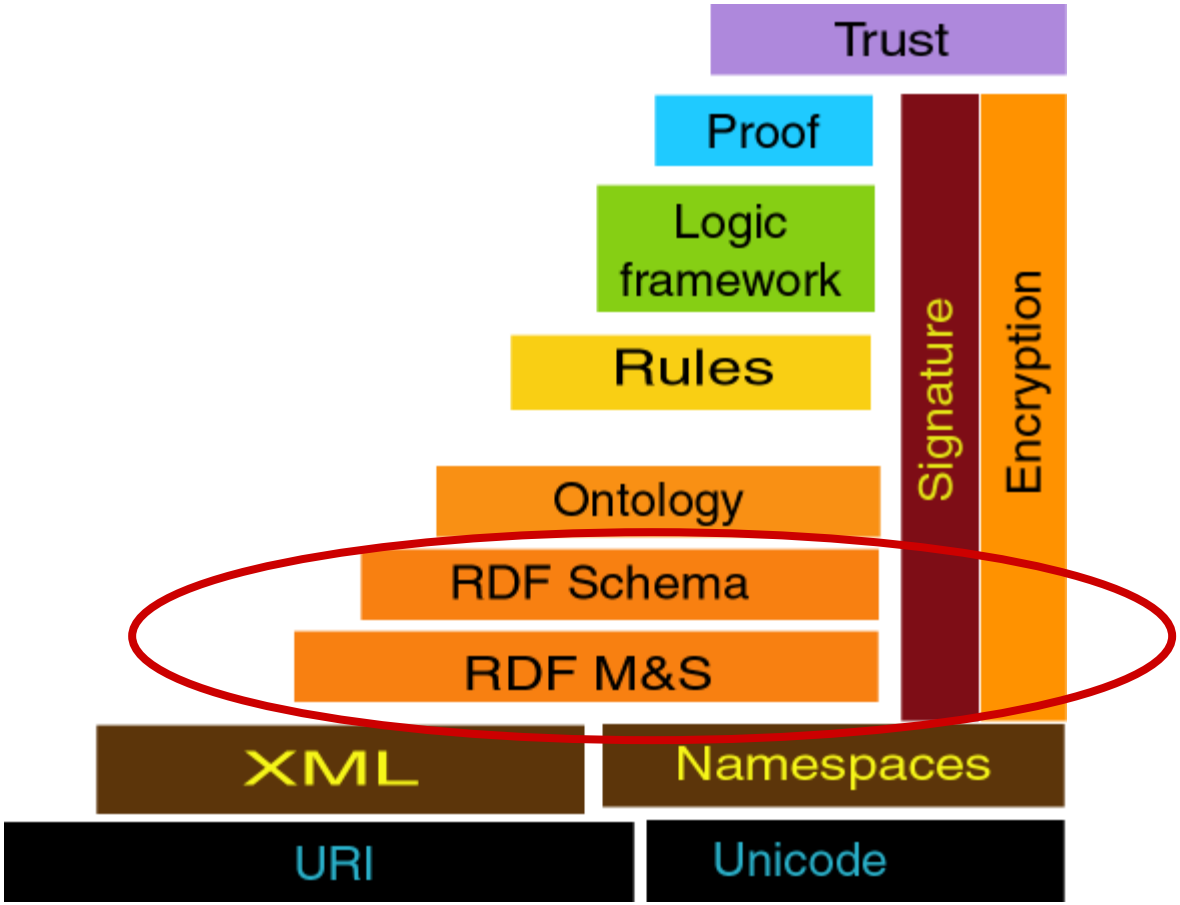
Ontologies et Web sémantique

Cours 6: RDF & RDFS

Dr. TA Tuan Anh
ttanh@ciid.vast.vn

Rappel


Trust
Proof
Logic framework
Rules
Ontology
RDF Schema
RDF M&S
XML
Namespaces
URI
Unicode
Signature
Encryption

Plan

- Modèle RDF
- Syntaxe de sérialisation
- Schéma RDF (RDFS)
- La sémantique de RDF et RDFS
- Langage de requête SPARQL
- Les outils

Notion

- Ressources
  - Tous les choses identifiables par URI
  - E.g. http://www.w3.org/
- Propriétés
  - Relations binaires entre ressources ou ressources / valeurs littérales
  - Identifiées également par URI
  - E.g., http://purl.org/dc/elements/1.1/author
- Littéraux
  - Valeurs atomiques
  - E.g. "John Smith", "2006-03-07"

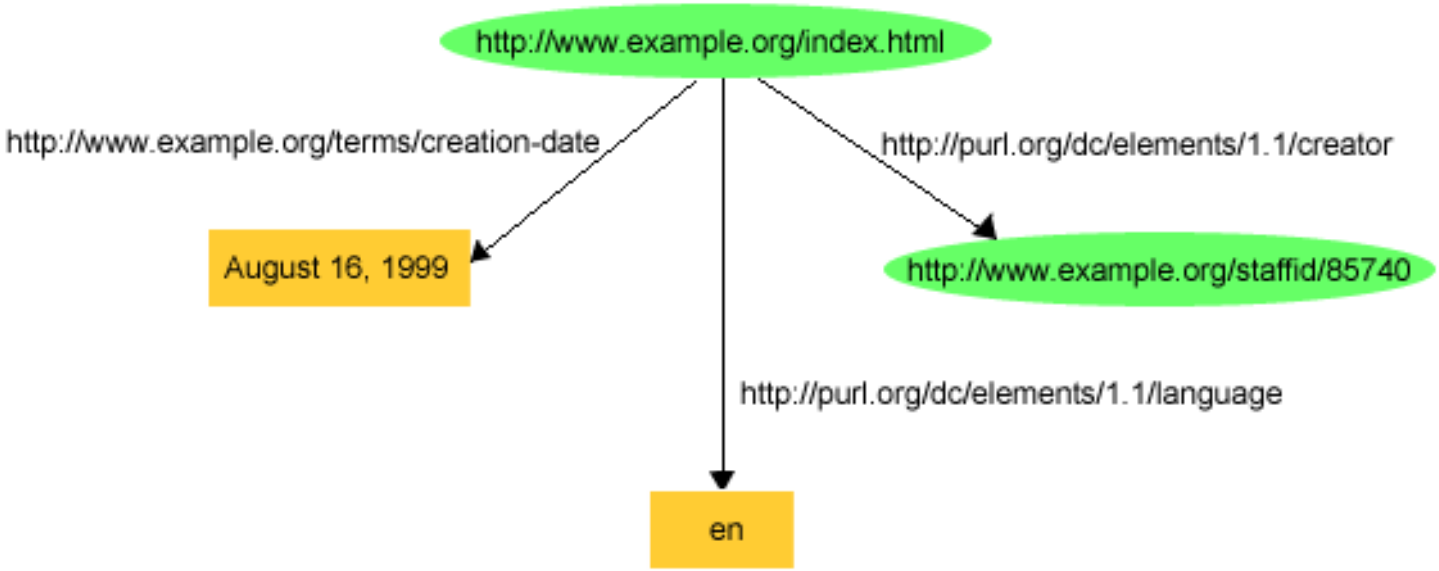
Ressource

- URI (Universal Resource Identification) permet d'identifier les ressources sur le web
  - Une identification donne le sens plus général qu'une location
- Deux types de ressources
  - Ressources adressables identifiées par URL
    - http://www.w3.org/, mailto:anhtt@it-hut.edu.vn
  - Ressources non-adressable identifiées par URN
    - urn:isbn:0-345-33971-1
- RDF permet de donner des énonces sur les ressources, i.e. un sorte de méta données
  - http://www.w3.org/ a le format "text/html"
  - urn:isbn:0-345-33971-1 a l'auteur "Tolkien"

Modèle de données RDF

- RDF est un modèle de triplets simple
  - « sujet - prédicat – objet »
  - « ressource - propriété – valeur »
  - Les valeurs sont soit des ressources, soit des littéraux (valeurs atomiques)
  - Exemples
    - (http://www.w3.org/, http://purl.org/dc/elements/1.1/type, "text/html")
    - (urn:isbn:0-345-33971-1, http://purl.org/dc/elements/1.1/author, "Tolkien")
- Graphe orienté et étiqueté
  - Les nœuds sont soit des ressources, soit des littéraux
  - Les arcs sont étiquetés avec propriétés

Exemple RDF

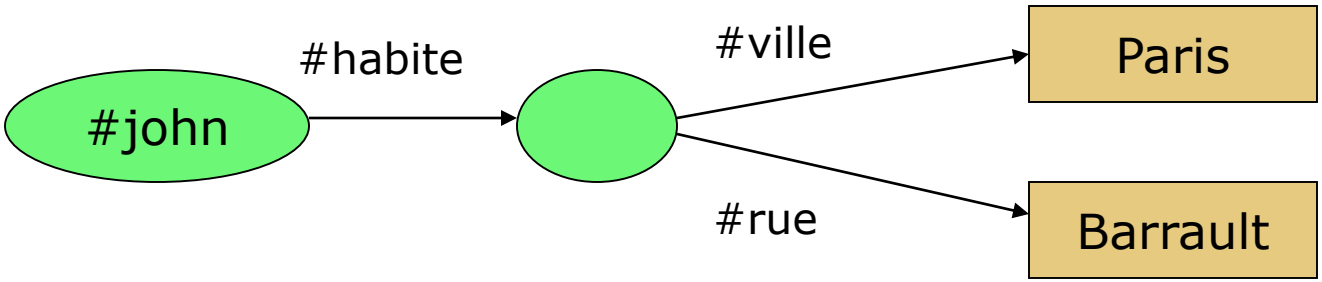


(ex:index.html, dc:creator, exstaff:85740)
(ex:index.html, exterms:creation-date, "August 16, 1999")
(ex:index.html, dc:language, "en")

- namespace ex: http://www.example.org/
- namespace exterms: http://www.example.org/terms/
- namespace exstaff: http://www.example.org/staffid/

Nœuds blancs

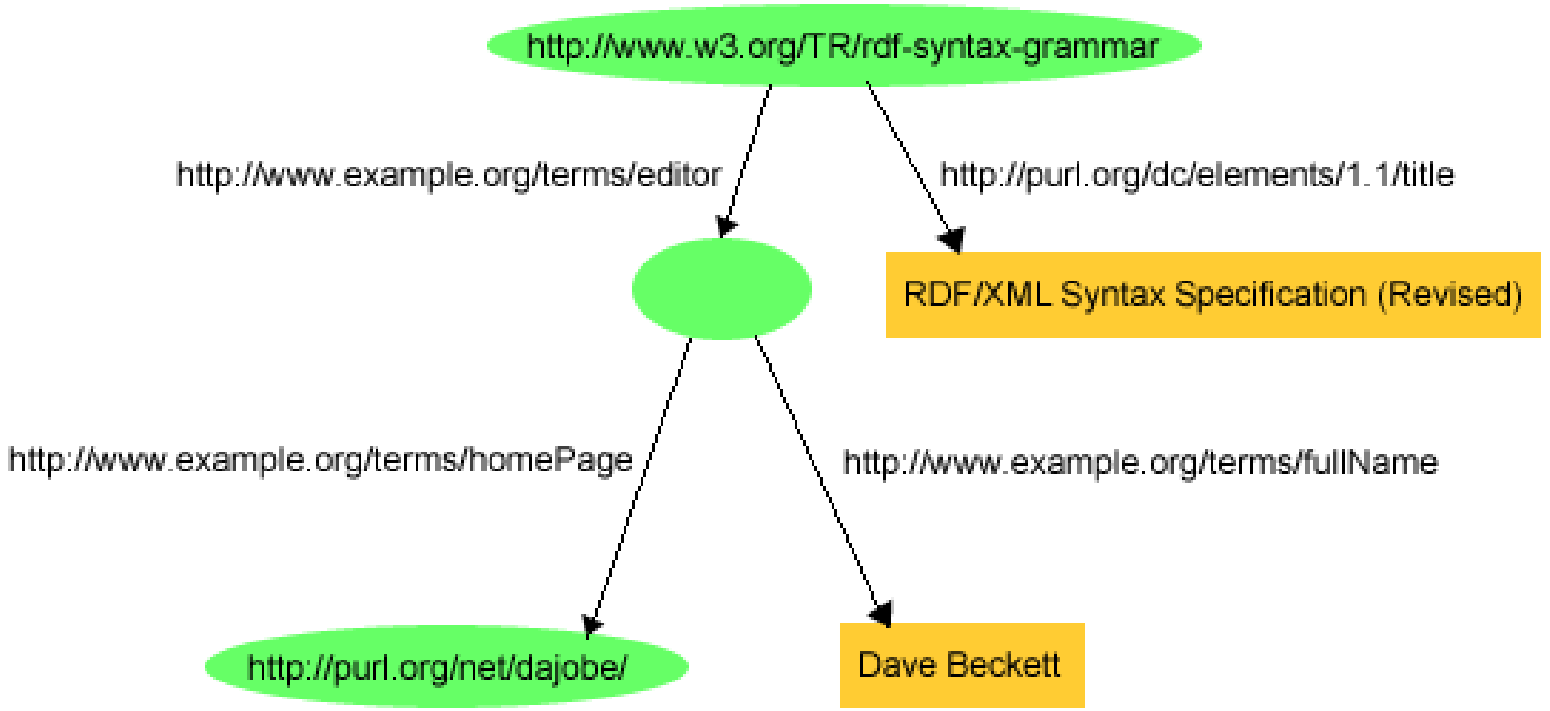
- Utilisés pour représenter des ressources sans URI
- Permettre de définir des descriptions complexes
  - E.g., John habite dans une adresse composée



- Les noeuds blancs ont un identifiant interne (indépendant de la syntaxe)
  - (#john, #habite, _:1001)
  - (_:1001, #rue, ”Barrault”)
  - (_:1001, #ville, ”Paris”)

Exercice

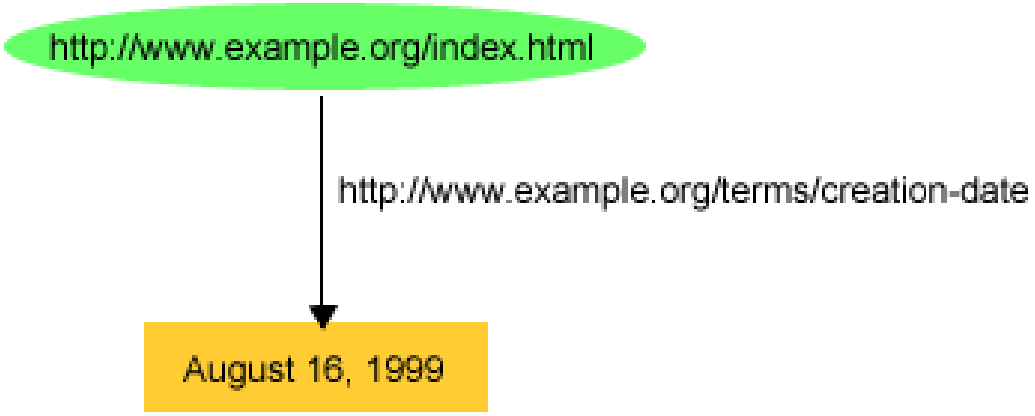
- Quels sont les triplets du graphe



Sérialisation en XML

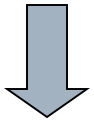
- Sérialiser les données RDF en XML comme le format d'échange sur le Web

```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:exterms="http://www.example.org/terms/">
<rdf:Description rdf:about="http://www.example.org/index.html">
    <exterms:creation-date>August 16, 1999</exterms:creation-date>
</rdf:Description>
</rdf:RDF>
```



Groupement

```
<rdf:Description rdf:about="http://www.example.org/index.html">
    <exterms:creation-date>August 16, 1999</exterms:creation-date>
</rdf:Description>
<rdf:Description rdf:about="http://www.example.org/index.html">
    <dc:language>en</dc:language>
</rdf:Description>
<rdf:Description rdf:about="http://www.example.org/index.html">
    <dc:creator rdf:resource="http://www.example.org/staffid/85740"/>
</rdf:Description>
```

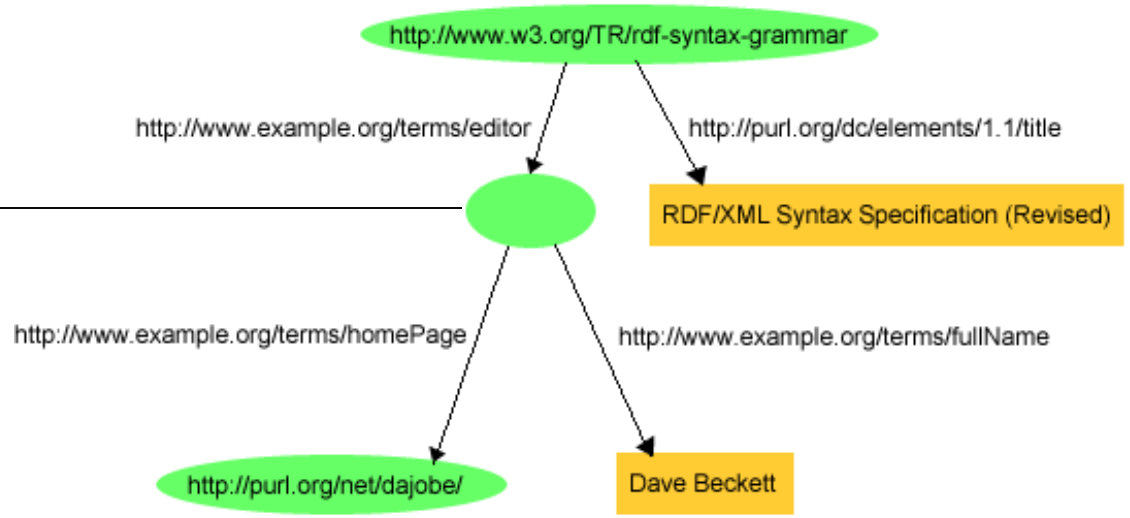


```
<rdf:Description rdf:about="http://www.example.org/index.html">
    <exterms:creation-date>August 16, 1999</exterms:creation-date>
    <dc:language>en</dc:language>
    <dc:creator rdf:resource="http://www.example.org/staffid/85740"/>
</rdf:Description>
```

Exemple (cont.)

```
<rdf:Description rdf:about="http://www.w3.org/TR/rdf-syntax-grammar">
    <dc:title>RDF/XML Syntax Specification (Revised)</dc:title>
<exterms:editor rdf:nodeID="123"/>
</rdf:Description>
<rdf:Description rdf:nodeID="123">
    <exterms:fullName>Dave Beckett</exterms:fullName>
    <exterms:homePage  rdf:resource="http://purl.org/net/dajobe/"/>
</rdf:Description>
```

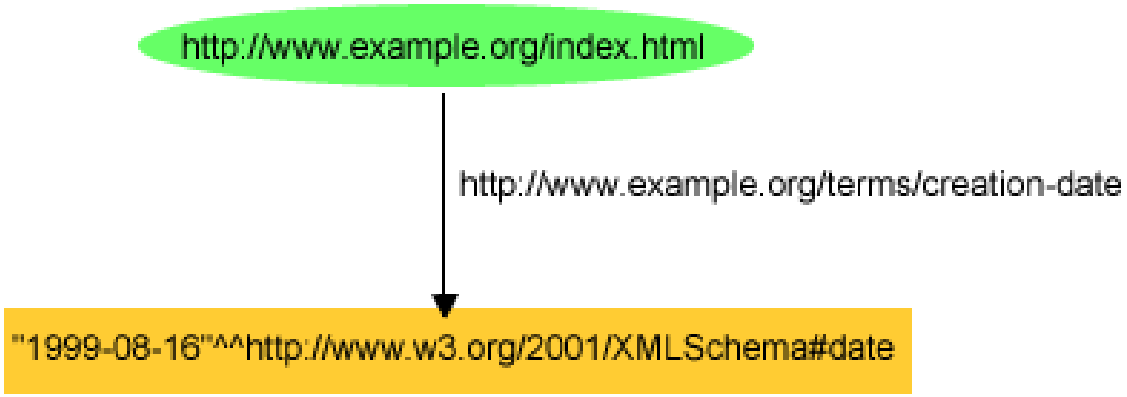
rdf:nodeID="123"



Littéraux typés

- Les valeurs atomiques peuvent être typés avec les types définis par le schéma XML
  - E.g. "hello"^^xsd:string, "1"^^xsd:integer
- Le parseur RDF ne vérifie pas la validité des littéraux typés
  - E.g. "pumpkin"^^xsd:integer est passable
- Les types de base du schéma XML sont recommande'
  - E.g.: xsd:string, xsd:integer, xsd:float, xsd:anyURI, xsd:boolean
- Le type rdf:XMLLiteral représente les littéraux en syntaxe XML

Exemple




```
<?xml version="1.0"?>
<rdf:RDF
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:ex="http://www.example.org/terms/">
<rdf:Description rdf:about="http://www.example.org/index.html">
 <ex:creation-date
      rdf:datatype="http://www.w3.org/2001/XMLSchema#date">1999-08-
      16</ex:creation-date>
</rdf:Description>
</rdf:RDF>
```

XMLLiteral

```
<?xml version="1.0"?>
<rdf:RDF
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:dc="http://purl.org/dc/elements/1.1/"
    xml:base="http://www.example.com/books">
<rdf:Description rdf:ID="book12345">
   <dc:title rdf:parseType="Literal">
      <span xml:lang="en">
         The <em>&lt;br /&gt;</em>
         Element Considered Harmful.
      </span>
   </dc:title>
</rdf:Description>
</rdf:RDF>
```

Type de ressources

- RDF permet de spécifier le typage de ressources en utilisant la propriété rdf:type
- Exemple

  (_:jane, exterms:mailbox, <mailto:jane@example.org>)

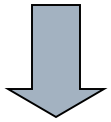
  (_:jane, rdf:type, exterms:Person)

  (_:jane, exterms:name, "Jane Smith")
- exterms:Person est une URI de vocabulaire pour identifier une classe de ressources
- Syntaxe XML

```
<rdf:Description rdf:nodeID="jane">
    <rdf:type rdf:resource="&exterms;Person"/>
    <exterms:name>Jane Smith</exterms:name>
</rdf:Description>
```

Abréviation

```
<rdf:Description rdf:nodeID="jane">
    <rdf:type rdf:resource="&exterms;Person"/>
    <exterms:name>Jane Smith</exterms:name>
</rdf:Description>
```



```
<exterms:Person rdf:nodeID="jane">
    <exterms:name>Jane Smith</exterms:name>
</exterms:Person>
```

NB: Il faut distinguer le namespace exterms et l'entite XML &exterms;

- <!ENTITY exterms "http://www.example.org/terms/">
- xmlns:exterms="http://www.example.org/terms/"

Définition de vocabulaires

- exterms: Person, exterms:name, … et un vocabulaire défini comme des classes et des propriétés qui peuvent être utilisées dans les descriptions RDF
- RDFS (RDF Schema) est un langage pour definir des vocabulaires RDF
  - Classes : comme des ressources de type rdfs:Class
  - Propriétés: comme des ressources de type rdf:Property
  - Relations entre classes ou propriétés
- RDFS est une extension du modèle RDF
  - Tous les définition des classes et des propriétés sont des triplets

xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"

Exemple : classes

(#Student, rdf:type, rdfs:Class)
(#Person, rdf:type, rdfs:Class)
(#Student, rdfs:subClassOf, #Person)

```
<rdfs:Class rdf:ID="Person"/>
<rdfs:Class rdf:ID="Student">
   <rdfs:subClassOf rdf:resource="#Person"/>
</rdfs:Class>
```

- <rdf:ID> est similaire a l'attribut ID dans XML. On utilise rdf:ID pour décrire les ressources dans le namespace local.
  - rdf:resource="#Person" se référence a une URI dans le namespace local

Exemple : propriétés

(#hasName, rdf:type, rdf:Property)
(#hasName, rdfs:domain, #Person)
(#hasName, rdfs:range, xsd:string)

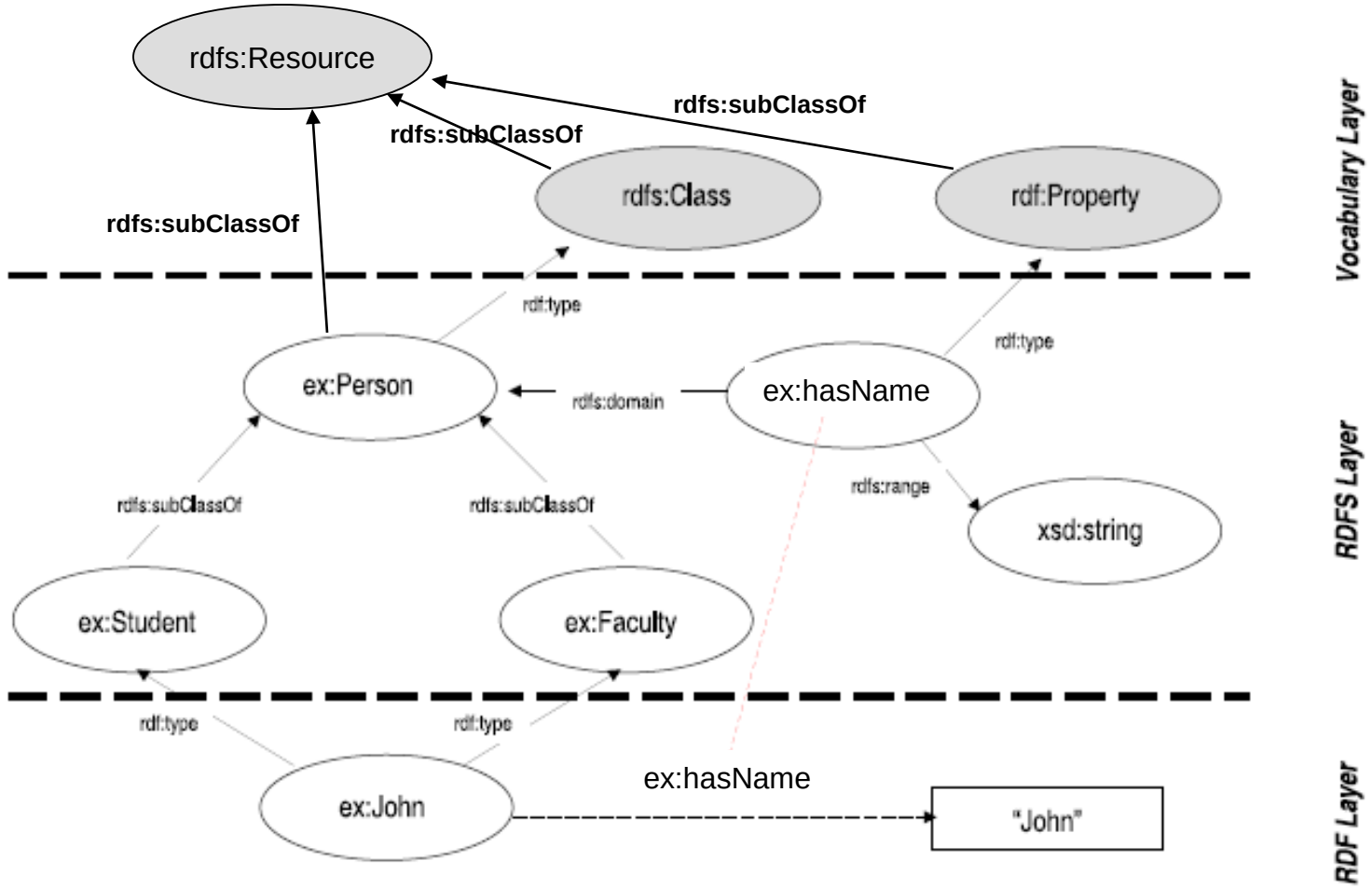
```
<rdf:Property rdf:ID="hasName">
   <rdfs:domain rdf:resource="#Person"/>
   <rdfs:range rdf:resource="&xsd;string"/>
</rdf:Property>
```

```
xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
```

Schéma RDFS

- rdfs:Resource: toutes les ressources sont implicitement des instances de rdfs:Resource
- rdfs:Class: pour décrire une ressource de type classe, une classe est elle-même une ressource
  - rdfs:subClassOf permet de définir des hiérarchies de classes
- rdf:Property: un sous ensemble de ressources est utilisé comme des propriétés dans les descriptions RDF
  - rdfs:domain permet de décrire un domaine appliquant les propriétés
  - rdf:range permet de décrire un domaine de valeurs pour les propriétés
  - rdfs:subPropertyOf permet de définir des hiérarchies de propriétés

Les couches de description



Exercice

- Quel est le graphe RDF(S) de l'exemple suivant

```
<rdfs:Class rdf:ID='Cours'/>
<rdfs:Class rdf:ID='CoursDEA'>
    <rdfs:subClassOf rdf:resource='#Cours'/>
</rdfs:Class>
<rdfs:Class rdf:ID='CoursESSI'>
    <rdfs:subClassOf rdf:resource='#Cours'/>
</rdfs:Class>
<rdfs:Class rdf:ID='CoursCommun'>
    <rdfs:subClassOf rdf:resource='#CoursDEA'/>
    <rdfs:subClassOf rdf:resource='#CoursESSI'/>
</rdfs:Class>
<rdfs:Property rdf:ID='titre'>
    <rdfs:domain rdf:resource='#Cours'/>
    <rdfs:range rdf:resource= '&rdfs;Literal'/>
</rdfs:Property>
```

Exercice (cont.)

```
<rdf:Property rdf:ID='enseignant'>
    <rdfs:domain rdf:resource='#Cours'/>
    <rdfs:range rdf:resource='#Personne'/>
</rdf:Property>
<rdf:Property rdf:ID='nom'>
    <rdfs:domain rdf:resource='#Personne'/>
    <rdfs:range rdf:resource=’&rdfs;Literal'/>
</rdf:Property>
<rdfs:Class rdf:ID='Personne'/>
<rdfs:Class rdf:ID='MaitreDeConf'>
    <rdfs:subClassOf rdf:resource='#Personne'/>
</rdfs:Class>
<rdfs:Class rdf:ID='Chercheur'>
    <rdfs:subClassOf rdf:resource='#Personne'/>
</rdfs:Class>
```

Exercice (cont.)

```
<rdf:Description about='http://www.essi.fr/cours/log11'>
    <rdf:type rdf:resource='#CoursCommun'/>
    <titre>Modélisation des connaissances</titre>
    <num>Log11</num>
    <enseignant>
        <Chercheur
            rdf:about='http://www.inria.fr/Olivier.Corby'>
            <nom>Olivier Corby</nom>
            <institut>INRIA</institut>
        </Chercheur>
    </enseignant>
</rdf:Description>
```

Exercice (cont.)

```
<CoursCommun about='http://www.essi.fr/cours/log11'>
    <titre>Modélisation des connaissances</titre>
    <num>Log11</num>
    <enseignant
    rdf:resource="http://www.inria.fr/Olivier.Corby"/>
</CoursCommun>
<Chercheur  rdf:about='http://www.inria.fr/Olivier.Corby’>
    <nom>Olivier Corby</nom>
    <institut>INRIA</institut>
</Chercheur>
```

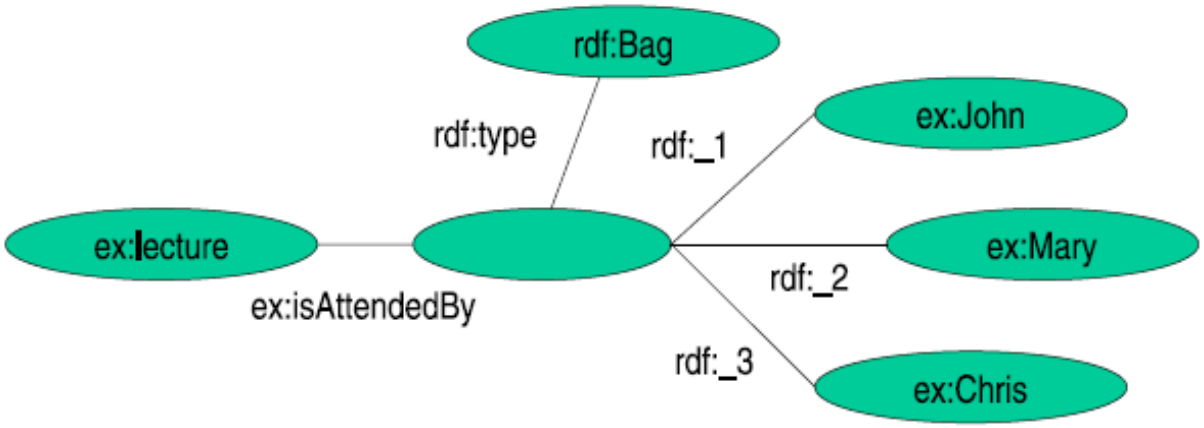
Littéraux en RDFS

- Littéraux doivent être pris comme objets dans les triplets
- Etant donné (_:john, #hasName, “John”)
  - (“John”, rdf:type, rdfs:Literal) est valid?
  - Non! Pas de sujets comme littéraux
- rdfs:Literal permet de définir la range d’une propriété
  - e.g., (#hasName, rdfs:range, rdfs:Literal)
- rdfs:Datatype est une sous classe de rdfs:Literal

Container RDF

- Dans le vocabulaire RDF, il existe 3 type de containers
  - rdf:Bag – un ensemble d'éléments sans ordre
  - rdf:Seq – une liste avec un ordre
  - rdf:Alt – un ensemble alternative
- Tous les containers sont des instances de soit rdf:Bag, soit rdf:Seq, soit rdf:Alt
- rdf:_1, rdf:_2, … sont des propriétés pour décrire les membres d'un container
- Dans la syntaxe XML, on remplace rdf:_1, rdf:_2, … par rdf:li

Exemple : Bag




```
<rdf:Description rdf:about="&ex;lecture">
   <ex:isAttended>
       <rdf:Bag>
          <rdf:li rdf:resource="&ex;John"/>
          <rdf:li rdf:resource="&ex;Mary"/>
          <rdf:li rdf:resource="&ex;Chris"/>
       </rdf:Bag>
    </ex:isAttended>
</rdf:Description>
```

Réification

- Enoncé sur un autre énoncé
  - E.g., Mary dits que le nom de John est ”John Smith”.
- La réification est une instance de rdf:Statement qui indique le sujet (rdf:subject), le predicat (rdf:predicate), l’objet (rdf:object) d’un énoncé

(_:abc, rdf:type, rdf:Statement)
(_:abc, rdf:subject, #John)
(_:abc, rdf:predicate, #hasName)
(_:abc, rdf:object, ”John Smith”)
(_:abc, dc:creator, #mary)

```
<rdf:Statement rdf:nodeID="abc">
   <rdf:subject rdf:resource="#john"/>
   <rdf:predicate rdf:resource="#hasName"/>
   <rdf:object>John Smith</rdf:object>
   <dc:creator rdf:resource="#mary"/>
</rdf:Statement>
```

Vocabulaire RDF

http://www.w3.org/1999/02/22-rdf-syntax-ns#

- Classes
  - rdf:Property
  - rdf:Statement
  - rdf:XMLLiteral
  - rdf:Seq
  - rdf:Bag
  - rdf:Alt
  - rdf:List
- Properties
  - rdf:type
  - rdf:subject
  - rdf:predicate
  - rdf:object
  - rdf:first
  - rdf:rest
  - rdf:_n
  - rdf:value

Vocabulaire RDFS

http://www.w3.org/2000/01/rdf-schema#

- Classes
  - rdfs:Resource
  - rdfs:Class
  - rdfs:Literal
  - rdfs:Datatype
  - rdfs:Container
  - rdfs:ContainerMembershipProperty
- Properties
  - rdfs:domain
  - rdfs:range
  - rdfs:subPropertyOf
  - rdfs:subClassOf
  - rdfs:member
  - rdfs:seeAlso
  - rdfs:isDefinedBy
  - rdfs:comment
  - rdfs:label

Sémantique de RDF(S)

- La sémantique de RDF(S) se repose sur un modèle de graphe
- Un graphe E peut être valide par une interprétation I dans laquelle tous les triplets sont vrais.
- E |= G : pour toutes les interprétations I, si I(E) $\rightarrow$ I(G)
- Axiomes : les triplets sont dans toutes les interprétations
  - Axiomes RDF
  - Axiomes RDFS
- Règles d'inférence : permet d'ajouter de nouveaux triplets a partir de ce qui existe
  - Règles d'inférence RDF
  - Règles d'inférence RDFS

Axiomes RDF

(rdf:type, rdf:type, rdf:Property)
(rdf:subject, rdf:type, rdf:Property)
(rdf:predicate, rdf:type, rdf:Property)
(rdf:object, rdf:type, rdf:Property)
(rdf:first, rdf:type, rdf:Property)
(rdf:rest, rdf:type, rdf:Property)
(rdf:value, rdf:type, rdf:Property)
(rdf:_1, rdf:type, rdf:Property)
(rdf:_2, rdf:type, rdf:Property)
...
(rdf:nil, rdf:type, rdf:List)

Axiomes RDFS

- (rdf:type, rdfs:domain, rdfs:Resource)
- (rdfs:subPropertyOf, rdfs:domain, rdf:Property)
- (rdfs:subClassOf, rdfs:domain, rdfs:Class)
- …
- (rdf:type, rdfs:range, rdfs:Class)
- (rdfs:subPropertyOf, rdfs:range, rdf:Property)
- (rdfs:subClassOf, rdfs:range, rdfs:Class)
- …
- (rdf:Alt, rdfs:subClassOf, rdfs:Container)
- (rdf:Bag, rdfs:subClassOf, rdfs:Container)
- (rdf:Seq, rdfs:subClassOf, rdfs:Container)
- …
- (rdf:_1, rdfs:domain, rdfs:Resource)
- (rdf:_1, rdfs:range, rdfs:Resource)
- (rdf:_1, rdf:type, rdfs:ContainerMembershipProperty)

Axiomes RDFS (cont.)

- rdfs:Resource rdf:type rdfs:Class
- rdfs:Class rdf:type rdfs:Class
- rdfs:Literal rdf:type rdfs:Class
- rdf:XMLLiteral rdf:type rdfs:Class
- rdfs:Datatype rdf:type rdfs:Class
- rdfs:Container rdf:type rdfs:Class
- rdfs:ContainerMembershipProperty rdf:type rdfs:Class
- rdf:Property rdf:type rdfs:Class
- …

- rdfs:domain rdf:type rdf:Property
- rdfs:range rdf:type rdf:Property
- rdfs:subPropertyOf rdf:type rdf:Property
- rdfs:subClassOf rdf:type rdf:Property
- …

Règles d'inférence RDF(S)

- (A, B, C) |= (A, rdf:type, rdfs:Resource)
- (A, B, C) (C is not a literal) |= (C, rdf:type, rdfs:Resource)
- (A,rdf:type,rdfs:Class) |= (A, rdfs:subClassOf, rdfs:Resource)
- (A, rdf:type, B), (B, rdfs:subClassOf, C) |= (A, rdf:type, C)
- (A, rdfs:subClassOf, B), (B, rdfs:subClassOf, C) |= (A, rdfs:subClassOf, C)
- (A, rdf:type, rdfs:Class) |= (A, rdfs:subClassOf, A)
- (A, rdfs:subPropertyOf, B), (B, rdfs:subPropertyOf, C) |= (A, rdfs:subPropertyOf, C)
- (P, rdf:type, rdf:Property) |= (P, rdfs:subPropertyOf, P)
- (P, rdfs:domain, C), (A, P, B) |= (A, rdf:type, C)
- (P, rdfs:range, C), (A, P, B) |= (B, rdf:type, C)
- etc

RDFS entailment rules (2)

- (A, rdf:type, rdfs:Class) |= (A, rdfs:subClassOf, A)
- (A, rdfs:subPropertyOf, B), (B, rdfs:subPropertyOf, C) |= (A, rdfs:subPropertyOf, C)
- (P, rdf:type, rdf:Property) |= (P, rdfs:subPropertyOf, P)
- (P, rdfs:domain, C), (A, P, B) |= (A, rdf:type, C)
- (P, rdfs:range, C), (A, P, B) |= (B, rdf:type, C)
- …

Exemple

<http://example.org/#john> rdf:type <http://example.org/#Student>
<http://example.org/#Student> rdfs:subClassOf <http://example.org/#Person>

entails

<http://example.org/#john> rdf:type <http://example.org/#Person>

<http://example.org/#hasName> rdfs:domain <http://example.org/#Student>
<http://example.org/#mary> <http://example.org/#hasName> “Mary”

entails

<http://example.org/#mary> rdf:type <http://example.org/#Student>

<http://example.org/#john> <http://example.org/#hasMother> <http://example.org/#mary>
<http://example.org/#hasMother> rdfs:subPropertyOf <http://example.org/#hasParent>

entails

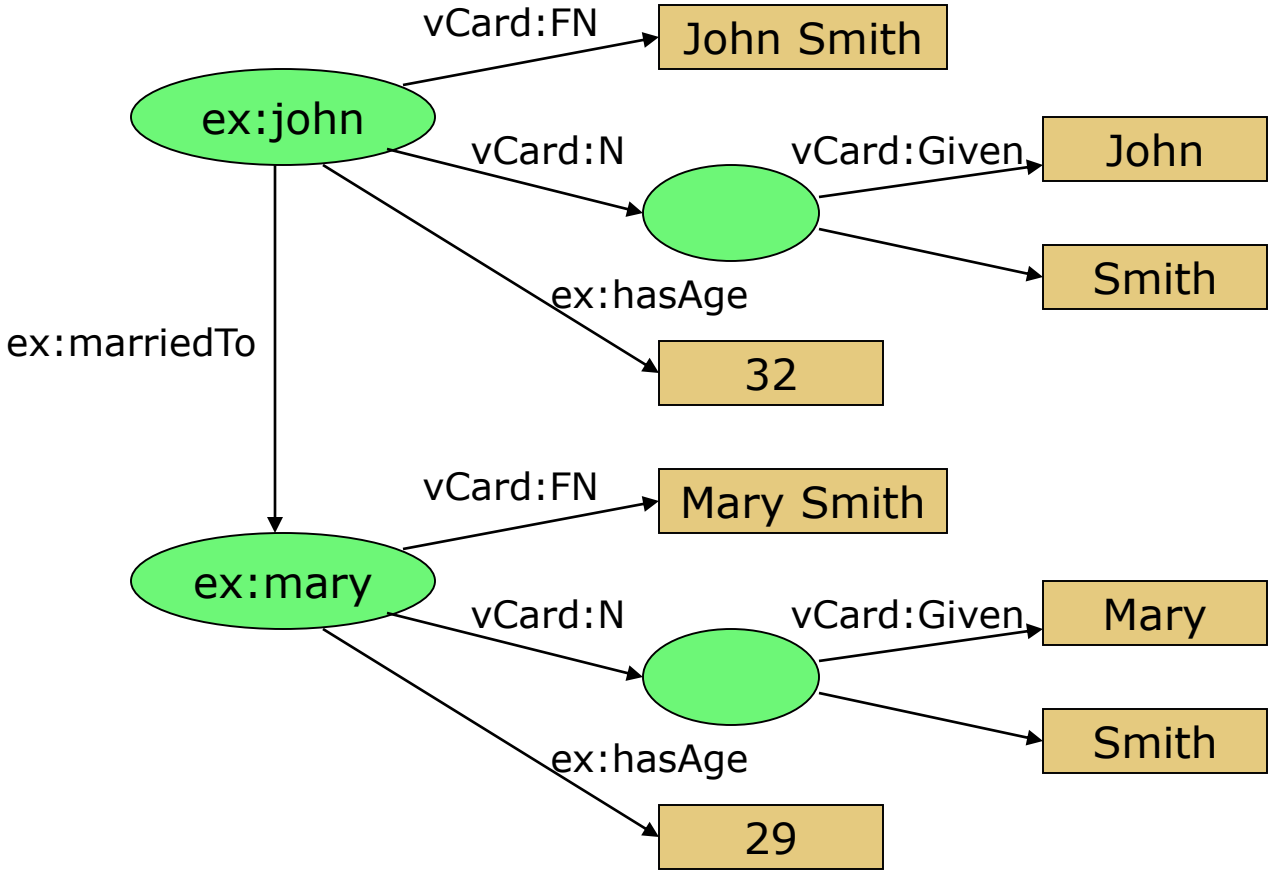
<http://example.org/#john> <http://example.org/#hasParent <http://example.org/#mary>

SPARQL

- Langage de requête RDF propose' par W3C
  - Base' sur RDQL
  - Syntaxe SQL-like
- SPARQL est aussi un protocole de manipulation a distance a base de service web
- Syntaxe: [PREFIX …] SELECT … WHERE …
  - PREFIX déclare des namespaces
  - SELECT identifies des variables de projection
  - WHERE spécifie un patron de sélection
- Exemple

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

Exemple : Base RDF



Exemple: valeur

"Return the full names of all people in the graph"

```
PREFIX vCard: <http://www.w3.org/2001/vcard-rdf/3.0#>
SELECT ?fullName
WHERE {?x  vCard:FN  ?fullName}
```

result:

```
fullName
==================
"John Smith"
"Mary Smith"
```

Exemple: propriété

"Return the relation between John and Mary"

```
PREFIX ex: <http://example.org/#>
SELECT ?p
WHERE {ex:john  ?p  ex:mary}
```

result:

```
p
==================
<http://example.org/#marriedTo>
```

Exemple: jointure

"Return the spouse of a person by the name of John Smith"

```
PREFIX vCard: <http://www.w3.org/2001/vcard-rdf/3.0#>
PREFIX ex: <http://example.org/#>
SELECT ?y
WHERE {?x vCard:FN "John Smith".
       ?x ex:marriedTo ?y}
```

result:

```
y
==================
<http://example.org/#mary>
```

Exemple: nœud blanc

"Return the name and the first name of all people in the KB"

```
PREFIX vCard: <http://www.w3.org/2001/vcard-rdf/3.0#>
SELECT ?name, ?firstName
WHERE {?x vCard:N  ?name .
       ?name vCard:Given ?firstName}
```

result:

```
name firstName
================
_:a "John"
_:b "Mary"
```

Exemple: contraint

"Return all people over 30 in the KB"

```
PREFIX ex: <http://example.org/#>
SELECT ?x
WHERE {?x hasAge  ?age .
       FILTER(?age > 30)}
```

result:

```
x
==================
<http://example.org/#john>
```

Autres langages de requête

- RDQL
  - Langage de base pour SPARQL
  - Outil: Jena
- RQL, SeRQL
  - A base de OQL
  - Outils: RDFSuite, Sesame
- Versa
  - A base de XPath
  - Outil: 4Suite
- TRIPLE
  - A base de F-Logic

References en plus

- Consulter le wikipedia pour avoir plus de references vers outils et etudes de cas
  - http://en.wikipedia.org/wiki/SPARQL

Ressources bibliographiques

- RDF Primer: http://www.w3.org/TR/rdf-primer/
- RDF Concepts and abstract syntax: http://www.w3.org/TR/rdf-concepts/
- RDF/XML syntax specification: http://www.w3.org/TR/rdf-syntax-grammar/
- RDF Vocabulary Description Language 1.0: RDF Schema: http://www.w3.org/TR/rdf-schema/
- RDF Semantics: http://www.w3.org/TR/rdf-mt/
- SPARQL Query Language for RDF: http://www.w3.org/TR/rdf-sparql-query/
- SPARQL Protocol for RDF: http://www.w3.org/TR/rdf-sparql-protocol/
- "Ontology Storage and Querying", Technical Report, ICS-FORTH, 2002