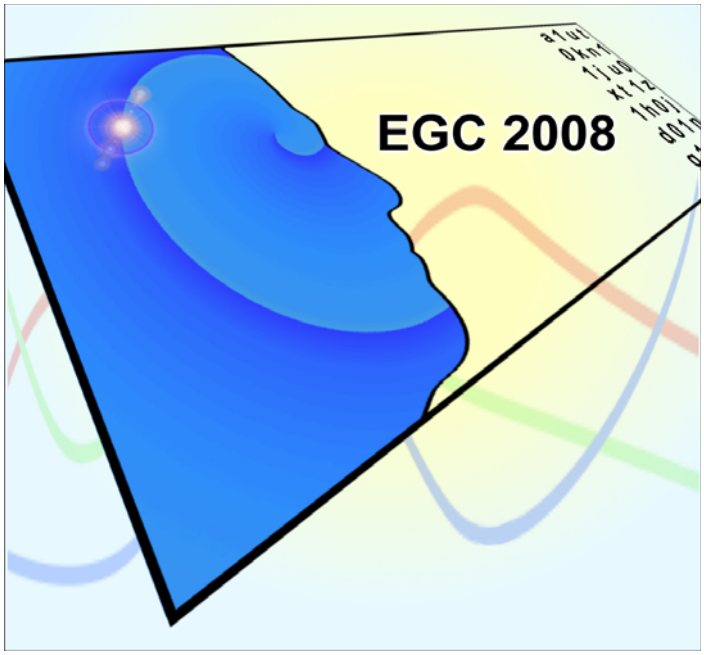
8èmes Journées Francophones
Extraction et Gestion des Connaissances
Sophia Antipolis 29 janvier 2008

Tutoriel



Initiation aux standards du Web Sémantique

RDF/S, SPARQL, OWL, Règles, GRDDL et RDFa

Organisateurs : Fabien Gandon (INRIA), Olivier Corby (INRIA) et Catherine Faron-Zucker (I3S)

Responsables des tutoriels EGC
Hicham Behja (INRIA, Sophia Antipolis)
Bernard Senach (INRIA, Sophia Antipolis)

http://www-sop.inria.fr/axis/egc08

Initiation aux standards du Web Sémantique

RDF/S, SPARQL, OWL, Règles, GRDDL et RDFa

Fabien Gandon(*), Olivier Corby(*) et Catherine Faron-Zucker (* & **)

(*) INRIA Sophia Antipolis, Projet Edelweiss,
2004, route des Lucioles, BP 93
06902 Sophia Antipolis, France
{Fabien.Gandon,Olivier.Corby}@sophia.inria.fr
(**) MAINLINE - I3S UNSA
930 route des Colles, bât. ESSI, BP 145
06903 Sophia Antipolis cedex
Catherine.Faron@essi.fr

Résumé. Les langages du web sémantique continuent à être intégrés dans de plus en plus d'applications rendant ainsi explicites des données et des schémas de données qui jusque là restaient enfouis au cœur des implantations. En ouvrant leurs données à tout le monde, ces applications permettent la création d'autres applications pouvant nourrir ou se nourrir de ces mêmes données. Les langages du web sémantique ouvrent donc de nouvelles perspectives d'interopérabilité à l'échelle du web ; un web de données.
Quatre langages sont maintenant des recommandations du W3C pour permettre ce web de données: le modèle RDF et sa syntaxe XML; SPARQL, le langage de requêtes pour RDF; RDFS pour la description d'ontologies légères; OWL et ses trois couches d'expressivité, pour une définition plus formelle des ontologies.
Ce tutoriel est une visite guidée de ces quatre langages et ce termine par un aperçu des évolutions en cours: l'extension OWL 1.1; RIF et les règles; RDFa pour intégrer du RDF aux pages web; GRDDL pour extraire du RDF d'autres langages XML.

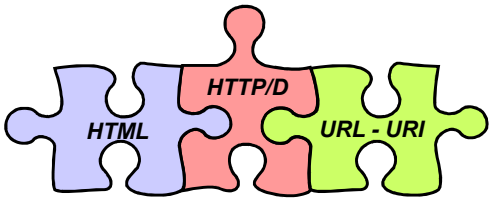
Le matériel pédagogique du tutoriel est disponible en ligne :

http://www-sop.inria.fr/edelweiss/wiki/wakka.php?wiki=TutorielEGC2008

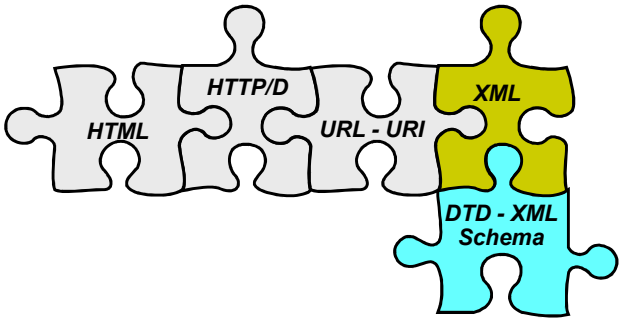
Mots-clés
Web sémantique, RDF/S, SPARQL, OWL, GRDDL, RDFa

Et l'Homme créa le Web

Une introduction historique.



- Une couche applicative au dessus d'Internet:
- **HTML** 1.0 (1992) HyperText Markup Language
  http://www.w3.org/History/19921103-hypertext/hypertext/WWW/MarkUp/MarkUp.html
- Uniform Resource Locator (**URL**)
  RFC 1738 Dec. 1994
- HTML 2.0: RFC 1866 Novembre 1995
- **HTTP** 1.0: RFC 1945 in Mai 1996
- HTML 3.2 Recommandation 1997 (1996)
  - Normaliser les extensions les plus courantes
    Tableaux, texte autour images, applets, indices/exposants
  - Evolutions suivantes (Scripts, Stylesheets)



- Extensible Markup Language 1.0 Rec. 1998
  XML 1.0 (Fourth Edition) W3C Recommendation 16 August 2006
  XML 1.1 (Second Edition), W3C Recommendation, 16 August 2006
  - Un format textuel d'échange de données structurées
  - Standard pour définir des langages balisés

```
<user>
  <id>fgandon</id>
  <home>/fg</home>
</user>
```

- Structurer ≠ présenter : données et structures ***vs.*** affichage ou traitement
- XML ≠ HTML (fond / forme, contenu / présentation données & structures / affichage & disposition)
- Méta-langage / format / famille de langages balisés: MathML, CML, SVG, XMI, P3P, XACML, SAML, SMIL, BPML, XSLT, …

- Définir ses tag / balises / étiquettes / éléments

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<post_it>
  <urgent />
  <sujet>billets d'avion</sujet>
  <date>2005-11-28</date>
  <message>tes billets sont sur mon
  bureau</message>
</post_it>
```

- Version et encodage
- Eléments en XML bien formé : une seule racine, balises emboîtées, balises fermées, balises vides, majuscule/minuscules, pas commencer par un chiffre ou par "xml", pas d'espaces dans les noms de balises.

- Paramétrer / préciser une balise

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<post_it importance="urgent">
  <sujet>billets d'avion</sujet>
  <date>2005-11-28</date>
  <message xml:lang="fr">tes billets sont sur mon
  bureau</message>
</post_it>
```

- Attributs en XML bien formé:
  - attributs dans une balise ouvrante ou vide
  - valeurs des attributs entre guillemets ou apostrophes
- Balises vs. attributs:
  - Les attributs donnent une structure plus simple
  - Les attributs ne sont pas extensibles

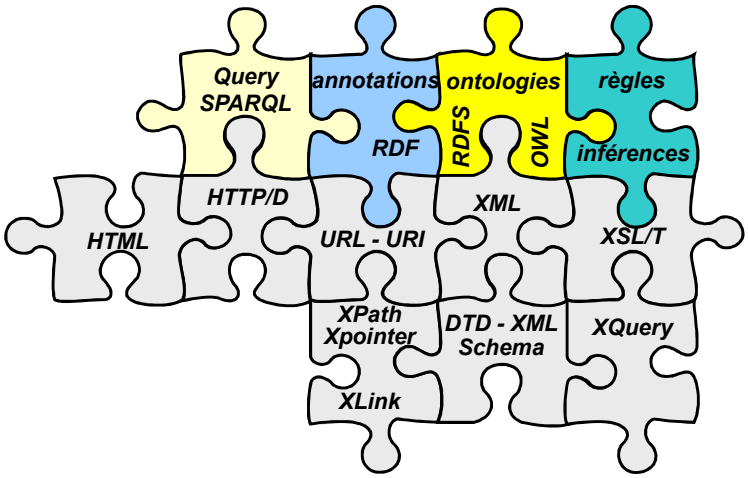
Attributs des balises 7

- Un document bien formé respecte le format XML
- Un document valide est un document bien formé qui respecte une DTD ou un Schéma XML
- DTD / Schéma: standardiser et échanger structures
  - DTD: balises autorisées, attributs et enchaînements
  - XML Schema est son successeur: une syntaxe XML et des extensions (datatypes, types complexes, etc.)
- Uniform Resource Identifier
  (URI RFC 2396 Août 1998, RFC 3986 Janvier 2005)
- Identificateur unique d'une ressource abstraite ou physique exemple:
  `ldap://[2001:db8::7]/c=GB?objectClass?one`
- URL = sous ensemble des URI (ID et localisation)

Bien formé vs. valide 8

- Espaces de nommage: éviter les conflits de noms
  Namespaces in XML (Second Edition), W3C Recommendation, 16 August 2006
  Namespaces in XML 1.1 (Second Edition), W3C Recommendation, 16 August 2006
  - Identifier, réutiliser, combiner les définitions des balises
- Principe: un nom local (préfix) associé à un namespace (URI)
  - Identifier vocabulaires, éviter les collisions, qname
  - Namespace : l'URI. ex : `http://www.ugb.sn/recherche/lani`
  - Préfix + nom de balise = nom qualifié ex: `<ugb:note />`
    `<ugb:note xmlns:ugb='http://www.ugb.sn/'>18</ugb:note>`
  - Définitions héritées dans l'arbre XML
  - Namespace par défaut `xmlns="…"`
- Association à un schéma XML:
  `<schema targetNamespace="http://www.ugb.sn/recherche/lani">`

Définitions nommées (namespaces / espaces de nommage) *9*



Another brick in the… *10*

Une brève introduction aux ontologies

Un petit peu de sémantique peut vous emmener très loin.

Bruit ≠ Précision

Manqué ≠ Rappel

Agences l'RAM

La Galère
148, rue Victor Hugo
76600 Le Havre

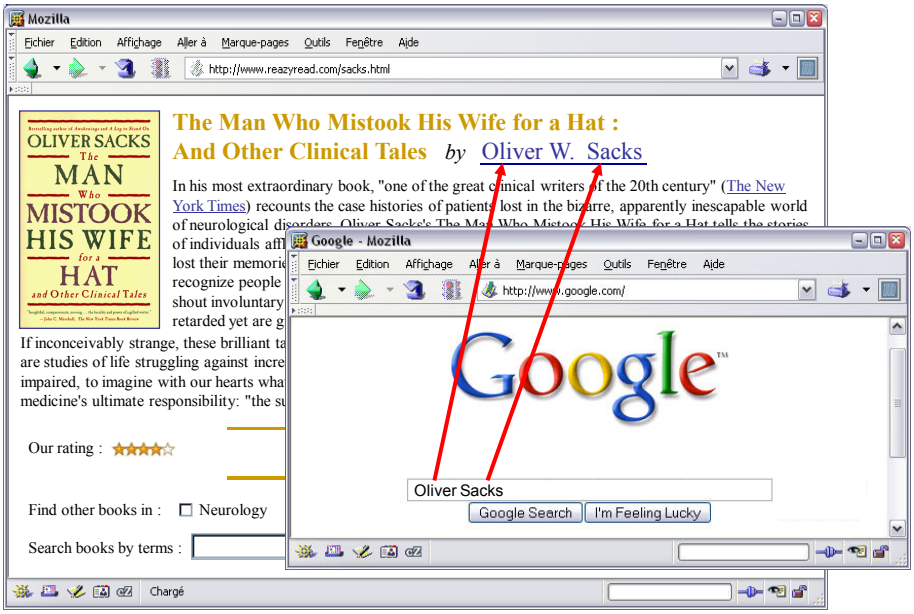
L'Agence de la Presse et des Livres
38, rue Saint Dizier BP 445
54001 Nancy Cédex

RESUME DU ROMAN DE VICTOR HUGO

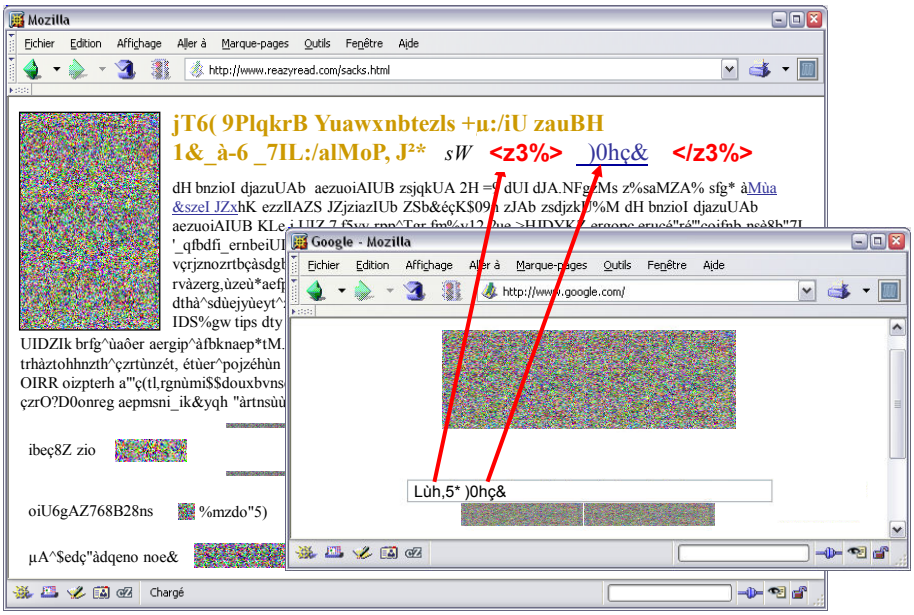
NOTRE DAME DE PARIS (1831) - 5 parties

L'enlèvement . Livres 1-2 : 6 janvier 1482. L'effrayant bossu Quasimodo

Exemple simple d'un problème… *12*



Le Web pour nous… 13



Le Web pour les machines… 14

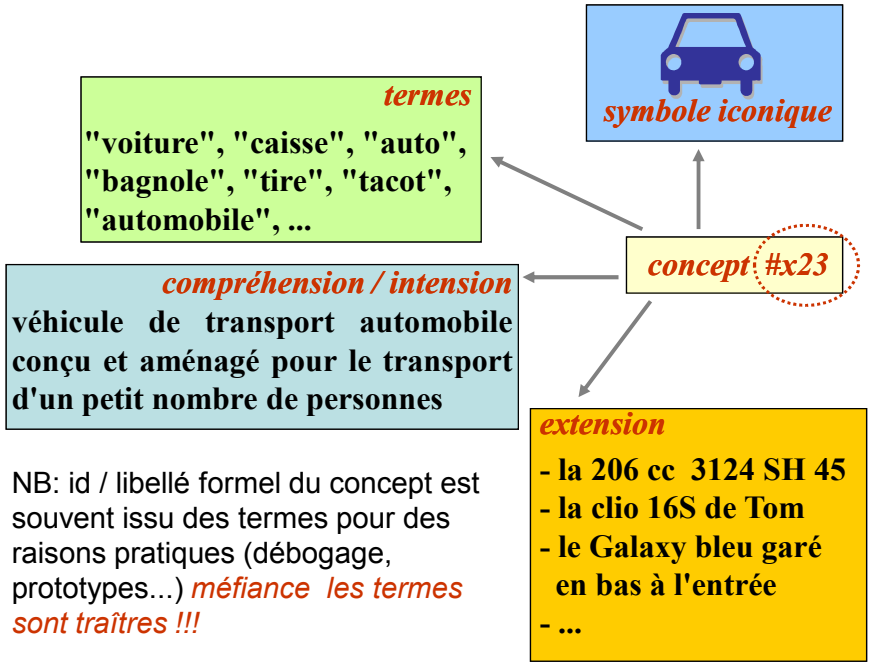
Ne lisez pas le panneau suivant.

15

16

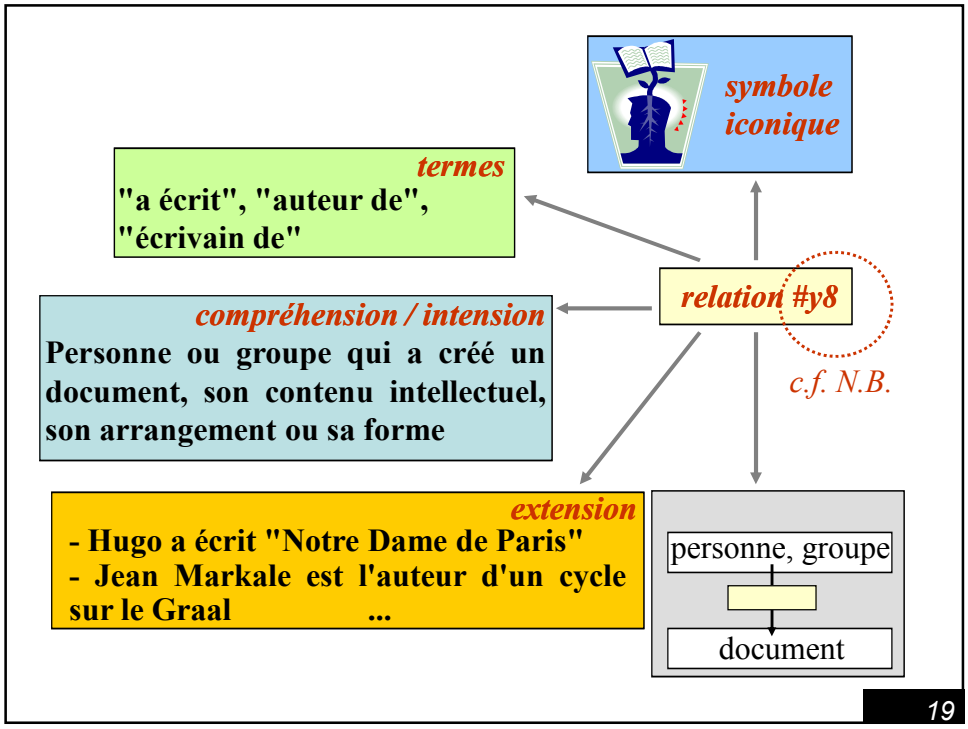
Nous interprétons,
les machines non.

17



NB: id / libellé formel du concept est souvent issu des termes pour des raisons pratiques (débogage, prototypes...) *méfiance les termes sont traîtres !!!*

18

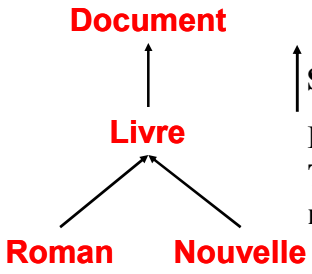


- Le dernier document que vous avez lu ?
- Réponse basée sur structuration des concepts:
  - objets / catégorie & identification
  - hiérarchie de catégories : structure d'abstraction spécialisation / généralisation
- Réponse basée sur un consensus (émetteur, public, récepteur)
- Cette structure et ce consensus sont ce que l'on appelle une 'ontologie'

- Manque une connaissance → identification
- Types de documents → acquisition
- Modéliser et formaliser → représentation

"Un roman et une nouvelle sont des livres."
"Un livre est un document."

Informel



Formel

Ontologie & subsomption 21

- Manque une connaissance → identification
- Types de documents → acquisition
- Modéliser et formaliser → représentation

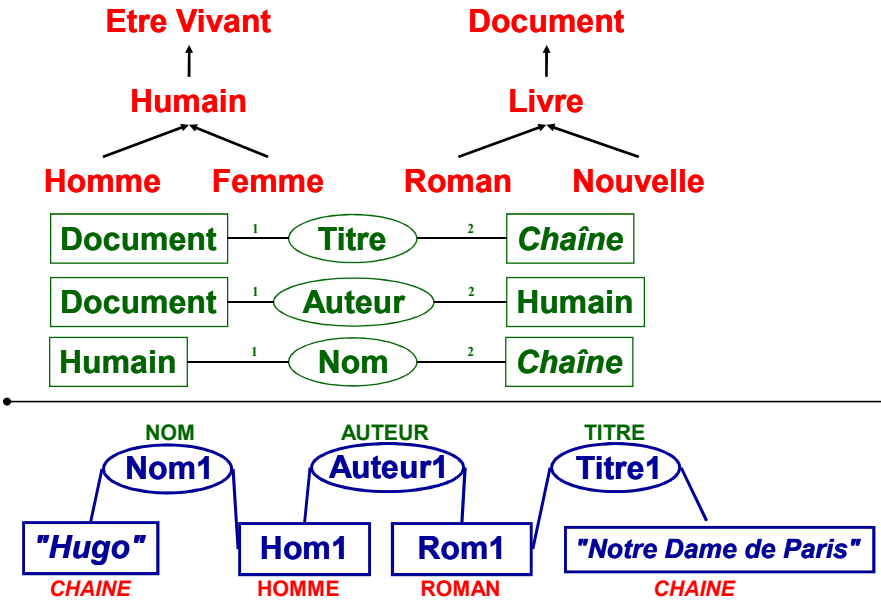
"Un document a un titre.
Un titre est une chaîne de caractères"

Informel

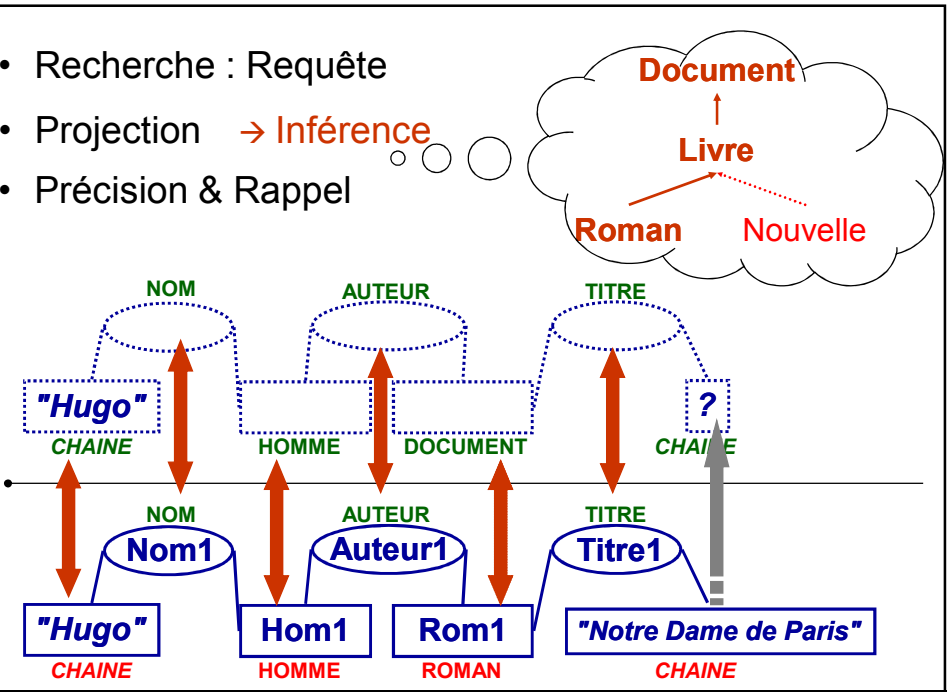
Formel

Ontologie & relation binaire 22

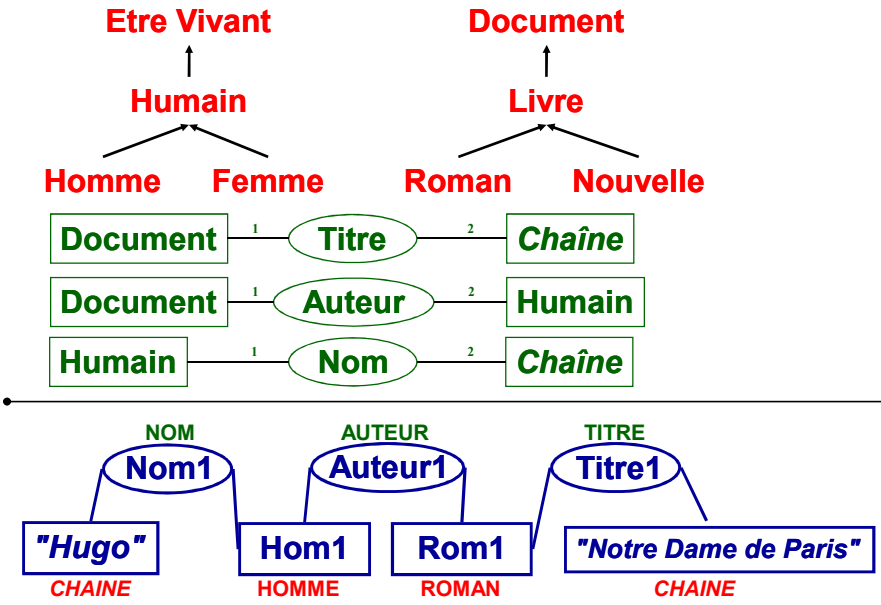
Hugo est l'auteur de Notre Dame de Paris



- Recherche : Requête
- Projection → Inférence
- Précision & Rappel

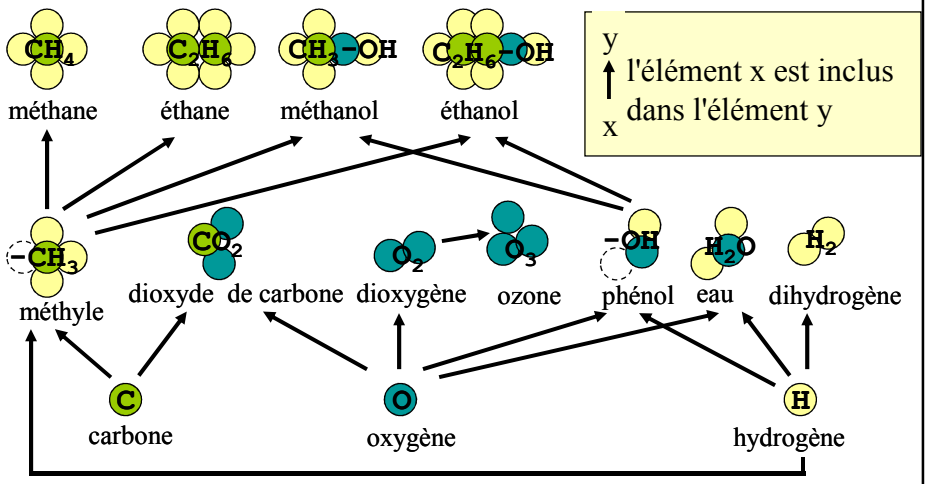


Hugo est l'auteur de Notre Dame de Paris



Une ontologie n'est pas,
une taxonomie.

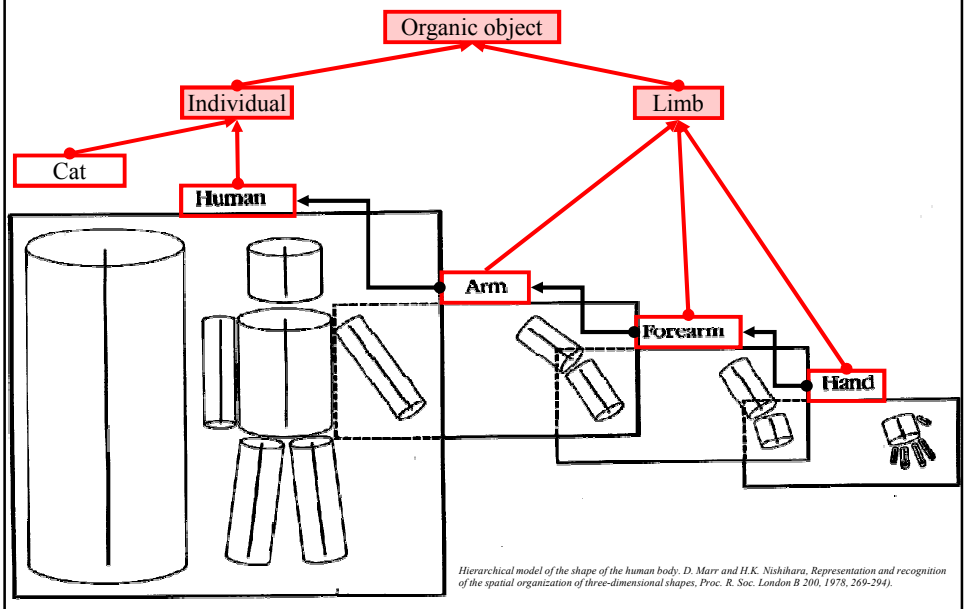
- organisation par composition
- donc nouveau type de lien (transitif réflexif)



- recherche "hydrogène sur la lune" ?

Exemple de partonomie 27

- Une ontologie peut contenir différents types de connaissances



combiner 28

- Pousser plus loin l'utilisation d'une logique:

  `personne (x) < être_vivant (x)`

  `directeur (x) := personne(x) ∧ organisation(y) ∧ dirige (x,y)`

  – Types primitifs / définis, axiomes règles
- Modèles causaux:

  `manger salé = cause de soif`

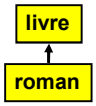
  `soif = cause pour boire`

  `manger salé = cause pour boire`
- Parfois des instances / objets globaux
  – ex. constantes (g, c, etc...)
  – objet unique ex. un thème "les mathématiques"
- …

- Logiques des prédicats

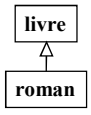
  `(∀x) (Roman(x) ⊃ Livre(x))`
- Graphes Conceptuels

  `Roman < Livre`



- Langages à objets

  `public class Roman`
  `extends Livre`



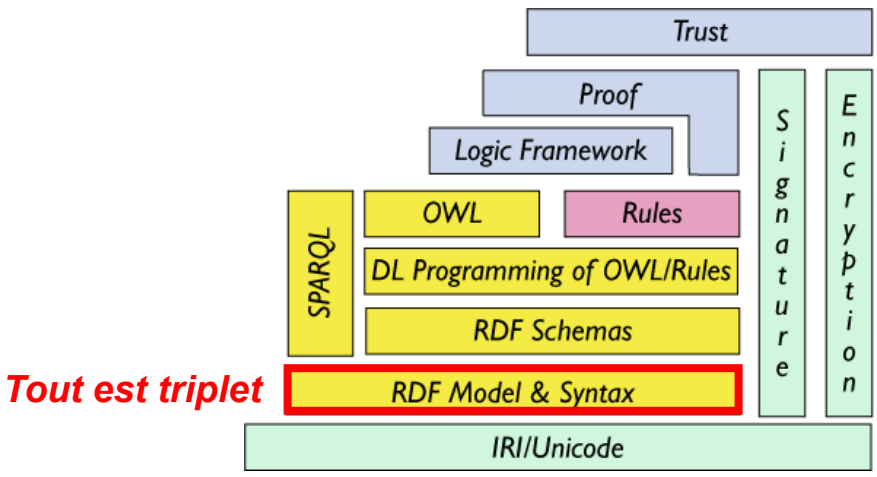
- Logiques de descriptions

  `Roman ≤ (and Livre (not Essai))`
- Web Semantique RDFS et OWL

```
<rdfs:Class rdf:ID="Roman">
 <rdfs:label xml:lang="en">novel</rdfs:label>
 <rdfs:label xml:lang="fr">roman</rdfs:label>
 <rdfs:subClassOf rdf:resource="Livre"/>
</rdfs:Class>
```

Resource Description Framework

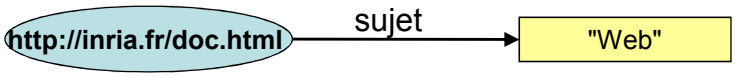
Présentation détaillée du model RDF pour l'annotation de ressources sur le web.



W3C, T Berners-Lee, Ivan Herman

- Langage de représentation de connaissances
  - sur les ressources localisées sur le Web (par une URL)
  - sur des entités identifiées sur le Web (par une URI)
- Standard permettant l'échange de métadonnées sur le web et donc leur traitement automatique
- Modèle & syntaxe d'annotation :
  - Modèle simple avec une sémantique formelle (Graphes RDF)
  - Format d'échange basé sur une syntaxe XML (RDF/XML)
- Modèle ouvert:
  - Vocabulaire extensible basé sur les URI et XML schema datatypes
  - Autorise quiconque à faire des déclarations sur n'importe quelle ressource

- Représenter un ensemble d'assertions à propos de ressources (Web)
  - Ex: doc.html a pour auteur Fabien et parle du Web
- Les assertions donnent des propriétés des ressources et leurs valeurs ex: doc.html a pour sujet "Web"
  - propriétés binaires (relient 1 ressource à 1 valeur)



  - les valeurs de propriétés sont :
    des littéraux (chaîne de caractères) "Web"
    ou
    des ressources http://inria.fr/cv.html
- En ajoutant des propriétés chacun peut participer

- Les assertions peuvent être décomposées en triplets de la forme `(sujet, propriété, valeur)` (statement)
- Ex: doc.html a pour auteur Fabien et parle du Web

```
(http://inria.fr/doc.html , auteur , urn://~fgandon)
(urn://~fgandon , nom , "Fabien")
(http://inria.fr/doc.html , sujet , "Web")
```

- Les règles des triplets/ sont:
  - Le sujet est toujours une ressource (pas un littéral)
  - La propriété binaire est d'un type identifié par une URI
  - La valeur est une ressource ou un littéral
- Les ressources sont identifiables par des URI
  - Si l'URI est un URL alors ressource du Web
  - Si non, URI d'une ressource physique, abstraite, etc.
  - Jointure entre les assertions même si elles sont distribuées
  - Liant entre les couches réseau (Internet, Web, Web sémantique)
  - Blank node : ressource anonyme i.e. pas d'URI

Modèle de triplets 35

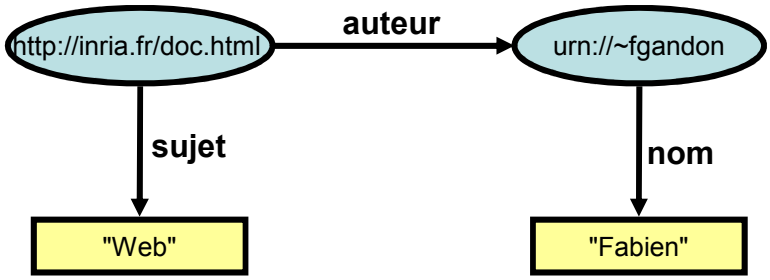
- Chaque triplet représente un prédicat binaire en logique

```
(http://inria.fr/doc.html , auteur , urn://~fgandon)
(urn://~fgandon , nom , "Fabien")
(http://inria.fr/doc.html , sujet , "Web")

auteur(http://inria.fr/doc.html, urn://~fgandon)
nom(urn://~fgandon, "Fabien")
sujet(http://inria.fr/doc.html, "Web")
```

- Sémantique formelle: RDF sous-ensemble logique du premier ordre
  - Avec: prédicats binaires, quantification existentielle(∃),conjonction
  - Sans: disjonction, négation, quantification universelle (∀)
- La quantification existentielle (∃) est introduite par les blank nodes / ressources anonymes.
- Tout énoncé RDF est considéré comme vrai et RDF est monotone i.e. ce qui est vrai et ce que l'on peut déduire reste vrai si l'on rajoute de nouveaux énoncés.

Modèle logique 36

- Modèle de graphe: jointure /conjonction de triplets
- Chaque triplet représente un arc étiqueté entre deux sommets d'un graphe orienté pas forcément connexe:
  - (http://inria.fr/doc.html , auteur , urn://~fgandon)
  - (urn://~fgandon , nom , "Fabien")
  - (http://inria.fr/doc.html , sujet , "Web")



Modèle de graphe 37

- Triplets/graphe RDF peuvent être sérialisés en un arbre XML
- Une racine unique et un namespace:

```
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">(…)</rdf:RDF>
```

- Sérialisation pas unique (syntaxe, graphe/arbre)!

```
<rdf:Description rdf:about="http://inria.fr/doc.html">
  <auteur>
    <rdf:Description rdf:about="urn://~fgandon">
      <nom>Fabien</nom>
    </rdf:Description>
  </auteur>
   <sujet>Web</sujet>
</rdf:Description>
<rdf:Description rdf:about="http://inria.fr/doc.html">
  <auteur rdf:resource="urn://~fgandon"  />
   <sujet>Web</sujet>
</rdf:Description>
<rdf:Description rdf:about="urn://~fgandon">
  <nom>Fabien</nom>
</rdf:Description>
<rdf:Description rdf:about="urn://~fgandon" nom="Fabien" />
```

- Autre syntaxe plus humaine: N3

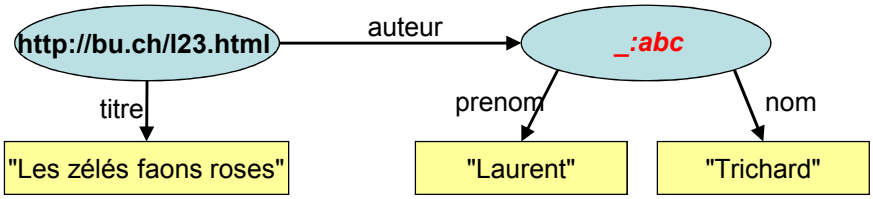
```
<http://inria.fr/doc.html> auteur <urn://~fgandon>
<urn://~fgandon> nom "Fabien"
<http://inria.fr/doc.html> sujet "Web"
```

Syntaxe XML 38

- Une ressource peut ne pas être identifiée ; sémantique = quantification existentielle
  il existe une resource telle que… { ∃ r ; …}

```
<rdf:Description rdf:about="http://bu.ch/l23.html ">
  <auteur>
   <rdf:Description>
     <nom>Trichard</nom>
     <prenom>Laurent</prenom>
   </rdf:Description>
  </auteur>
  <titre>Les zélés faons roses</titre>
</rdf:Description>
```

∃ x ; auteur(http://bu.ch/l23.html,x)
nom(x,"Trichard")
prenom(x,"Laurent")



Les nœuds anonymes (blank nodes) 39

- Utilisation d'un ID local pour identifier les blank nodes d'un graphe (vital pour les sérialisations)

```
<rdf:Description rdf:about="http://bu.ch/123.html ">
  <auteur rdf:nodeID="abc123"/>
  <titre>Les zélés faons roses</titre>
</rdf:Description>

<rdf:Description rdf:nodeID="abc123">
  <nom>Trichard</nom>
  <prenom>Trichard</prenom>
</rdf:Description>
```

  (Équivalent au premier cas car le parseur créait l'ID)
- Omettre un blank node dans la syntaxe XML:

```
<rdf:Description rdf:about="http://bu.ch/123.html ">
  <auteur rdf:parseType="Resource" >
    <nom>Trichard</nom>
    <prenom>Laurent</prenom>
  </auteur>
  <titre>Les zélés faons roses</titre>
</rdf:Description>
```

- Exercice: décrivez une personne et son adresse.

Les nœuds anonymes avec ID ! 40

- Réification d'un triplet: rendre un triplet explicite pour pouvoir en parler i.e. l'utiliser comme le sujet ou l'objet d'une propriété.
  - Un triplet est réifié par un statement
  - Le statement fait du triplet une ressource
  - Cette ressource peut être décrite à son tour

```
<rdf:Statement rdf:nodeID="decFab">
 <rdf:subject rdf:resource="http://inria.fr/doc.html" />
 <rdf:predicate rdf:resource="&dc;auteur"/>
 <rdf:object rdf:resource="urn://~fgandon" />
</rdf:Statement>

<rdf:Description rdf:nodeID="decFab">
  <auteur rdf:resource="http://inria.fr/"/>
</rdf:Description>
```

Réification d'un statement 41

- Obliger le parseur à ignorer la structure du contenu

```
<rdf:Description rdf:ID="reportR-25">
  <dc:title rdf:parseType="Literal">
    The world <i>wild</i> web
  </dc:title>
 </rdf:Description>
```

- XML schema datatypes
  - Les littéraux standards sont des chaînes de caractères
  - Pour typer les valeurs littérales, RDF repose sur les datatypes de XML Schema

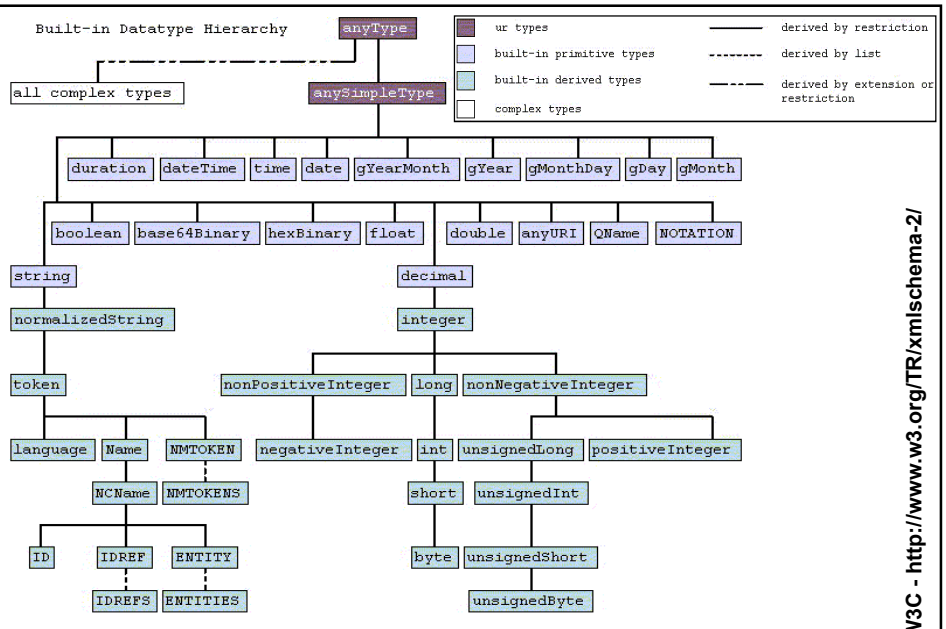
- Notation en N3:

```
c:id1 c:age "22"^^xsd:integer
```

Forcer une valeur littérale & datatypes 42

- Syntaxe XML pour les datatypes en RDF

```
<rdf:Description rdf:about="#Fabien">
 <faitDesEnseignements
  rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean">
  true</faitDesEnseignements>
 <naissance
  rdf:datatype="http://www.w3.org/2001/XMLSchema#date">
  1975-07-31</naissance>
 <langueMaternelle
  rdf:datatype="http://www.w3.org/2001/XMLSchema#language">
  fr</langueMaternelle>
</rdf:Description/>

<!ENTITY xsd "http://www.w3.org/2001/XMLSchema#" > (…)
<rdf:Description rdf:about="#Fabien">
 <faitDesEnseignements rdf:datatype="&xsd;#boolean">
  true</faitDesEnseignements>
 <naissance rdf:datatype="&xsd;#date">
  1975-07-31</naissance>
(…)
</rdf:Description/>
```

- Exercice: livre, titre, date, disponible ou pas

- On peut spécifier une langue avec xml:lang

```
<Livre>
 <titre xml:lang='fr'>Seigneur des
anneaux</titre>
 <titre xml:lang='en'>Lord of the rings</titre>
</Livre>
```

- En N3

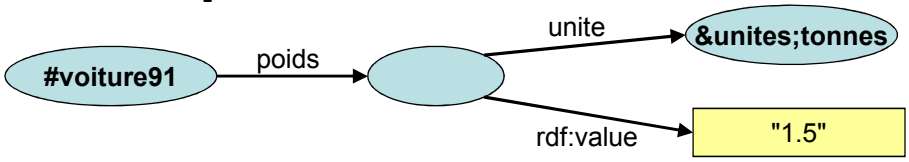
```
c:book c:title "Lord of the rings"@en
```

- Attention: les littéraux avec langue et sans langue sont différents

```
"Fabien" ≠ "Fabien"@en ≠ "Fabien"@fr
```


- Relations n-aires dans le cas d'une valeur litérale *ou* valeur complexe dans une propriété
  - Sélectionner un sujet principal
  - Réifier la relation par une ressource anonyme
  - Déclarer de propriétés pour chaque autre valeur

```
<rdf:Description rdf:about="#voiture91">
  <poids rdf:parseType="Resource">
   <rdf:value rdf:datatype="&xsd;decimal">1.5</rdf:value>
   <unite rdf:resource="&unites;tonnes"/>
  </poids>
</rdf:Description>
```



- Exercice: patient, température, tendance.

- On peut typer les ressources en utilisant des URI pour identifier les types

```
<urn://~fgandon> rdf:type <http://www.ugb.sn/schema#Personne>
```

- L'instanciation d'un type suffit à faire exister une ressource

```
_:x rdf:type http://www.ugb.sn/schema#Personne
```

- Une ressource peut avoir plusieurs types

```
<urn://~fgandon> rdf:type <http://www.ugb.sn/schema#Personne>
<urn://~fgandon> rdf:type <http://www.ugb.sn/schema#Chercheur>
<urn://~fgandon> rdf:type <http://www.ugb.sn/schema#Enseignant>
```

- Syntaxes XML:

```
<rdf:Description rdf:about="urn://~fgandon">
 <rdf:type rdf:resource="http://www.ugb.sn/schema#Personne" />
 <nom>Fabien</nom>
</rdf:Description>

<sn:Personne rdf:about="urn://~fgandon">
  <nom>Fabien</nom>
</sn:Personne>
```

Typage de ressource 47

- Un groupe de ressources ou littéraux sans ordre

```
<rdf:Description rdf:about="http://www.inria.fr/rrrt/rr-5663.html">
 <auteur>
   <rdf:Bag>
     <rdf:li>Moussa Lo</rdf:li>
     <rdf:li>Fabien Gandon</rdf:li>
  </rdf:Bag>
 </auteur>
</rdf:Description>
```

```
<http://www.inria.fr/rrrt/rr-5663.html> auteur _:a
_:a rdf:_1 "Moussa Lo"
_:a rdf:_2 "Fabien Gandon"
```

Groupe simple sans ordre ni sens 48

- Groupe ordonné de ressources ou littéraux

```
<rdf:Description rdf:about="#partition">
  <contient>
   <rdf:Seq>
    <rdf:li rdf:about="#Do"/>   rdf:_1
    <rdf:li rdf:about="#Do"/>   rdf:_2
    <rdf:li rdf:about="#Do"/>   rdf:_3
    <rdf:li rdf:about="#Re"/>   rdf:_4
    <rdf:li rdf:about="#Mi"/>   rdf:_5
   </rdf:Seq>
 </contient>
</rdf:Description>
```

- Accès: `rdf:_1`, `rdf:_2`, `rdf:_3`, `rdf:_4`, etc.

Séquence 49

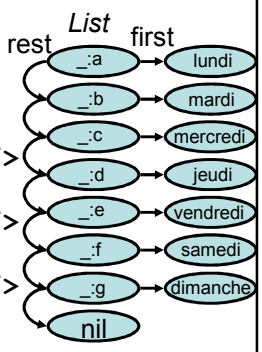
- Groupe de ressources ou littéraux alternatifs
  i.e. une seule valeur est la bonne
  ex: le titre d'un livre en plusieurs langues

```
<rdf:Description rdf:about="#livre">
  <titre>
   <rdf:Alt>

    <rdf:li xml:lang="fr">l'homme qui prenait sa femme
                          pour un chapeau</rdf:li>

    <rdf:li xml:lang="en">the man who mistook his wife
                          for a hat</rdf:li>

   </rdf:Alt>
 </titre>
</rdf:Description>
```

Alternatives 50

- Liste exhaustive et ordonnée de constituants (pour fermer une assertion)

```
<rdf:Description rdf:about="#Semaine">
 <seDiviseEn rdf:parseType="Collection">
  <rdf:Description rdf:about="#Lundi"/>
  <rdf:Description rdf:about="#Mardi"/>
  <rdf:Description rdf:about="#Mercredi"/>
  <rdf:Description rdf:about="#Jeudi"/>
  <rdf:Description rdf:about="#Vendredi"/>
  <rdf:Description rdf:about="#Samedi"/>
  <rdf:Description rdf:about="#Dimanche"/>
 </seDiviseEn>
</rdf:Description>
```



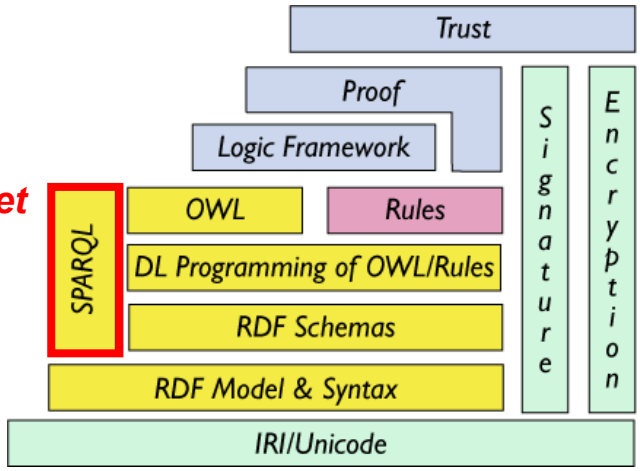
- First / Rest : Le premier et le reste (rdf:List / rdf:nil)
- A la LISP:

```
(Lundi (Mardi (Mercredi (Jeudi (Vendredi
(Samedi (Dimanche (NIL)))))))))
```


Requêtes sur RDF

Le langage de requêtes SPARQL
SPARQL Protocol And RDF Query Language

Tout est triplet



W3C, T Berners-Lee, Ivan Herman

- Trois morceaux:
  - Langage de requête avec syntaxe triplets simplifiée
    SPARQL QUERY LANGUAGE FOR RDF W3C REC 15 JAN. 2008
  - Protocole d'accès comme un service Web (SOAP)
    SPARQL PROTOCOL FOR RDF W3C REC 15 JAN. 2008
  - Langage de présentation des résultats
    SPARQL QUERY RESULTS XML FORMAT W3C REC 15 JAN. 2008
- Langage de requête sur des données RDF
  - Appariements de graphes / projection
  - Principe courant en trois clauses imitant SQL:
    - Select: clause sélectionnant les valeurs à retourner
    - Where: un pattern de triplets à matcher
    - Filter: des contraintes exprimées avec des fonctions de tests internes (XPath 2.0) ou externes

- Exemple de la syntaxe en triplets simplifiée avec des points d'interrogation pour marquer les variables:

```
?x rdf:type  ex:Personne
```

- Langage de patterns à matcher:

```
select ?sujet ? propriete ?valeur where
{?sujet      ?propriete     ?valeur}
```

- Le pattern est par défaut une conjonction de triplets

```
{ ?x    rdf:type        ex:Personne .
  ?x    ex:nom          ?nom }
```

- Deux formes possibles pour présentation résultats:
  - le binding i.e. la liste des valeurs sélectionnées pour chaque réponse rencontrée;
    (format XML stable ; bien avec XSLT)
  - les sous graphes des réponses rencontrées en RDF
    (format RDF/XML ; bien pour applications utilisant RDF)

- 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"      x2
_: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>
```

- Noms et prénoms des auteurs:

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

- Pour utiliser des namespaces:

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

- Namespace de base : BASE <>

Requête simple et namespace 57

- 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>
```

Exemple de binding 58

- Soit un schéma avec les relations "pere_de", "mere_de" et les types "Homme" et "Femme", demandez les femmes et leurs parents

```
PREFIX ex: <http://www.exemple.abc#>
SELECT ?femme ?pere ?mere
WHERE {
 ?femme rdf:type ex:Femme .
 ?mere ex:mere_de ?femme .
 ?pere ex:pere_de ?femme .
}
```


- 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" .
```

- Resource 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 iut: <http://www.iutnice.fr#>
SELECT ?etudiant
FROM http//www.iutnice.fr/data.rdf
WHERE {
 ?etudiant iut:inscrit ?x .
 ?x iut:siteweb http//www.iutnice.fr .
}
```

- Parties optionnelles

```
PREFIX iut: <http://www.iutnice.fr#>
SELECT ?etudiant ?nom
WHERE {
 ?etudiant iut:inscrit ?x .
 ?x iut:siteweb http//www.iutnice.fr .
 OPTIONAL {?etudiant iut:nom ?nom . }
 }
```

Source et Pattern optionnel 61

- Soit un schéma avec les relations "marie_avec", "nom" et les types "Homme" et "Femme", demandez le nom des hommes et optionnellement le nom de leur femme

```
PREFIX ex: <http://www.exemple.abc#>
SELECT ?nom_homme ?nom_femme
WHERE {
 ?homme rdf:type ex:Homme .
 ?homme ex:nom ?nom_homme .
 OPTIONAL {
   ?homme ex:marie_avec ?femme .
   ?femme ex:nom ?nom_femme . }
}
```

Exercice 62

- Donner des patterns alternatifs:

```
PREFIX iut: <http://www.iutnice.fr#>
SELECT ?etudiant
WHERE {
 ?etudiant iut:inscrit ?x .
 {
  {
   ?x iut:siteweb http//www.iutnice.fr .
  }
  UNION
  {
   ?x iut:siteweb http//www.fac.fr .
  }
 }
}
```

Union 63

- Soit un schéma avec les types "Homme", "Femme", et "Enfant" demandez le nom des femmes et des enfants

```
PREFIX ex: <http://www.exemple.abc#>
SELECT ?nom
WHERE {
 ?femme_ou_enfant ex:nom ?nom .
 {
  {
   ?femme_ou_enfant rdf:type ex:Femme .
  }
  UNION
  {
   ?femme_ou_enfant rdf:type ex:Enfant .
  }
 }
}
```

Exercice 64

```
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

- Soit un schéma avec le type "Personne", et la propriété "taille" (en centimètres) demandez les personnes entre 140 et 170 centimètres

```
PREFIX ex: <http://www.exemple.abc#>
SELECT ?x
WHERE {
 ?x rdf:type ex:Personne
 ?x ex:taille ?t
 FILTER ( ?t >= 140 && ?t <= 170 )
}

xsd:integer(?t)
```


- Tester si un pattern est introuvable:

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

- Soit un schéma avec le type "Personne", et la propriété "marie_avec" demandez les personnes non mariées

```
PREFIX ex: <http://www.exemple.abc#>
SELECT ?x
WHERE {
 ?x rdf:type ex:Personne .
 OPTIONAL { ?x ex:marie_avec ?y . }
 FILTER (! bound(?y))
}
```


- **Le négation par l'échec n'est pas une négation absolue**

```
PREFIX ex: <http://www.exemple.abc#>
SELECT ?personne
WHERE {
 ?personne rdf:type ?type .
 FILTER ! ( ?type = ex:Homme )
}
```

- Vérifier qu'il existe au moins une réponse:

```
PREFIX iut: <http://www.iutnice.fr#>
ASK {
 ?etudiant iut:inscrit ?x .
 ?x iut:siteweb www.iutnice.fr .
 ?etudiant iut:age ?age .
 FILTER (?age > 30)
}
```

Que demande cette requête ?

- 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>
```


- On peut créer un format de sortie de toute pièce:

```
PREFIX iut: <http://www.iutnice.fr#>
CONSTRUCT
{ ?etudiant rdf:type iut:FuturTechnicien . }
WHERE {
 ?etudiant iut:inscrit ?x .
 ?x iut:siteweb http://www.iutnice.fr .
}
```

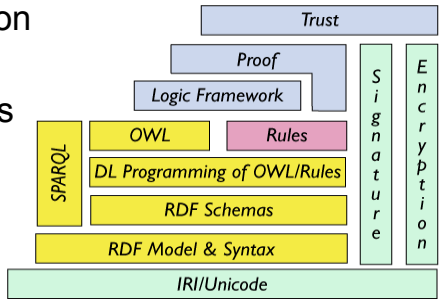
- On peut demander une description générale:

```
PREFIX iut: <http://www.iutnice.fr#>
DESCRIBE ?etudiant
WHERE {
 ?etudiant iut:inscrit ?x .
 ?x iut:siteweb http://www.iutnice.fr .
}
```

Les ontologies dans le web sémantique

Les formalismes proposés par le W3C pour l'échange d'ontologies sur le web.

- Les 4 principaux standards du Web sémantique
  - RDF: un modèle de triplets pour décrire et connecter des ressources anonymes ou identifiées par un URI `(sujet, prédicat, objet)` / graphe orienté étiqueté
  - SPARQL: un langage de requête sur les graphes RDF
  - RDFS est un langage de déclarations et descriptions légères; typage des ressources et de leurs relations `subClassOf, subPropertyOf, range, domain`
  - OWL: 3 couches d'extension de l'expressivité (logique)
  - Un modèle en couche dans une direction d'extension; RDF sans RDFS, RDFS sans OWL, …



Le ou la tour des standards du Web sémantique.

- Nommer et définir un vocabulaire conceptuel consensuel et faire des inférences élémentaires
  - Nommer les classes de ressources existantes
  - Nommer les relations qui existent entre ces classes et donner leur signature
  - Liens hiérarchiques entre classes **et entre propriétés**
  - *Donner un URI aux concepts qui vous sont importants*
- Proche mais différent des modèles objets: propriétés en dehors des classes, multi-instanciation, héritage multiple classes et propriétés, inférences positives monotones, conjonctives
- Squelette taxonomique d'une ontologie

RDF Schema: ontologies légères 75

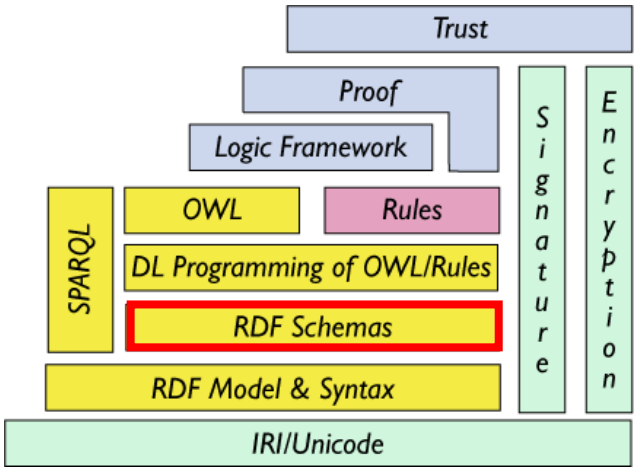
- OWL sur une restriction de RDF/S
  - OWL Lite / DL / Full
  - Logiques de description
  - Vérification, classification, identification
- Définition de classes (énumération, union, intersection, complément, disjonction, restriction valeur et cardinalité des propriétés)
- Caractérisation des propriétés (symétrique, transitive, fonctionnelle, inversement fonctionnelle, inverse)
- Gestion des équivalences, versions, documenter

OWL: Ontologies lourdes 76

RDF Schema pour les ontologies légères

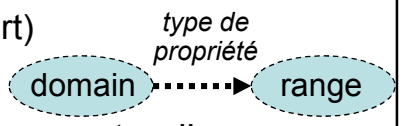
Présentation de la famille de primitives de formalisation proposées par RDFS.

- RDF : modèle de triplets pour annoter des ressources
- RDFS: décrit le vocabulaire (ontologies) utilisé pour ces annotations



W3C, T Berners-Lee, Ivan Herma

- Nommer et définir un vocabulaire conceptuel consensuel et faire des inférences élémentaires
- Nommer les classes de ressources existantes
- Nommer les relations qui existent entre ces classes
- Donner la signature de ces relations:
  - Le domaine (d'où la relation part)
  - Le range (où la relation arrive)

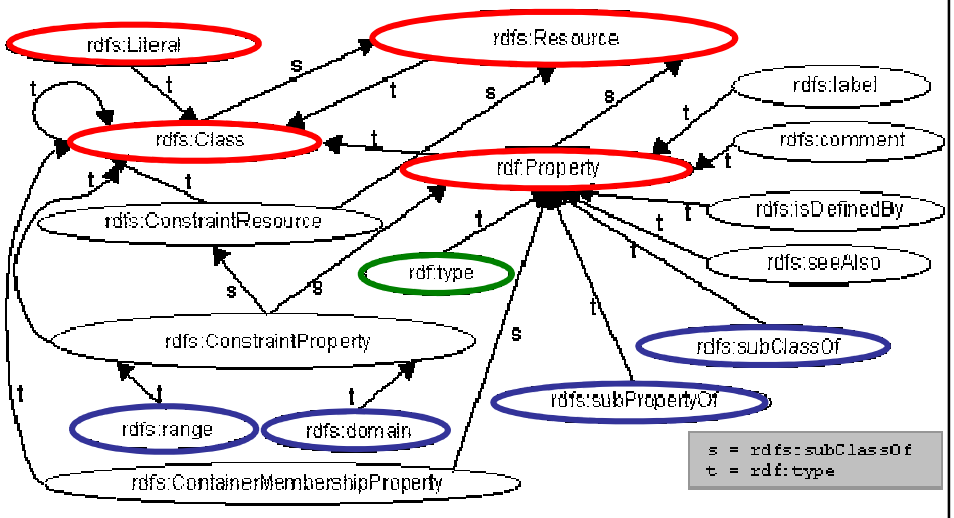


- Documenter ces notions en langue naturelle
- Squelette taxonomique d'une ontologie
  - Liens hiérarchiques des classes
  - Liens hiérarchiques des propriétés

- Tout est ressource.
- Parmi les ressources il y a en particulier…
  - … des classes de ressources qui représentent des types de ressources, des ensembles de ressources;
  - … des propriétés qui représentent des types de relations, des ensembles de relations possibles entre les ressources.
- Parmi les relations il y a en particulier…
  - … la relation de typage / d'instanciation pour dire qu'une ressource/un lien est d'un certain type;
  - … la relation de sous-type (subsomption) pour dire qu'une classe/propriété est sous classe /propriété d'une autre et que ses instances sont aussi instances de l'autre.

- Propriétés : définies en dehors des classes
  - Modèles ouverts permettant à tout le monde de contribuer
  - Pas de raffinement ; pas de surcharge
- Multi-instanciation
  - Le typage multiple d'une même entité
  - Peut être vu comme des facettes
- Héritage multiple classes et propriétés
  - Deux hiérarchies de types: les classes, les propriétés
  - Chaque type peut hériter de zéro, un ou plusieurs types
- Inférences positives ≠ contraintes / vérification
  RDF/S est monotone, conjonctif et positif.

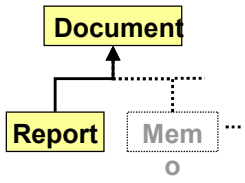
Ressemble à de la POO mais n'en est pas 81



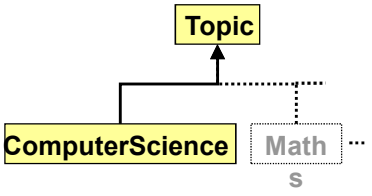
W3C http://www.w3.org/TR/2000/CR-rdf-schema-20000

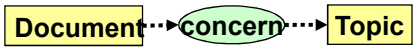
Anciens schémas de RDFS 82

```
class Document

class Report
    subClassOf Document
```

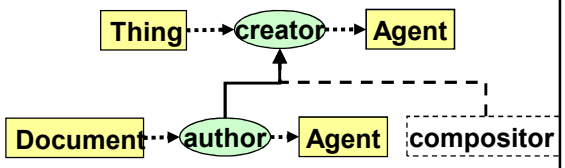



```
class Topic

class ComputerScience
    subClassOf Topic
```






```
property concern
    domain Document
    range  Topic
```




```
property author
    domain Document
    range  Person
    subPropertyOf creator
```

Le rapport RR-5663 a été écrit par le chercheur Moussa Lo et porte sur le sujet des Services Web Sémantiques

```
Report http://www.inria.fr/rrrt/rr-5663.html
  author urn://ugb.sn/mlo
  concern http://www.inria.fr/acacia#Java

Researcher urn://ugb.sn/mlo
  name "Moussa Lo"
```


```
<rdf:RDF xml:base  ="http://inria.fr/2005/humans.rdfs"
 xmlns:rdf ="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
 xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
 xmlns     ="http://www.w3.org/2000/01/rdf-schema#>
 <Class rdf:ID="Man">
  <subClassOf rdf:resource="#Person"/>
  <subClassOf rdf:resource="#Male"/>
   <label xml:lang="en">man</label>
   <comment xml:lang="en">an adult male person</comment>
 </Class>

 <rdf:Property rdf:ID="hasMother">
   <subPropertyOf rdf:resource="#hasParent"/>
   <range rdf:resource="#Female"/>
   <domain rdf:resource="#Human"/>
   <label xml:lang="en">has for mother</label>
   <comment xml:lang="en">to have for parent a
                                        female.</comment>
 </rdf:Property>
```

```
<rdf:RDF xmlns:rdf ="http://www.w3.org/1999/02/22-rdf-
  syntax-ns#"
  xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
  xmlns="http://www.essi.fr/icws/2005-2006/humans.rdfs#"
  xml:base="http://www.essi.fr/icws/2005-2006/humans.rdfs-
  instances" >

  <rdf:Description rdf:ID="Lucas">
   <rdfs:type rdf:resource="http://www.essi.fr/icws/2005-
  2006/humans.rdfs#Man"/>
   <hasMother rdf:resource="#Laura"/>
  </rdf:Description>

  <Man rdf:ID="Lucas">
     <hasMother rdf:resource="#Laura"/>
  </Man>

 <rdf:Description rdf:ID="Lucas">
  <hasMother rdf:resource="#Laura"/>
 </rdf:Description>

  <Man rdf:about="#Lucas" />
```

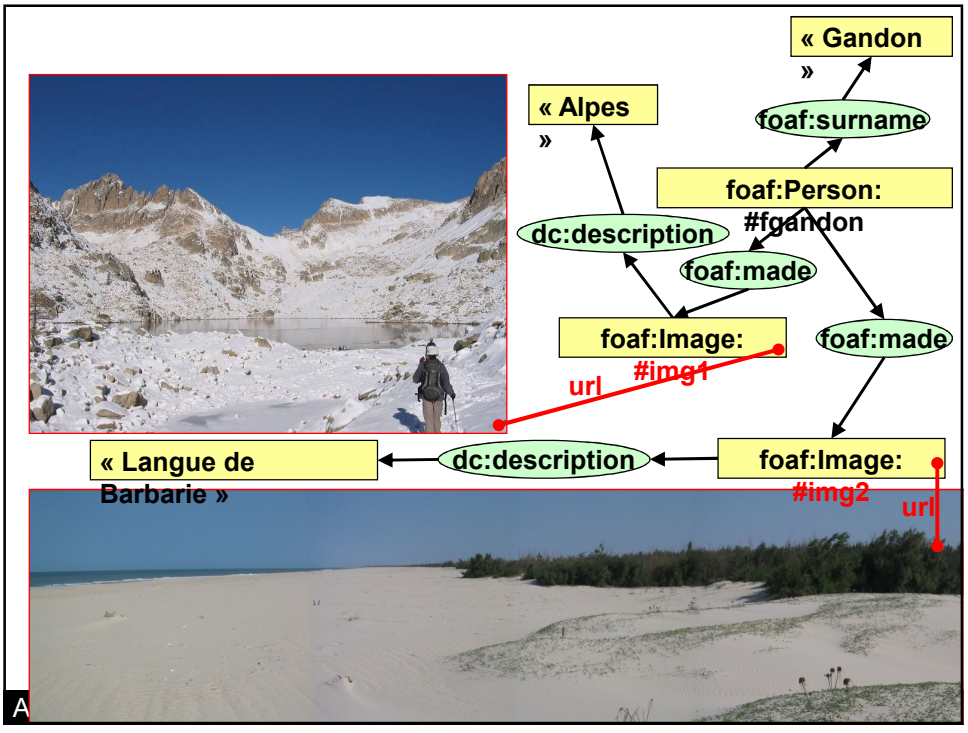
Exemple d'annotation 87

- URI pour les ressources annotées
  - URL de ressources web en ligne
  - URI de ressources abstraites ou physiques
- URI pour les types de ressources
  - URI pour identifier une classe, l'étendre, la spécialiser avec des sous-classes, etc.
  - URI pour typer une ressource
- URI pour les prédicats
  - URI pour identifier un type de propriété, l'étendre, la spécialiser avec des sous-relations
  - URI pour typer les liens entre les ressources

Importance des URIs 88

```
<rdf:RDF   xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
   xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
      xmlns:foaf="http://xmlns.com/foaf/0.1/"
      xmlns:dc="http://purl.org/dc/elements/1.1/">
(…)
   <foaf:Person rdf:nodeID="fabien_gandon">
    <foaf:title>Dr</foaf:title>
    <foaf:firstName>Fabien</foaf:firstName>
    <foaf:surname>Gandon</foaf:surname>
    <foaf:nick>Fab</foaf:nick>
    <foaf:gender>male</foaf:gender>
    <foaf:mbox rdf:resource="mailto:Fabien.Gandon@sophia.inria.fr" />

    <foaf:homepage
         rdf:resource="http://www-
   sop.inria.fr/acacia/personnel/Fabien.Gandon/"/>

    <foaf:img>
      <foaf:Image rdf:about="http://www-sop.inria.fr/acacia/personnel/
                          Fabien.Gandon/resources/images/me_small.gif">
        <dc:title>Fabien Gandon</dc:title>
        <dc:description>Picture of Fabien in 2004</dc:description>
        <dc:format>image/gif</dc:format>
      </foaf:Image>
    </foaf:img>
```

Exemple complet (FOAF et DC) *89*



- Une ressource peut avoir un ou plusieurs (labels) dans une ou plusieurs langues naturelles

```
<rdf:Property rdf:ID='name'>
 <rdfs:domain rdf:resource='Person'/>
 <rdfs:range rdf:resource='&rdfs;Literal'/>
 <rdfs:label xml:lang='fr'>nom</rdfs:label>
 <rdfs:label xml:lang='fr'>nom de famille</rdfs:label>
 <rdfs:label xml:lang='en'>name</rdfs:label>
</rdf:Property>
```

rdfs:label 91

- Les commentaires (comment) sont utilisés pour donner des définitions en langage naturel

```
<rdfs:Class rdf:about='#Woman'>
 <rdfs:subClassOf rdf:resource="#Person" />
 <rdfs:comment xml:lang='fr'>une personne adulte du sexe
  féminin</rdfs:comment>
 <rdfs:comment xml:lang='en'>a female adult
  person</rdfs:comment>
</rdfs:Class>
```

- Renvoi vers des notions connexes

```
<rdfs:Class rdf:about='#Man'>
 <rdfs:seeAlso rdf:resource='#Person'/>
</rdfs:Class>
```

rdfs:comment & rdfs:seeAlso 92

- Si (c_2, subClassOf, c_1) et (x, type, c_2) alors (x, type, c_1)
  - Exemple: (Lo, type, Homme) ⇒ (Lo, type, Humain)
- Si (p_2, subPropertyOf, p_1) et (x, p_2, y) alors (x, p_1, y)
  - Exemple: (Lo, auteur, Note) ⇒ (Lo, créateur, Note)
- Si (c_3, subClassOf, c_2) et (c_2, subClassOf, c_1) alors (c_3, subClassOf, c_1) *Transitivité*
- Si (p_3, subPropertyOf, p_2) et (p_2, subPropertyOf, p_1) alors (p_3, subPropertyOf, p_1) *Transitivité*
- Idem réflexivité subClassOf et subPropertyOf

- Si (p, range, c) et (x, p, y) alors (y, type, c)
  - Exemple: (aPourMere, range, Femme)
    (Fabien, aPourMere, Josette)
    ⇒ (Josette, type, Femme)

  Si (p, domain, c) et (x, p, y) alors (x, type, c)
- Domain & Range sont optionnels (typage par défaut sur Resource)
- La signature est héritée
- Signature effective = conjonction des signatures héritées et spécifiées
- Règles de la sémantique de RDF/S http://www.w3.org/TR/rdf-mt/

- Un même objet vu sous plusieurs points de vue

```
<Man rdf:about="#John">
    <age>32</age>
    <name>smith</name>
</Man>

<Researcher rdf:about="#John">
    <subject>Math</subject>
    <rdf:type rdf:resource="Lecturer"/>
</Researcher>

<Goalkeeper rdf:about="#John"/>
```

Multi-instanciation 95

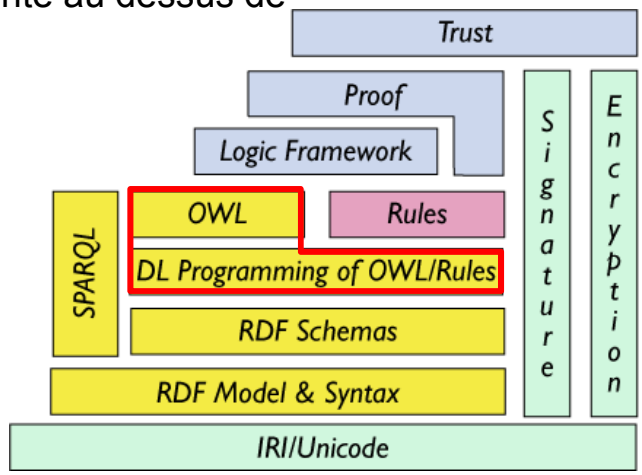
```
c:creator rdfs:domain c:Person
i:Man241 c:creator i:Image262
i:Man241 rdf:type c:Person
c:author rdfs:subPropertyOf c:creator
c:author rdfs:range c:Document
i:Woman297 c:author i:Book812
i:Book812 rdf:type c:Document
i:Woman297 c:creator i:Book812
i:Woman297 rdf:type c:Person
c:Woman rdfs:subClassOf c:Person
c:Document rdfs:subClassOf c:Object
i:Book812 rdf:type c:Object
c:Person rdfs:subClassOf c:Object
i:Man241 rdf:type c:Object
i:Woman297 rdf:type c:Object
c:Woman rdfs:subClassOf c:Object
c:aSoutenu rdfs:domain c:Docteur
c:aSoutenu rdfs:range c:These
i:Woman297 c:aSoutenu i:t127
i:Woman297 rdf:type c:Docteur
i:t127 rdf:type c:These
c:nbDeRoues rdfs:domain c:Vehicule
i:Man241 c:nbDeRoues "4"^^xsd:integer
i:Man241 rdf:type c:Vehicule
```

Question: donnez les inférences faites 96

Ontologies lourdes en OWL

Les trois niveaux d'expressivité supplémentaire

- OWL représenté au dessus de RDFS
  - OWL Lite
  - OWL DL
  - OWL Full



W3C, T Berners-Lee, Ivan Herman

- En réalité OWL est basé sur une restriction de RDF
  Classes ≠ Propriétés ≠ Individus
  Pas de modification du méta model

- Augmenter l'expressivité de la représentation des ontologies de RDFS :
  - Propriétés algébriques des relations
    ex: `ex:estMariéAvec` est symétrique.
  - Correspondances entre deux ontologies
    ex: `ex:Voiture` est équivalent à `ex:Car`
  - Contraintes de cohérence
    ex: `ex:Homme` est disjointe de `ex:Femme`
  - Définition formelles des classes
    ex: `ex:Manager(?x)` équivalent à `?x-(manage)-?y`
  - Restriction des propriétés et raffinement
    ex: pour `ex:Human` le range de `ex:child` est `ex:Human`

- OWL DL signifie OWL Description Logic
- Logiques de description séparation:
  - Concept / Rôle / Individu
  - Deux niveaux distincts:
    niveau terminologique: représentation et manipulation des concepts et des rôles (TBox) subsomption, hiérarchies de concepts et de rôles
    niveau factuel / assertionnel: description et manipulation des individus (ABox)
- Parallèle Concept ↔ Classe & Rôle ↔ Propriété
- Niveaux distincts: d'où la restriction de RDF/S

- Concept primitif (nommé) ou défini (définition formelle)
- Définition : description structurée (équations terminologiques)
- Les définitions utilisent des constructeurs pour donner:
  - les rôles associés au concept
  - les restrictions des rôles (co-domaine, cardinalité) valeurs de base / concepts

[Kayser][Ducourneau, Euzenat, Masini, Napoli]

- le *et / and /* $\cap$ permet de définir une conjonction d'expressions conceptuelles
- Le *non / not /* $\neg$ correspond à la négation et ne porte que sur les concepts primitifs
- la quantification universelle *tout / all /* $\forall$ permet de préciser le co-domaine d'un rôle $\forall$r.C
- la quantification existentielle non typée *some / certains /* $\exists$ permet d'affirmer l'existence d'au moins un couple d'individus ($\exists$r) en relation r

[Ducourneau, Euzenat, Masini, Napoli]

courneau, Euzenat, Masini, Napoli]

primitifs

```
Personne ≤ Top
Ensemble ≤ Top
Homme ≤ Personne
Femme ≤ (and Personne (not Homme))
membre ≤ toprole
chef ≤ membre
```

incompatibles / disjointes

nécessaire

définis

```
Equipe = (and Ensemble
              (all membre Personne)
              (atleast 2 membre))
```

nécessaire & suffisant → classification

```
Petite-équipe = (and Equipe
                     (atmost 5 membre))

Equipe-moderne = (and Equipe
                      (atmost 4 membre)
                      (atleast 1 chef)
                      (all chef femme))
```


- niveau factuel: [Ducourneau, Euzenat, Masini, Napoli]

```
Equipe-moderne(ACACIA)
Homme(OLIVIER)
Personne(ROSE)
membre(ACACIA, FABIEN)
membre(ACACIA, OLIVIER)
membre(ACACIA, ALAIN)
chef(ACACIA,ROSE)
(atmost 4 membre) (ACACIA)
```

- Inférences:
  - ACACIA est une petite équipe
  - FABIEN et ALAIN sont des personnes
    ROSE est une Femme

- Test de subsomption: vérifier qu'un concept en subsume un autre.
  (utile pour valider une classification)
- Classification : placer un concept ou un rôle dans la hiérarchie.
  (assistance à la construction et l'évolution des ontologies)
- Test de satisfiabilité: vérifier qu'un concept admet des instances
  (utile pour vérifier la cohérence)
- Identification : retrouver les concepts les plus spécifiques dont un individu est susceptible d'être une instance.
- Beaucoup de travaux sur la complexité algorithmiques // différentes familles de langages → Influence sur OWL

[Ducourneau, Euzenat, Masini, Napoli]

RDF Schema Features:

- Class (Thing, Nothing)
- rdfs:subClassOf
- rdf:Property
- rdfs:subPropertyOf
- rdfs:domain
- rdfs:range
- Individual

(In)Equality:

- equivalentClass
- equivalentProperty
- sameAs
- differentFrom
- AllDifferent
- distinctMembers

Property Characteristics:

- ObjectProperty
- DatatypeProperty
- inverseOf
- TransitiveProperty
- SymmetricProperty
- FunctionalProperty
- InverseFunctionalProperty

Property Restrictions:

- Restriction
- onProperty
- allValuesFrom
- someValuesFrom

Restricted Cardinality:

- minCardinality (only 0 or 1)
- maxCardinality (only 0 or 1)
- cardinality (only 0 or 1)

Header Information:

- Ontology
- imports

Class Intersection:

- intersectionOf

Datatypes

- xsd datatypes

Versioning:

- versionInfo
- priorVersion
- backwardCompatibleWith
- incompatibleWith
- DeprecatedClass
- DeprecatedProperty

Annotation Properties:

- rdfs:label
- rdfs:comment
- rdfs:seeAlso
- rdfs:isDefinedBy
- AnnotationProperty
- OntologyProperty

Class Axioms:

- *oneOf, dataRange*
- *disjointWith*
- *equivalentClass* (applied to class expressions)
- *rdfs:subClassOf* (applied to class expressions)

Boolean Combinations of Class Expressions:

- *unionOf*
- *complementOf*
- *intersectionOf*

Arbitrary Cardinality:

- *minCardinality*
- *maxCardinality*
- *cardinality*

Filler Information:

- *hasValue*

- Définition en extension d'une classe i.e. en énumérant tous ses membres (utile en particulier pour les domaines d'attributs)

```
<owl:Class rdf:id="CouleurYeux">
    <owl:oneOf rdf:parseType="Collection">
      <owl:Thing rdf:ID="Bleu"/>
      <owl:Thing rdf:ID="Vert"/>
      <owl:Thing rdf:ID="Marron"/>
    </owl:oneOf>
</owl:Class>
```

- Définition d'une classe par union de classes (utile pour les ranges par exemple)

```
<owl:Class>
   <owl:unionOf rdf:parseType="Collection">
       <owl:Class rdf:about="#Person"/>
       <owl:Class rdf:about="#Group"/>
  </owl:unionOf>
</owl:Class>
```

- Définition complète d'une classe par intersection d'autres classes (équivalence)

```
<owl:Class rdf:ID="Man">
  <owl:intersectionOf rdf:parseType="Collection">
   <owl:Class rdf:about="#Male"/>
   <owl:Class rdf:about="#Person"/>
  </owl:intersectionOf>
</owl:Class>
```


- Définition d'une classe complémentaire

```
<owl:Class rdf:ID="Male">
   <owl:complementOf rdf:resource="#Female"/>
</owl:Class>
```

- Imposer une disjonction

```
<owl:Class rdf:ID="Carre">
  <owl:disjointWith rdf:resource="#Rond"/>
</owl:Class>
```

- Contraindre toutes les valeurs:

```
<owl:Class rdf:ID="Herbivore">
  <subClassOf rdf:resource="#Animal"/>
  <subClassOf>
   <owl:Restriction>
    <owl:onProperty rdf:resource="#eats" />
    <owl:allValuesFrom rdf:resource="#Plant" />
   </owl:Restriction>
  </subClassOf>
</owl:Class>
```

- Contraindre au moins une valeur:

```
<owl:Class rdf:ID="Sportive">
 <owl:equivalentClass>
  <owl:Restriction>
   <owl:onProperty rdf:resource="#hobby" />
   <owl:someValuesFrom rdf:resource="#Sport" />
  </owl:Restriction>
 </owl:equivalentClass>
</owl:Class>
```

- Imposer une valeur exacte:

```
<owl:Class rdf:ID="Velo">
 <subClassOf>
  <owl:Restriction>
   <owl:onProperty rdf:resource="#nbRoues" />
   <owl:hasValue>2</owl:hasValue>
  </owl:Restriction>
 </subClassOf>
</owl:Class>
```


- Cardinalité d'une propriété: nombres d'instances différentes d'une propriété
  i.e. nombres de fois où une même ressource est utilisée comme point de départ (domain) d'une propriété avec des valeurs différentes
- Contraintes: nb minimum, nb maximum, nb exacte

```
<owl:Class rdf:ID="Person">
 <subClassOf>
  <owl:Restriction>
   <owl:onProperty rdf:resource="#nom" />
   <owl:maxCardinality>1</owl:maxCardinality>
  </owl:Restriction>
 </subClassOf>
</owl:Class>
```

- La super classe de tout : `owl:Thing`
- La classe vide (sans instances) : `owl:Nothing`

- Les `ObjectProperty` sont des relations entre les ressources uniquement. ex: aPourParent(#thomas,#stéphane)
- Les `DatatypeProperty` ont pour valeur un littéral possiblement typé ex:aPourNom(#thomas,"Thomas")
- Les `AnnotationProperty` sont ignorée dans les inférences, uniquement utilisées pour documenter ou pour des extensions hors des inférences DL

- Propriété symétrique, xRy ⇒ yRx, ex:
```
<owl:SymmetricProperty rdf:ID="hasSpouse" />
```
- Propriété transitive, xRy & yRz ⇒ xRz, ex:
```
<owl:TransitiveProperty rdf:ID="hasAncestor" />
```
- Propriété fonctionnelle, xRy & xRz ⇒ y=z, ex:
```
<owl:FunctionalProperty rdf:ID="hasMother" />
```
- Propriété inversement fonctionnelle,
  xRy & zRy ⇒ x=z, ex:
```
<owl:InverseFunctionalProperty rdf:ID="NumSSociale" />
```

Trois types de propriétés *113*

- Classes équivalentes: `owl:equivalentClass`
- Propriétés équivalentes: `owl:equivalentProperty`
- Instances identiques ou différentes: `owl:sameAs`, `owl:differentFrom`
- Deux propriétés inverses, $xR_1y \Leftrightarrow yR_2x$, ex:
```
<rdf:Property rdf:ID="hasChild">
  <owl:inverseOf rdf:resource="#hasParent"/>
</rdf:Property>
```
- Utilité dans la mise en correspondance d'ontologies:
```
<owl:Class rdf:about="&o1;Person">
 <owl:equivalentClass rdf:resource="&o2;Hito"/>
</owl:Class>
```
- Description de l'ontologie:
  `owl:Ontology`, `owl:imports`, `owl:versionInfo`, `owl:priorVersion`, `owl:backwardCompatibleWith`, `owl:incompatibleWith`
- Versions des classes et des propriétés:
  `owl:DeprecatedClass`, `owl:DeprecatedProperty`

Relations d'équivalence & Gestion de l'ontologie *114*

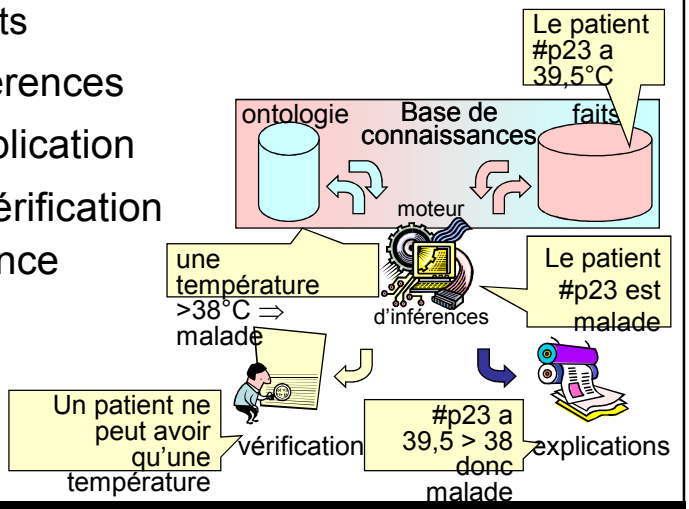
- OWL Full contient tout ce que l'on a mentionné mais OWL Full n'est pas décidable
- OWL DL (Description Logic) est une première restriction qui
  - Sépare: Class, Thing, ObjectProperty, DatatypePropery
  - N'autorise pas: rdfs:Class, extension méta-modèle, cardinalité sur propriété transitive
- OWL Lite est une seconde restriction (i.e. restriction de OWL DL) qui n'autorise pas:
  - Union
  - Cardinalité autre que 0 ou 1

- Nouveau groupe OWL 1.1 depuis Sept. 2007
  - Sucre syntaxique: `DisjointUnion`, `NegativeObjectPropertyAssertion` et `NegativeDataPropertyAssertion`
  - Restriction qualifiée de la cardinalité `ObjectMinCardinality(2 friendOf hacker)`
  - Restriction réflexivité locale `ObjectExistsSelf(likes)`
  - Réflexivité, Irréflexivité, Symétrie, Antisymétrie
  - Propriétés disjointes
  - Propriété impliquée par une chaine de propriétés `SubObjectPropertyOf(SubObjectPropertyChain(owns part) owns)`
  - Utilisation de datatypes personnalisés
  - Meta modélisation par séparation automatique des utilisations comme classes, propriétés ou individus.

Règles sur RDF

Factoriser des connaissances

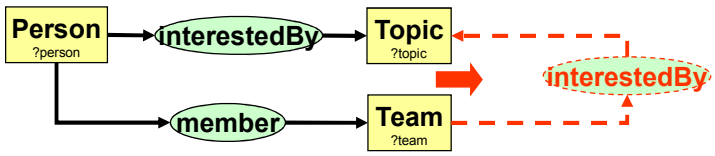
- Base de connaissances
  - ontologie : concepts, propriétés, relations, modèles (causaux, structurels, ...)
  - base de faits
- Moteur d'inférences
- Module d'explication
- Module de vérification de la cohérence



- Une règle de production code une déduction: si ses conditions sont respectées elle produit une nouvelle connaissance
- Une règle est en deux parties:
  - Une prémisse = conditions d'activation de la règle
    ex: "si un français a 18 ans ou plus"
  - Une conclusion = connaissance produit
    ex: "ce français est légalement adulte"
- Chaînage avant:
  - le système essaie d'appliquer toutes les règles
  - application à saturation: tant qu'il y a des déductions
- La prémisse est comme une requête: à chaque réponse trouvée pour la requête/prémisse une conclusion est ajoutée

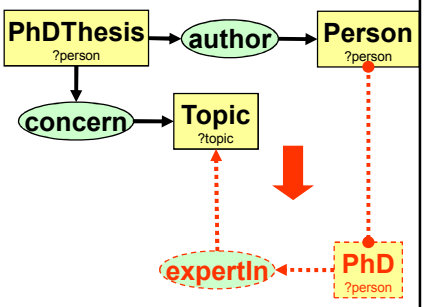
Si un membre d'une équipe a un centre d'intérêt alors l'équipe a aussi ce centre d'intérêt

```
?person interestedBy ?topic
?person member ?team
→
?team interestedBy ?topic
```

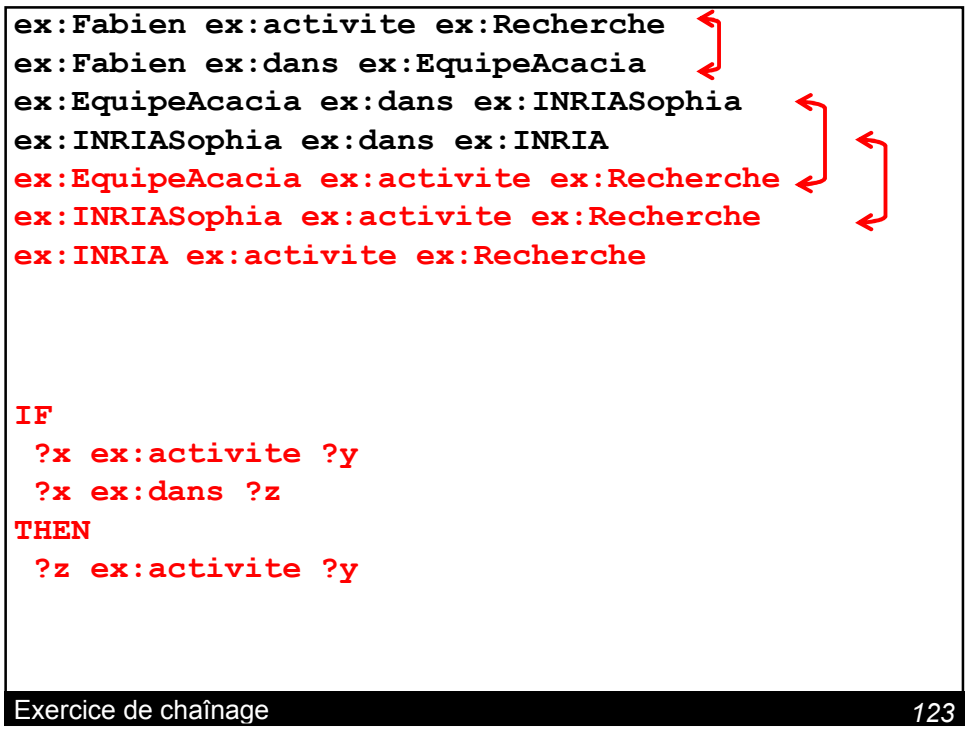


Si une personne a écrit une thèse sur un sujet alors c'est un docteur et un expert du sujet.

```
?person author ?doc
?doc rdf:type PhDThesis
?doc concern ?topic
→
?person expertIn ?topic
?person rdf:type PhD
```




```
<cos:rule>
    <cos:if>
      PREFIX humans: <...>
       {
         ?x rdf:type humans:Male
         ?x rdf:type humans:Person
       }
    </cos:if>

    <cos:then>
       { ?x rdf:type humans:Man }
    </cos:then>
</cos:rule>
```



Exercice de chaînage 123

- Soit un schéma avec les propriétés "parent_de", "enfant_de" comment coder leur équivalence?

```
IF                          IF
 ?x ex:parent_de ?y              ?x ex:enfant_de ?y
THEN                        THEN
 ?y ex:enfant_de ?x              ?y ex:parent_de ?x
```

- Peut-on faire pareil pour "pere_de", "fils_de"?
- Que faut-il changer?

```
IF                            IF
 ?x ex:pere_de ?y              ?x ex:fils_de ?y
 ?y rdf:type ex:Homme          ?y rdf:type ex:Homme
THEN                          THEN
 ?y ex:fils_de ?x              ?y ex:pere_de ?x
```

Exercices (1): inverse 124

- Soit un schéma avec les propriétés "parent_de" et "grand_parent_de" comment définir cette dernière?

```
IF
 ?x ex:parent_de ?y
 ?y ex:parent_de ?z
THEN
 ?x ex:grand_parent_de ?z
```

- Soit un schéma avec la propriétés "ancetre_de" comment coder sans transitivité? (les ancêtres de mes ancêtres sont mes ancêtres aussi)

```
IF
 ?x ex:ancetre_de ?y
 ?y ex:ancetre_de ?z
THEN
 ?x ex:ancetre_de ?z
```


- Soit un schéma avec le type "Objet" et les propriétés "couleur" et "inclus_dans" comment coder le fait que si un objet a une couleur et qu'il inclut un deuxième objet alors ce deuxième objet a la même couleur?

```
IF
 ?x rdf:type ex:Objet
 ?y rdf:type ex:Objet
 ?x ex:couleur ?c
 ?y ex:inclus_dans ?x
THEN
 ?y ex:couleur ?c
```

```
IF
 ?a ex:inclus_dans ?b
 ?b ex:inclus_dans ?c
THEN
 ?a ex:inclus_dans ?c
```

- Comment définir un adulte?

```
IF
 ?x rdf:type ex:Personne
 ?x ex:age ?age
 FILTER ( xsd:integer(?age) > 17 )
THEN
 ?x rdf:type ex:Adulte
```

- Comment dire que la propriété "marie_avec" est symétrique ?

```
IF
 ?x ex:marie_avec ?y
THEN
 ?y ex:marie_avec ?x
```

- Soit un schéma avec les types "Equipe", "EquipeModerne", "Femme" et la propriété "dirige" comment définir qu'une équipe dirigée par une femme est forcément une équipe moderne?

```
IF
 ?x rdf:type ex:Equipe
 ?y ex:dirige ?x
 ?y rdf:type ex:Femme
THEN
 ?x rdf:type ex:EquipeModerne
```

Exercices (4): symmétrie *127*

- Soit le même schéma avec en plus le type "LaboratoireModerne" et la propriété "membre_de" codez qu'une équipe moderne avec au moins 3 membres est un Laboratoire moderne.

```
IF
 ?x rdf:type ex:EquipeModerne
 ?p1 ex:membre_de ?x
 ?p2 ex:membre_de ?x
 ?p3 ex:membre_de ?x
 FILTER ( ?p1 != ?p2 and ?p1 != ?p3 and
          ?p2 != ?p3 )
THEN
 ?x rdf:type ex:LaboratoireModerne
```

Exercices (5): règles imbriquées *128*

- Soit le schéma avec en plus le type "Adulte" et la propriété "marie_avec" codez le fait que deux mariés sont forcément adultes.

```
IF
 ?x ex:marie_avec ?y
THEN
 ?x rdf:type ex:Adulte
 ?y rdf:type ex:Adulte
```


Autres activités dans le web sémantique

- GRDDL (Gleaning Resource Descriptions from Dialects of Languages) - s'intégrer au web actuel

  Gleaning Resource Descriptions from Dialects of Languages (GRDDL) - Rec. 11 Sept. 2007
  GRDDL Test Cases Rec. 11 Sept. 2007
  GRDDL Use Cases: Scenarios of extracting RDF data from XML documents Note 6 April 2007
  GRDDL Primer Note 28 June 2007

  – Déclarer qu'un document contient des données
  – Lier une transformation (en particulier en XSLT) pour extraire ces données (en particulier en RDF/XML)
- XHTML & XML dialectes (ex: spreadsheet)
- Utilisable aussi avec :
  – Microformats ex:

```
<span class="tel">
 <span class="type">home</span>:
 <span
class="value">+1.415.555.1212</span>
```

  – RDFa ex:

```
<h1 property="dc:title">Vacation in the South of France<
```



```
<head profile="http://www.w3.org/2003/g/data-view">
 <title>The man who mistook his wife for a hat</title>
 <link rel="transformation"
  href="http://www.w3.org/2000/06/ dc-extract/dc-extract.xsl" />
 <meta name="DC.Subject" content="clinical tales" />
 …
</head>
```

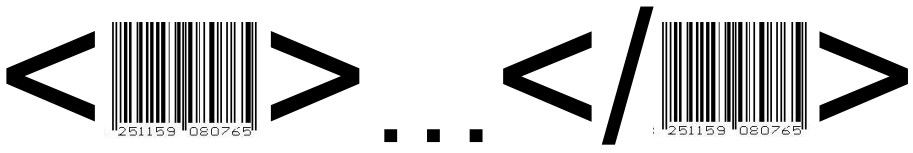
```
# dc:title "The man who mistook his wife for a hat"
# dc:subject "clinical tales"
```

- SW Best Practices and Deployment Working Group:
  - Relations n-aire ex: température de 38 et en hausse
    Defining N-ary Relations on the Semantic Web: Use With Individuals Note 12 April 2006, Noy and Rector (eds.)
  - Les classes comme valeur de propriétés
    Representing Classes As Property Values on the Semantic Web Note 5 April 2005, Noy (ed.)
  - Partitions de valeurs possibles
    Representing Specified Values in OWL: "value partitions" and "value sets" Note 17 May 2005, Rector (ed.)
  - Introduction pour les programmeurs objet
    A Semantic Web Primer for Object-Oriented Software Developers Note 9 March 2006, Knublauch, Oberle, Tetlow, Wallace (eds.)
  - Correspondance Topic Maps - RDF
    A Survey of RDF/Topic Maps Interoperability Proposals Note 10 Feb 2006, Pepper, Vitali, Garshol, Gessa, Presutti (eds.)
  - XML Schema Datatypes in RDF and OWL Note 14 March 2006, Carroll, Pan (eds.)

- Semantic Web Deployment Working Group:
  - Publier un vocabulaire RDF
    Best Practice Recipes for Publishing RDF Vocabularies Working Draft 14 March 2006, Miles, Baker, Swick (eds.)
  - RDFa : intégrer du RDF dans une page web (X)HTML)
    RDFa Primer 1.0; Embedding RDF in XHTML Working Draft 12 March 2007, Adida, Birbeck (eds.) ; RDFa in XHTML: Syntax and Processing Working Draft 18 October 2007, Adida, Birbeck, McCarron, Pemberton (eds.) RDFa Use Cases: Scenarios for Embedding RDF in HTML Working Draft 30 March 2007, Adida, Hausenblas (eds.)
  - Elements et attributs d'annotation en HTML
    Metainformation Module and Metainformation Attributes Module of XHTML 2.0 Working Draft 26 July 2006
  - SKOS : représenter des ressources linguistiques
    SKOS Use Cases and Requirements Working Draft 16 May 2007, Isaac, Phipps, Rubin (eds.) ; SKOS Core Vocabulary Specification Working Draft 2 November 2005, Miles, Brickley (eds.) ; SKOS Core Guide Working Draft 2 November 2005, Miles, Brickley (eds.)

Résumé

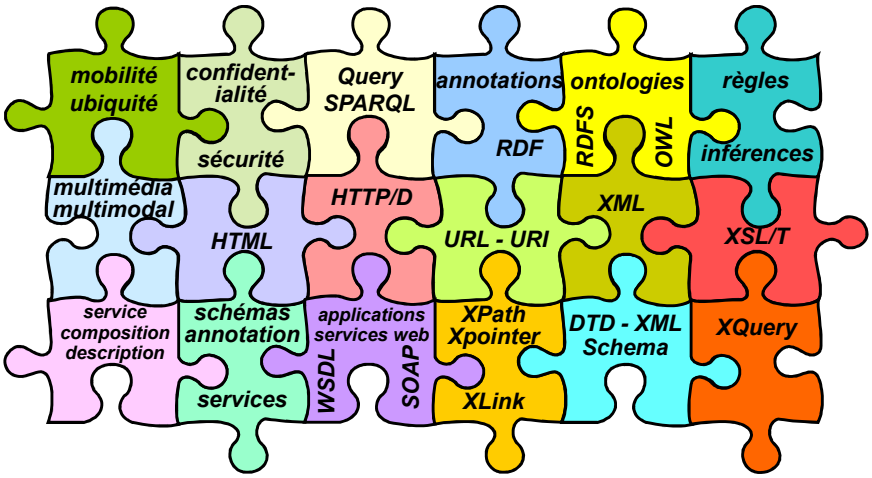
- Intégration de données à l'échelle du Web
  - Web actuel: en langage naturel pour les humains
  - Web sémantique: idem + en langage formel pour les machines; Évolution et non révolution
  - Metadonnée = donnée au dessus des données i.e. des données au dessus du web actuel
- But: interopérabilité, automatisation, réutilisation



- Langages, modèles et formats pour échanger…
  - Structure et nommage: XML, Namespaces, URI
    Roman -> http://essi.fr/ontologie#roman
  - Modèles et ontologies: RDF/S & OWL
    essi:Roman(x) ⇒ essi:Livre(x)
  - Protocoles et requêtes: HTTP, SOAP, SPARQL
  - A venir: règles, web services sémantiques, sécurité, etc.
- Rendre explicite ce qui existe déjà mais est implicite:
  - Capter, ex: types de ressources, auteur, date
  - Exposer ex: structures des formats ex: jpg/mpg, doc/xsl
  - Plein d'outils ont ce potentiel

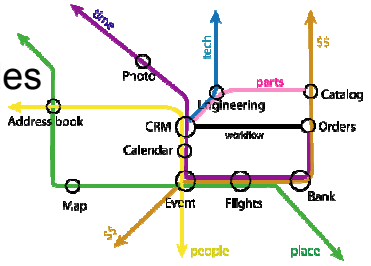
- Compréhension partagée de l'information
  - Entre les personnes
  - Entre les applications
  - Entre les personnes et les applications

Et ça continue...



Another brick in the…

- Le web 3.0 …
  - … un media modifiable (le contenu, les services, les liens, les vues, les interfaces, etc.)
  - … paradigmes de programmation abstraits & déclaratifs,
  - … programmation orientée services et modèles
  - … une immense machine virtuelle
  - … un *Model – View – Controller* à l'échelle du Web
- Chantiers: sécurité, confidentialité, qualité, traçabilité, confiance, *ergonomie*, etc.
- Un web de communautés multiples où chacun a ses rôles, ses données, ses services, etc.
- Un *lieu* de rencontre: présence partage, collaboration,



T. Berners-Lee, RuleML kickoff, 2005

Références

- **RDF :** 6 documents sur http://www.w3.org/RDF
  1. RDF Primer
  2. RDF Concepts and Abstract Data Model
  3. RDF/XML Syntax Specification (Revised)
  4. RDF Schema
  5. RDF Semantics
  6. RDF Test Cases
- **SPARQL:**
  1. SPARQL Query Language for RDF
     http://www.w3.org/TR/rdf-sparql-query/
  2. SPARQL Query Results XML Format
     http://www.w3.org/TR/rdf-sparql-XMLres/
  3. SPARQL Protocol for RDF
     http://www.w3.org/TR/rdf-sparql-protocol/

- W3C documents at http://www.w3.org
  www.w3.org/XML - www.w3.org/RDF - http://www.w3.org/2001/sw/
- W3C Tutorials: http://www.w3.org/2002/03/tutorials
- W3C 10th Anniversary http://www.w3.org/2004/Talks/w3c10-Overview/
- W3School: http://www.w3schools.com/
- Tutorials on Semantic Web Technologies by Ivan Herman
- www.oasis-open.org
- Méthodes et outils pour la gestion des connaissances, R. Dieng et. al. Dunod
- Action Web sémantique CNRS http://www.lalic.paris4.sorbonne.fr/stic/
- Bulletin AFIA avril 2003 http://www.lalic.paris4.sorbonne.fr/stic/articles/
- XML Revolution: http://www.brics.dk/~amoeller/XML/index.html
- O'Reilly XML.com http://www.xml.com/
- Websemantique http://semanticweb.org/