <rue>rue Saint Dominique</rue>
      <num rdf:datatype="xsd:integer">
        10</num>
    </rdf:Description>
  </habite>
</rdf:Description>
```

ou

```
<rdf:Description rdf:ID="Jean"/>
  <habite rdf:parseType="Resource">
    <ville>Paris</ville>
    <rue>rue Saint Dominique</rue>
    <num rdf:datatype="xsd:integer">
       10</num>
  </habite>
</rdf:Description>
```

- Collection

```
<rdf:Description rdf:ID='ETIS'>
  <membres>
    <rdf:Bag>
      <rdf:li rdf:resource="Inbar"/>
      <rdf:li rdf:resource="Michel"/>
      <rdf:li rdf:resource="Dan"/>
    </rdf:Bag>
  </membres>
</rdf:Description>
```


OWL

- OWL (Web Ontology Language) = extension de RDFS
  - Contraintes plus puissantes
  - Possibilités de raisonnement
- RDF/RDFS
  - Seules contraintes: *rdfs:subClassOf* et *rdfs:subPropertyOf*
  - Définition de classes: par référence (URI) + déclarations instances
    - Hypothèse du monde ouvert: une info manquante n'est pas forcément fausse

    → L'ensemble des instances d'une classe n'est pas connu
  - Peu de possibilités de raisonnement

Définition de classes OWL

- Plusieurs façons de définir une classe
  - par une référence (URI)
  - par l'énumération de ses instances
  - par ses propriétés
  - comme union, intersection, complément d'autres classes
- Exemple énumération

```
<owl:Class rdf:ID="mycontinents">
    <owl:oneOf rdf:parseType="Collection">
        <owl:Thing rdf:about="#Eurasia"/>
        <owl:Thing rdf:about="#Africa"/>
        <owl:Thing rdf:about="#NorthAmerica"/>
        <owl:Thing rdf:about="#SouthAmerica"/>
        <owl:Thing rdf:about="#Australia"/>
        <owl:Thing rdf:about="#Antarctica"/>
    </owl:oneOf>
</owl:Class>
```


Définition de classes OWL (suite)

- Par les propriétés
  - Valeur des propriétés : *owl:allValuesFrom, owl:someValuesFrom, owl:hasValue*
  - Cardinalité : *owl:maxCardinality, owl:minCardinality, owl:Cardinality*

Exemple: classes qui ont une propriété *membre* de type *Student*

```
  <owl:Restriction>
     <owl:onProperty rdf:resource="#membre" />
     <owl:allValuesFrom rdf:resource="#Student" />
  </owl:Restriction>
```

- Par calcul : *owl:intersectionOf, owl:unionOf, owl:complementOf*

```
<owl:Class>
    <owl:intersectionOf rdf:parseType="Collection">
        <owl:Class rdf:about="mycontinents">
        <owl:Class>
            <owl:oneOf rdf:parseType="Collection">
                <owl:Thing rdf:about="#Europe" />
                <owl:Thing rdf:about="#Asia" />
                <owl:Thing rdf:about="#America" />
            </owl:oneOf>
        </owl:Class>
    </owl:intersectionOf>
</owl:Class>
```

Relations entre classes en OWL

- *rdfs:subClassOf*
  - l'extension d'une classe est incluse dans l'extension de l'autre
- *owl:equivalentClass*
  - classes avec la même extension, mais qui ne désignent pas le même concept

  <footballTeam owl:equivalentClass us:soccerTeam />
- *owl:disjointWith*
  - deux classes disjointes

Définition de propriétés OWL

- RDF Schema : *rdfs:subPropertyOf*, *rdfs:domain* et *rdfs:range*
- Relations entre propriétés
  - *owl:equivalentProperty* : les deux propriétés ont la même extension, mais ne sont pas identiques
  - *owl:inverseOf* : une propriété est l'inverse de l'autre

```
<owl:ObjectProperty rdf:ID="enfant">
    <owl:inverseOf rdf:resource="#parent"/>
</owl:ObjectProperty>
```

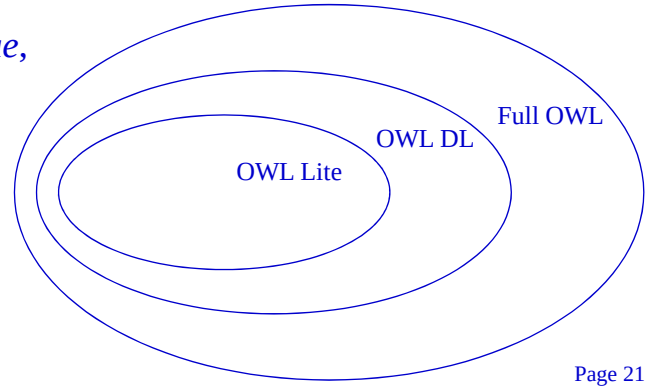
- Contraintes de cardinalité
  - Propriétés mono-valuées :
    `<owl:FunctionalProperty rdf:about="#époux" />`
  - Propriétés mono-valuées inverse :

```
<owl:InverseFunctionalProperty rdf:ID="mèreBiologique">
    <rdfs:domain rdf:resource="#femme"/>
    <rdfs:range rdf:resource="#personne"/>
</owl:InverseFunctionalProperty>
```

- Contraintes logiques
  - *owl:SymmetricProperty* (époux)
  - *owl:TransitiveProperty* (ancêtre)

Hiérarchies de langages OWL

- Full OWL: RDF/RDFS + nouveaux opérateurs OWL
  - Puissant, mais raisonnement non décidable
- OWL DL (Description Logic)
  - Restrictions sur Full OWL qui assurent un raisonnement décidable
  - Ex. une classe ou une propriété ne peut pas être une instance
- OWL Lite
  - Restrictions sur OWL DL qui assurent un raisonnement efficace
  - Ex. interdiction de *owl:unionOf*, *owl:complementOf*, *owl:hasValue*, *owl:disjointWith*, …



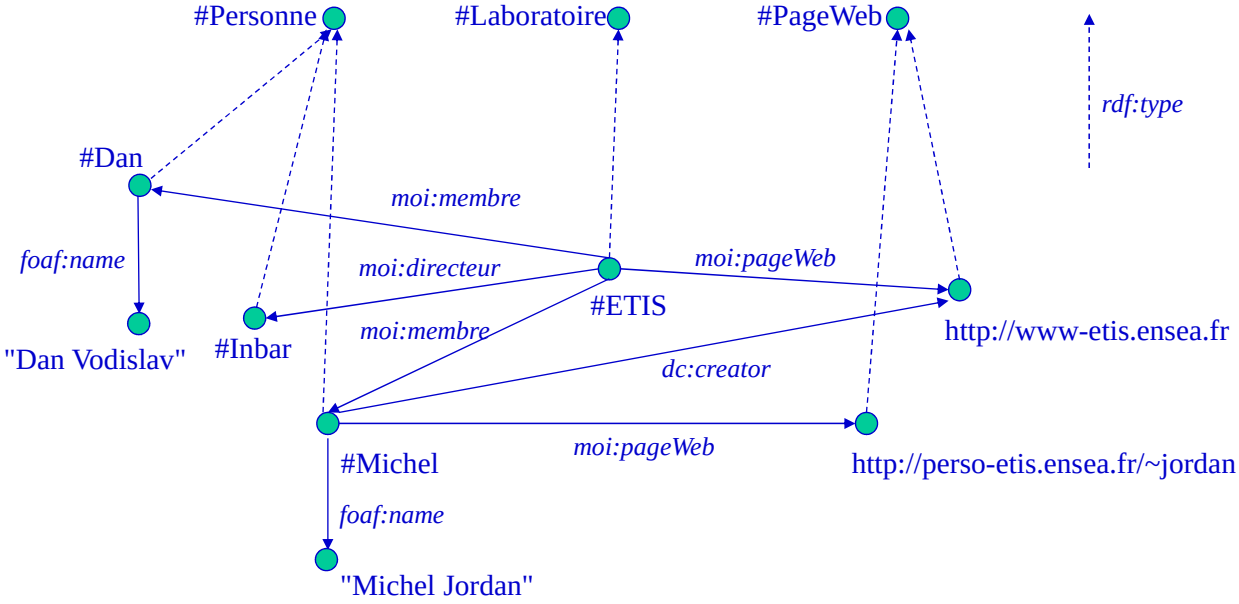
SPARQL

- Langage d'interrogation pour données RDF
  - Recommandation W3C 2008
- Forme la plus courante des requêtes

```
SELECT [DISTINCT] ?var1 ?var2 … ?varm
WHERE { pattern1 .
        pattern2 .
        …
        patternn }
```

  - Les patterns sont des triplets en format TURTLE (pas RDF/XML)
  - Les variables apparaissent dans les patterns
  - Requêtes conjonctives
  - Résultat: table de valeurs (bindings) correspondant à (?var_1, …, ?var_m)

Exemple



Requêtes SELECT

- Un ou plusieurs patterns
  - Tout élément d'un pattern (triplet) peut être une variable

  Ex. *Les membres du laboratoire ETIS*

```
SELECT ?x
WHERE {
  <http://monappli.monorg.com/ETIS> <http://monappli.monorg.com/membre> ?x .
}
```

ou

```
PREFIX moi: <http://monappli.monorg.com/>
PREFIX : <http://monappli.monorg.com/>
SELECT ?x
WHERE {
  :ETIS moi:membre ?x .
}
```

Résultat

x
<http://monappli.monorg.com/Michel>
<http://monappli.monorg.com/Dan>

Requêtes SELECT (suite)

Ex. *Qui a des pages web et quelles sont ces pages*

```
PREFIX moi: <http://monappli.monorg.com/>
PREFIX : <http://monappli.monorg.com/>
SELECT ?x ?y
WHERE {
  ?x moi:pageWeb ?y .
}
```

x	y
<http://monappli.monorg.com/ETIS>	<http://www-etis.ensea.fr>
<http://monappli.monorg.com/Michel>	<http://perso-etis.ensea.fr/~jordan>

Ex. *Qui a créé la page web du laboratoire ETIS*

```
PREFIX moi: <http://monappli.monorg.com/>
PREFIX dc: <http://purl.org/dc/elements/1.1/>
PREFIX : <http://monappli.monorg.com/>
SELECT ?x
WHERE {
  ?x dc:creator ?y .
  :ETIS moi:pageWeb ?y .
}
```

x
<http://monappli.monorg.com/Michel>

Regroupement par sujet

Ex. *Le nom et la page web de Michel*

```
PREFIX moi: <http://monappli.monorg.com/>
PREFIX : <http://monappli.monorg.com/>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?nom ?page
WHERE {
  :Michel foaf:name ?nom ;
          moi:pageWeb ?page .
}
```

nom	page
"Michel Jordan"	<http://perso-etis.ensea.fr/~jordan>

Patterns optionnels

Ex. *Le nom et la page web des membres d'ETIS*

```
PREFIX moi: <http://monappli.monorg.com/>
PREFIX : <http://monappli.monorg.com/>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?nom ?page
WHERE {
  :ETIS moi:membre ?x .
  ?x foaf:name ?nom .
  OPTIONAL{
    ?x moi:pageWeb ?page .
  }
}
```

nom	page
"Michel Jordan"	<http://perso-etis.ensea.fr/~jordan>
"Dan Vodislav"	

Union

Ex. *Les membres d'ETIS, y compris son directeur*

```
PREFIX moi: <http://monappli.monorg.com/>
PREFIX : <http://monappli.monorg.com/>
SELECT ?x
WHERE {
  {
    :ETIS moi:membre ?x .
  }
  UNION
  {
    :ETIS moi:directeur ?x .
  }
}
```

x
<http://monappli.monorg.com/Michel>
<http://monappli.monorg.com/Dan>
<http://monappli.monorg.com/Inbar>

Tri, limite, offset

Ex. *La seconde personne en ordre alphabétique du nom*

```
PREFIX : <http://monappli.monorg.com/>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?x
WHERE {
  ?x a :Personne ;
     foaf:name ?y .
}
ORDER BY ASC(?y)
LIMIT 1
OFFSET 1
```

- Remarques
  - (?x a *Type)* est une abréviation pour (?x rdf:type *Type*)
  - ORDER BY peut utiliser ASC ou DESC (par défaut ASC)
  - LIMIT *n* réduit le nombre de résultats retournés à *n*
  - OFFSET *m* saute les *m* premiers résultats

Filtrage

Ex. *Les personnes dont le nom commence par un « M »*

```
PREFIX moi: <http://monappli.monorg.com/>
PREFIX : <http://monappli.monorg.com/>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?x
WHERE {
  ?x a :Personne ;
     foaf:name ?y .
  FILTER (regex(?y, "^m", "i"))
}
```

- FILTER : condition booléenne sur les valeurs des variables
  - Opérateurs arithmétiques pour les numériques
  - Tests : *isURI*, *isBlank*, *isLitteral* , *bound*
  - Opérateurs de comparaison
  - Opérateurs logiques pour combiner les conditions: &&, | |, !
  - Opérateur ***regex***(*texte*, *pattern* [, *option*])

Autres types de requêtes

- ASK: test d'existence

  Ex. *Existe-t-il une page web du laboratoire ETIS?*

```
PREFIX moi: <http://monappli.monorg.com/>
PREFIX : <http://monappli.monorg.com/>
ASK {
  :ETIS moi:pageWeb ?x .
}
```

- DESCRIBE: retour d'une description des ressources
  - Description non standard retournée par le service SPARQL
  - Généralement: les valeurs des propriétés de la ressource

  Ex. *Description des personnes et de leurs pages web*

```
PREFIX moi: <http://monappli.monorg.com/>
PREFIX : <http://monappli.monorg.com/>
DESCRIBE ?x ?y
WHERE {
  ?x a :Personne ;
     moi:pageWeb ?y .
}
```


CONSTRUCT

- Construit un graphe en résultat

  Ex. *Les membres d'un laboratoire avec leur nom, leur directeur de laboratoire et la page web du laboratoire*

```
PREFIX moi: <http://monappli.monorg.com/>
PREFIX : <http://monappli.monorg.com/>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
CONSTRUCT {
  ?m foaf:name ?n ;
     moi:directeur ?d ;
     moi:pageWebLab ?p .
}
WHERE {
  ?l a :Laboratoire ;
     moi:membre ?m ;
     moi:directeur ?d ;
     moi:pageWeb ?p .
  ?m foaf:name ?n .
}
```

Bibliographie

- RDF Primer : http://www.w3.org/TR/rdf-primer/
  - En Français: http://www.yoyodesign.org/doc/w3c/rdf-primer/
- SPARQL
  - SPARQL Query Language: http://www.w3.org/TR/rdf-sparql-query/
  - En Français: http://www.yoyodesign.org/doc/w3c/rdf-sparql-query/
  - SPARQL by Example: http://www.cambridgesemantics.com/semantic-university/sparql-by-example