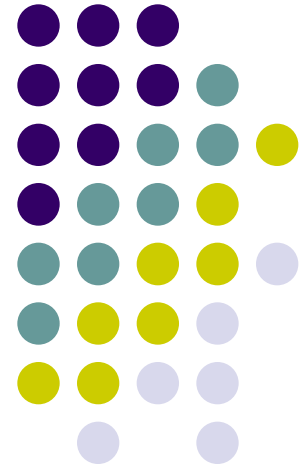
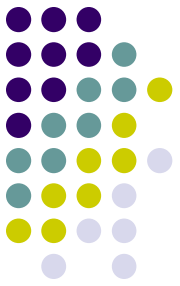


Chapitre III: Langages du Web Sémantique

III.4 OWL (Web Ontologie Language)

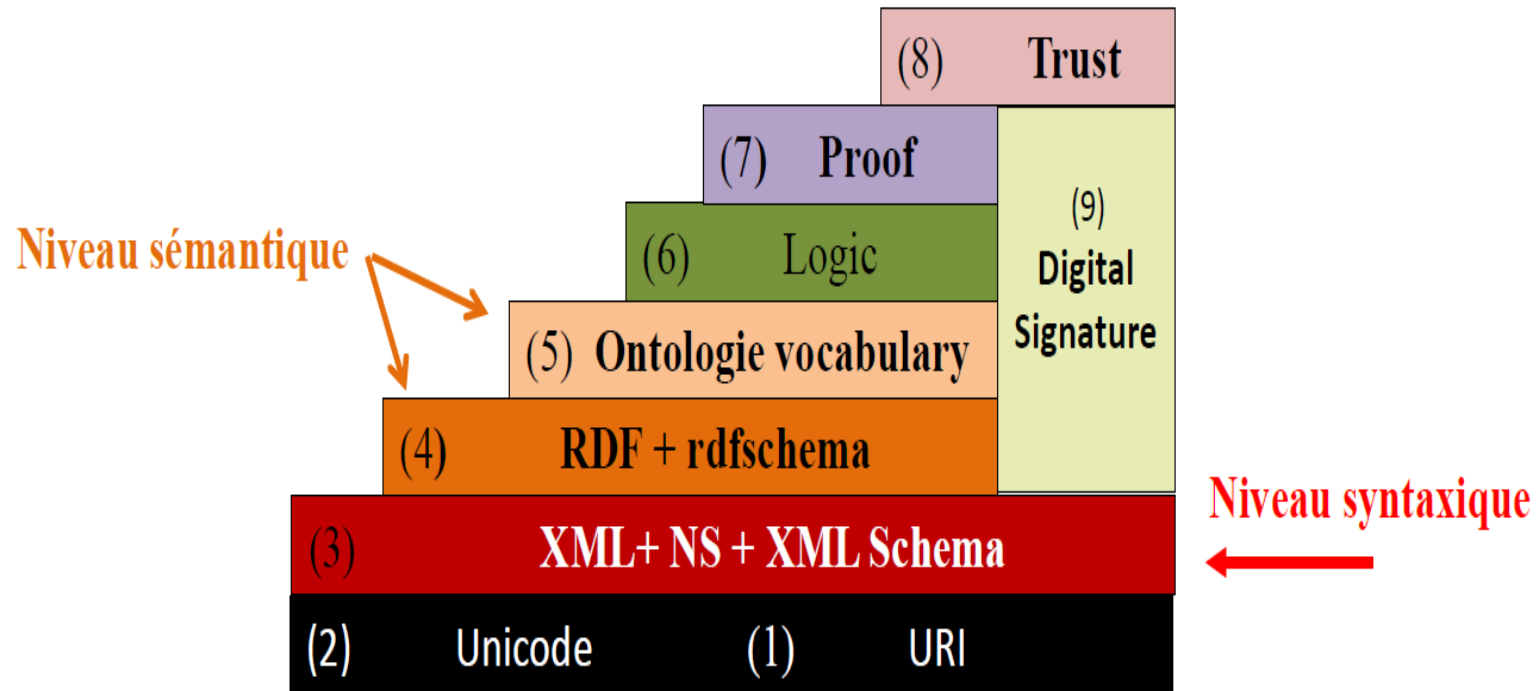
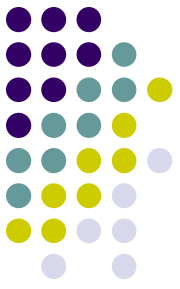




Plan du cours

- Introduction.
- Owl .
- Trois sous langage OWL.
- Structure d'un document OWL.
 - Déclaration d'Onologie.
 - Déclaration de classes.
 - Déclaration de propriétés.
 - Déclaration de faits.
- Exemple.
- Conclusion.

Introduction

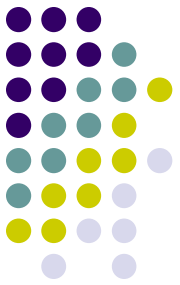




Introduction

- RDF/RDFS permettent de représenter des quelque aspects d'une ontologies(ontologies simples) mais ne permette pas de:
 - combiner des classes :créer de nouvelles classes à partir d'autres classes (union ,intersection,..)
 - Spécifier les cardinalités (une personne doit avoir un seul nom un cours est enseigné par un seul prof)
 - Caractéristiques spéciales pour les propriétés :unicité ,symétrie ,inverse,transitivité ...)
 - Relations entre classes:disjonction ,complément...
 - Autres ...à découvrir dans ce cours.
- **RDF/RDFS est moins expressifs :ontologies légères**

Introduction



- Le langage OWL est conçu pour être un langage d'ontologie (complexes)
- OWL est un langage XML profitant de l'universalité de XML
- Fondé sur la syntaxe RDF/XML.
- Permet de représenter des ontologies complexes muni d'une sémantique formelle

Trois sous langage OWL

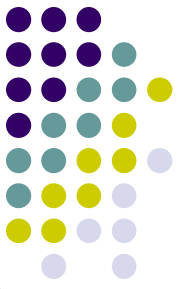
- OWL offre trois sous langages d'expression croissante destinés à des communautés de développeurs et d'utilisateurs spécifiques

1. OWL lite
2. OWL DL
3. OWL Full



- Chaque sous-langage représente une extension de son redresseur :
- Toute ontologie OWL lite est une Ontologie de OWL DL
- Toute ontologie OWL DL est une Ontologie de OWL full

OWL lite



- OWL lite est le sous langage OWL le plus simple.
- Il est destiné aux utilisateurs qui ont besoins d'une hiérarchie de concepts simple.

OWL lite

● Liste des structures OWL



Caractéristiques de RDFs :

[Class \(Thing, Nothing\)](#)

[rdfs:subClassOf](#)

[rdf:Property](#)

[rdfs:subPropertyOf](#)

[rdfs:domain](#)

[rdfs:range](#)

[Individual](#)

(In)égalité :

[equivalentClass](#)

[equivalentProperty](#)

[sameAs](#)

[differentFrom](#)

[AllDifferent](#)

[distinctMembers](#)

Caractéristiques de propriété :

[ObjectProperty](#)

[DatatypeProperty](#)

[inverseOf](#)

[TransitiveProperty](#)

[SymmetricProperty](#)

[FunctionalProperty](#)

[InverseFunctionalProperty](#)

Restrictions de propriété :

[Restriction](#)

[onProperty](#)

[allValuesFrom](#)

[someValuesFrom](#)

Propriétés d'annotation :

[rdfs:label](#)

[rdfs:comment](#)

[rdfs:seeAlso](#)

[rdfs:isDefinedBy](#)

[AnnotationProperty](#)

[OntologyProperty](#)

Cardinalité restreinte :

[minCardinality](#) (seul 0 ou 1)

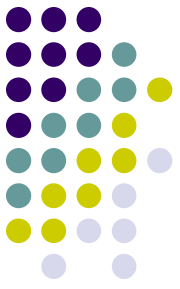
[maxCardinality](#) (seul 0 ou 1)

[cardinality](#) (seul 0 ou 1)

Intersection de classe :

[intersectionOf](#)

OWL DL (Description Logic)



- Plus complexe que OWL lite
- Une expressivité élevée
- Fondé sur la logique de description
- Garantie la complétude (toutes les inférences sont calculables)
- Décidabilité (inférences avec durée finie)

Inférence = générer de nouveaux faits implicites

OWL DL

- Liste des structures OWL DL et OWL Full



Axiomes de classe :

[oneOf, DataRange](#)

[disjointWith](#)

[equivalentClass](#)

[rdfs:subClassOf](#)

Combinaisons booléennes d'expressions de classe :

[unionOf](#)

[complementOf](#)

[intersectionOf](#)

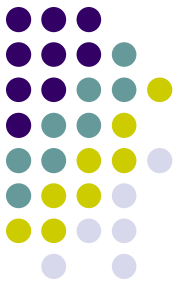
Cardinalité arbitraire :

[minCardinality](#)

[maxCardinality](#)

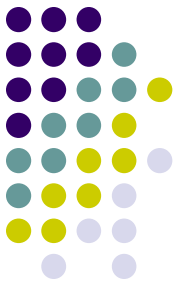
[cardinality](#)

OWL Full



- La version la plus complexe
- Un plus haut niveau d'expressivité
- complétude et la décidabilité des calculs non garanties
- possibilité d'étendre le vocabulaire par défaut de OWL.

Structure d'un document OWL



- Un document OWL possède la syntaxe RDF/XML
- Un document OWL possède la structure suivante:

<rdf:RDF>

[déclaration des espaces de noms]

<owl:ontologie rdf:about="" ">

[déclaration des classes]

[déclaration des propriétés et relations]

[déclaration faits(instances)]

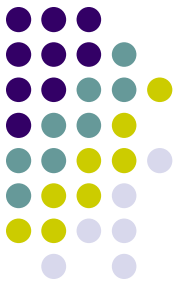
</owl:ontologie>

</rdf:RDF>

Structure Ontologie OWL

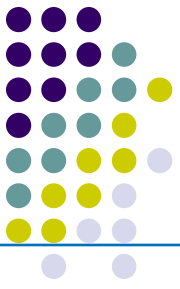
1- espaces de nommage

- Une ontologie commence par l'introduction des espaces de noms utilisés pour définir :
- Le vocabulaire propre à l'ontologie en cours
- Le Vocabulaire des langages (RDF,RDFS,OWL ...)
- Introduire d'autres Ontologies intégrées dans l'ontologie actuelle



Structure Ontologie OWL

1- espaces de nommage



```
<rdf:RDF
```

```
xmlns      ="http://domaine.tld/path/humanite#"
```

```
Xmlns:humanite="http://domaine.tld/path/humanite#"
```

```
Xmlns:base="http://domaine.tld/humanite#"
```

```
Xmlns:vivar="http://domaine.tld/humanite#"
```

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

URI de base de l'ontologie
en cours

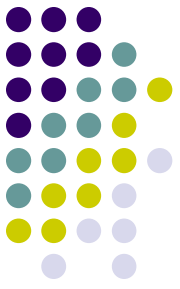
Espace de nommage de URI
de l'ontologie en cours pour
les éléments non qualifiés

Espace de nommage de URI
de l'ontologie en cours

```
</rdf:RDF>
```

Structure Ontologie OWL

1- espaces de nommage



```
<rdf:RDF
xmlns      ="http://domaine.tld"
Xmlns:humanite="http://domaine.tld/humanite#"
Xmlns:base="http://domaine.tld/humanite#"
Xmlns:vivants="http://domaine.tld/humanite/vivants#"
Xmlns:owl="http://www.w3.org/2002/07/owl#"
Xmlns:rdf="http://www.w3.org/2000/01/rdf-schema#"
Xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
Xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
.....
</rdf:RDF>
```

Espace de nommage de URI
de l'ontologie utilisée
(importée)

Espaces de nom des
langages utilisés
:owl,rdf,rdfs,xmlschema

Structure Ontologie OWL

2- entete de l'ontologie

- À la suite de la déclaration des espaces de noms on peut introduire l'entete de l'ontologie grâce à la balise :owl:ontologie

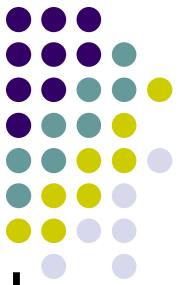
<owl:ontologie rdf:about=".....">

<rdfs:comment>commentaire décrivant l'ontologie
</rdfs:comment>

<rdfs:imports> URIs des ontologies utilisées (une ligne par ontologie **</rdfs:imports>**

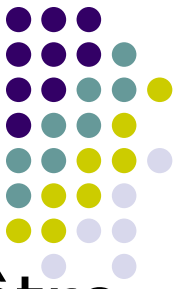
<rdfs:label>etiquete de l'ontologie**</rdfs:label>**

</owl:ontologie>



Structure Ontologie OWL

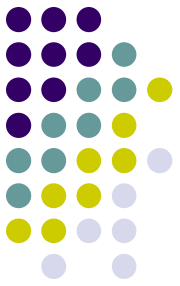
3- déclaration des classes



- La classe **owl:Thing** : est la classes ancêtre de toutes les classes owl.
- La classe **OWL:noThing** : qui est sous-classe de toutes les classes OWL. Cette classe ne peut avoir aucune instance.

Structure Ontologie OWL

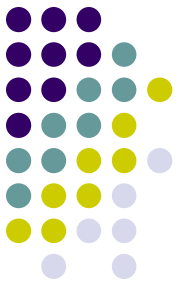
3- déclaration des classes



- Diverse forme de déclaration :
 1. Identification de la classe
 2. Enumération des instances
 3. Restriction de propriétés
 4. Union
 5. Intersection
 6. complément

Structure Ontologie OWL

3- déclaration des classes :identification



- Se fait par le nommage directe de la classe.
- **Syntaxe :**

`<owl:class rdf:ID=[nom de la classe]>`

- ce type de déclaration est le la seule manière pour définir un nom de classe .

- **Exemple : déclarer une classe 'enseignant':**

`<owl:class rdf:ID="enseignant">`

Structure Ontologie OWL

3- déclaration des classes : énumération des instances (DL,FULL)



- Se fait par énumération des instances de la classe par la balise `owl:oneOf` . La classe créée est anonyme

- **Syntaxe :**

`<owl:class>`

`<owl:oneOf rdf:parseType=" collection">`

`<Owl:Thing rdf:about=[URI individu1]>`

`<Owl:Thing rdf:about=[URI individu2]>`

.....

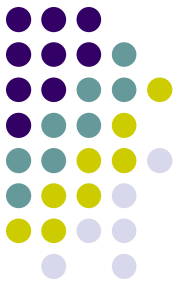
`<Owl:Thing rdf:about=[URI indevidun]>`

`<owl:oneOf`

`<owl:class>`

Structure Ontologie OWL

3- déclaration des classes : énumération des instances (DL,FULL)



- **Exemple:**

```
<owl:class>
```

```
<owl:oneOf rdf:parseType=" collection">
```

```
  <Owl:Thing rdf:about="#JhonSmith">
```

```
  <Owl:Thing rdf:about="#RebeccaJones">
```

```
  <Owl:Thing rdf:about="#AnnaSmith">
```

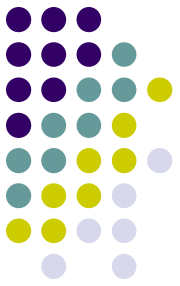
```
  <Owl:Thing rdf:about="#KarlBilly">
```

```
<owl:oneOf
```

```
<owl:class>
```

Structure Ontologie OWL

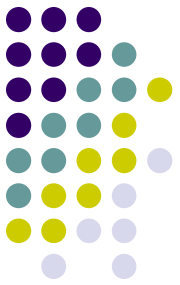
3- déclaration des classes : restriction de propriété



- La description par restriction de propriété permet de définir une classe anonyme composée de toutes les instances de **owl:Thing** qui satisfont une ou plusieurs propriétés.
- Ces contraintes peuvent être de deux types :
 - contrainte de valeur
 - contrainte de cardinalité

Structure Ontologie OWL

3- déclaration des classes :restriction de propriété



- contrainte de valeur

- s'exerce sur la valeur d'une certaine propriété de l'individu (valeur,co-domaine) **exemple:** pour un individu de la classe Humain, sexe = Homme.
- Les deux restrictions **owl:AllValuesFrom** et **owl:SomeValuesFrom** permettent de réaliser la restriction
- **Syntaxe:**

```
<owl:Restriction >
```

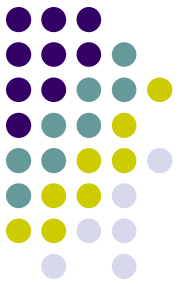
```
<owl:onProperty rdf:resource=[propriété] />
```

```
[contrainte sur la propriété]
```

```
</owl:Restriction >
```

Structure Ontologie OWL

3- déclaration des classes :restriction de propriété



- contrainte de valeur

- exemple1 :les personnes dont tous les enfants sont des médecins

```
<owl:Restriction >
```

```
  <owl:onProperty      rdf:resource="#parentDe/>
```

```
  <owl:AllValuesFrom  rdf:resource="#medecin>
```

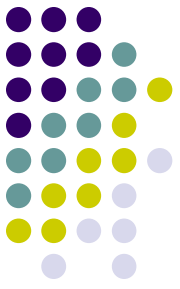
```
</owl:Restriction >
```

- allValuesFrom :Pour tous les personnes, **si** ils ont des enfants, tous ses enfants sont des médecins.

Structure Ontologie OWL

3- déclaration des classes :restriction de propriété

- contrainte de valeur



- exemple2 : les personnes dont au moins un enfant médecin.

```
<owl:Restriction >
```

```
  <owl:onProperty      rdf:resource="#parentDe/>
```

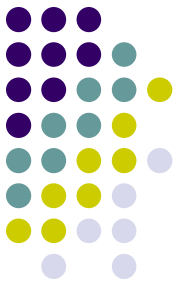
```
  <owl:someValuesFrom  rdf:resource="#medecin>
```

```
</owl:Restriction >
```

- Pour tous les personnes, ils ont au moins un enfant qui est un médecin .

Structure Ontologie OWL

3- déclaration des classes :restriction de propriété



•contrainte de cardinalité

- porte sur le nombre de valeurs que peut prendre une propriété.
- **Exemple:** pour un individu de la classe Humain, parentDe est une propriété qui peut ne pas avoir de valeur, ou avoir plusieurs valeurs, suivant le nombre de fils de l'individu .
- La restriction de cardinalité e fait par
 - owl:minCardinality :
 - owl:maxCardinality
 - owl:Cardinality

Structure Ontologie OWL

3- déclaration des classes : restriction de propriété

- **contrainte de cardinalité** : owl:maxCardinality

- décrit la classe de tous les individu ayant au plus N (N est un entier positif) valeurs distinctes pour la propriété concernée

- **exemple :**

```
<owl:Restriction >
```

```
<owl:onProperty rdf:resource="#parentDe/>
```

```
< owl:maxCardinality  
  rdf:dataType:"xsd:nonNegativeInteger">2
```

```
</owl:maxCardinality>
```

```
</owl:Restriction >
```



Structure Ontologie OWL

3- déclaration des classes :restriction de propriété

- contrainte de cardinalité :`owl:minCardinality`

- décrit la classe de tous les individu ayant au moins N (N et un entier positif) valeurs distinctes pour la propriété concernée

- contrainte de cardinalité :`owl:Cardinality`

- décrit la classe de tous les individu ayant exactement N (N et un entier positif) valeurs distinctes pour la propriété concernée



Structure Ontologie OWL

3- déclaration des classes :intersection



- La balise owl:intersectionOf relie une classe à une liste de descriptions de classe.
- Elle décrit la classe dont l'extension (instances) est le résultat de l'intersection des extensions des classes décrites.

- **Syntaxe**

```
<owl:intersectionOf rdf:parseType="Collection">
```

```
[classe1]
```

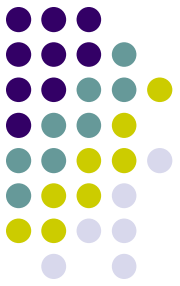
```
[classe2]
```

```
.....
```

```
</owl:intersectionOf>
```

Structure Ontologie OWL

3- déclaration des classes :intersection



```
<owl:Class>
```

```
<owl:intersectionOf rdf:parseType="Collection">
```

```
<owl:Class rdf:about="#etudiantsENST" />
```

```
<owl:Restriction>
```

```
  <owl:onProperty rdf:resource="#aPourFrere" />
```

```
  <owl:cardinality
```

```
    rdf:datatype="&xsd;nonNegativeInteger" >  
    2</owl:cardinality>
```

```
</owl:Restriction >
```

```
</owl:intersectionOf>
```

```
</owl:Class>
```

Structure Ontologie OWL

3- déclaration des classes :union



- La balise **owl:unionOf** relie une classe à une liste de descriptions de classe.
- Elle décrit la classe dont l'extension (instances) est le résultat de l'union des extensions des classes décrites.

- **Syntaxe**

```
< owl:unionOf rdf:parseType="Collection">
```

```
[classe1]
```

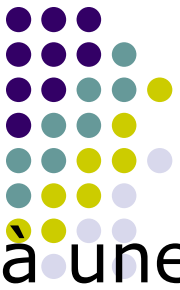
```
[classe2]
```

```
.....
```

```
</ owl:unionOf >
```

Structure Ontologie OWL

3- déclaration des classes :complement



- La balise `owl:complementOf` relie une classe à une seule description de classe.
- Elle décrit la classe dont les instances n'appartenant pas à la classe classes décrite.
- **Syntaxe**

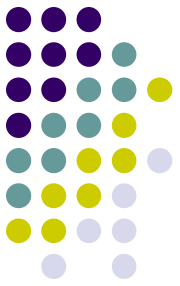
`< owl:complementOf >`

`<owl classe rdf:about [classe2]`

`</ owl:complementOf >`

Structure Ontologie OWL

3- déclaration des classes :complement



- Exemple

```
<owl:Class>
```

```
< owl:complementOf >
```

```
<owl classe rdf:about="#etudiant ">
```

```
</ owl:complementOf >
```

Structure Ontologie OWL

3- déclaration des classes (taxinomie)

- hiérarchie des classes (taxinomie)

- Il existe dans toute ontologie OWL une superclasse, nommée Thing, dont toutes les autres classes sont des sous-classes.
- Ceci nous amène directement au concept d'héritage, disponible à l'aide de la propriété subClassOf :
- **Syntaxe**

`<rdfs:subClassOf rdf:resource=[classe mère]>`



Structure Ontologie OWL

3- déclaration des classes

- hiérarchie des classes (taxinomie)

- Exemple

```
<owl:Class rdf:ID="Humain">
```

```
<rdfs:subClassOf rdf:resource="#EtreVivant" />
```

```
</owl:Class>
```

```
<owl:Class rdf:ID="Homme">
```

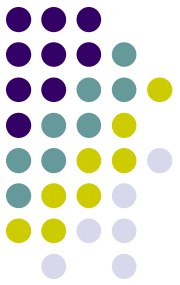
```
<rdfs:subClassOf rdf:resource="#Humain" />
```

```
</owl:Class>
```

```
<owl:Class rdf:ID="Femme">
```

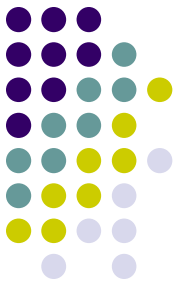
```
<rdfs:subClassOf rdf:resource="#Humain" />
```

```
</owl:Class>
```



Structure Ontologie OWL

3- déclaration des classes



•Classes équivalentes

- Owl:equivalentClass relie une classe à une deuxième classe
- Elle décrit que l'extention de la classe est le meme que la deuxième

● Syntaxe

< Owl:equivalentClass rdf:about:[2emme classe]>

● Exemple

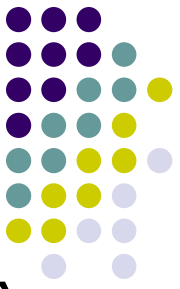
<owl:class rdf:about="#enseignant">

< Owl:equivalentClass rdf:about="#prof"/>

</owl:class>

Structure Ontologie OWL

3- déclaration des classes



•Classes disjointes

- Owl:disjointWith relie une classe à une deuxième classe
- Elle décrit que les extentions des deux classe non auun undividu en commun

- syntaxe

< Owl: Owl:disjointWith rdf:about=[2emme classe]>

- Exemple

```
<owl:class rdf:about="#hommr">
```

```
< Owl:disjointWith rdf:about="#femme"/>
```

```
</owl:class>
```

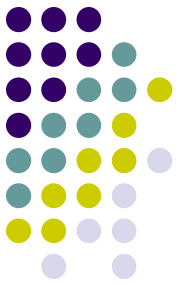
Structure Ontologie OWL

3- déclaration des classes

- Classes disjointes

- Exemple2

```
<owl:Class rdf:ID="Pâte">  
  <rdfs:subClassOf rdf:resource="#NourritureComestible"/>  
  <owl:disjointWith rdf:resource="#Viande"/>  
  <owl:disjointWith rdf:resource="#poulet"/>  
  <owl:disjointWith rdf:resource="#FruitDeMer"/>  
  <owl:disjointWith rdf:resource="#Dessert"/>  
  <owl:disjointWith rdf:resource="#Fruit"/>  
</owl:Class>
```



Structure Ontologie OWL

4- déclaration des propriétés

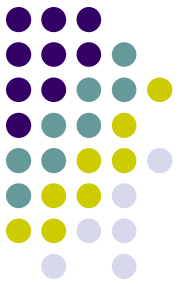


- OWL fait la distinction entre deux types de propriétés :
 1. les propriétés d'objet permettent de relier des instances à d'autres instances (relations)
 2. les propriétés de type de donnée permettent de relier des individus à des valeurs de données.
- Une propriété d'objet est une instance de la classe `owl:ObjectProperty`, une propriété de type de donnée étant une instance de la classe `owl:DatatypeProperty`. Ces deux classes sont elles-mêmes sous-classes de la classe RDF `rdf:Property`.

Structure Ontologie OWL

4- déclaration des propriétés

1.forme minimale



- la déclaration d'une propriété d'objet prend la forme minimale:

<owl:ObjectProperty rdf:ID=[nomPropriété] />

- Exemple :

<owl:ObjectProperty rdf:ID="parentDe">

- la déclaration d'une propriété de type prend la forme minimale:

<owl:DatatypeProperty rdf:ID=[nomPropriété] />

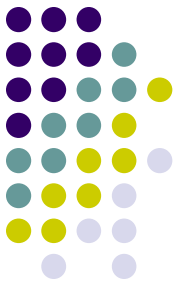
- Exemple :

<owl:DatatypeProperty rdf:ID="nom">

Structure Ontologie OWL

4- déclaration des propriétés

2- domane et co-domaine



- on peut préciser le domaine et l'image de la propriété par les deux balises `rdfs:range` et `rdfs:domain`

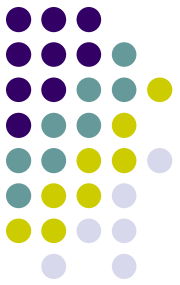
- Exemple 1

```
<owl:ObjectProperty rdf:ID="habite">  
<rdfs:domain rdf:resource="#Humain" />  
<rdfs:range rdf:range="#Pays" />  
</owl:ObjectProperty>
```

Structure Ontologie OWL

4- déclaration des propriétés

2- domaine et co-domaine



- exemple 2

```
<owl:DatatypeProperty rdf:ID="nbrEnfants">  
<rdfs:domain rdf:resource="#Personne" />  
<rdfs:range rdf:resource="&xsd;positiveInteger"/>  
</owl:DatatypeProperty>
```

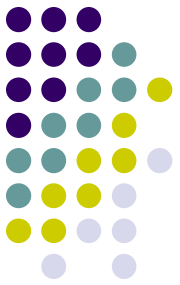
- exemple 3

```
<owl:DatatypeProperty rdf:ID="nom">  
<rdfs:domain rdf:resource="#Personne" />  
<rdfs:range rdf:resource="&xsd;String"/>  
</owl:DatatypeProperty>/
```

Structure Ontologie OWL

4- déclaration des propriétés

3- transitivité



- Si on déclare une propriété P comme transitive :
 $p(a,b)$ et $P(b,c) \rightarrow P(a,c)$

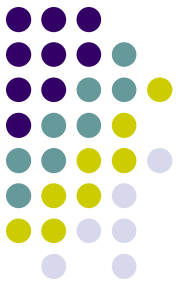
- Exemple

```
<owl:ObjectProperty rdf:ID=« FrereDe»  
  <rdf:type rdf:resource="&owl;TransitiveProperty"/>  
  <rdfs:domain rdf:resource="#Humain" />  
  <rdfs:range rdf:resource="#Humain" />  
</owl:ObjectProperty>
```

Structure Ontologie OWL

4- déclaration des propriétés

4- symetrie



- Si on déclare une propriété P comme symétrique: $p(a,b) \rightarrow P(b,a)$
- On la transitive par le type:TransitivProperty:
`<rdf:type rdf:resource="&owl;SymmetricProperty"/>`

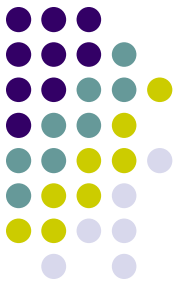
- Exemple

```
<owl:ObjectProperty rdf:ID="FrereDe">  
<rdf:type rdf:resource="&owl;SymmetricProperty" />  
  <rdfs:domain rdf:resource="#Humain" />  
  <rdfs:range rdf:resource="#Humain" />  
</owl:ObjectProperty>
```

Structure Ontologie OWL

4- déclaration des propriétés

5- propriété fonctionnelle



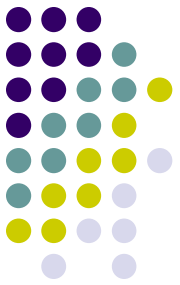
- Une propriété fonctionnelle ne peut avoir qu'une seule valeur par instance
- Une propriété fonctionnelle est déclarée par la balise
- `Owl:functionalProperty` pour les propriétés d'objet que pour les propriétés de type
- Exemple

```
< Owl:functionalProperty  rdf:ID="filsDe">  
<rdfs:domain rdf:resource="#Humain" />  
<rdfs:range  rdf:resource="#Humain" />  
</owl:ObjectProperty
```

Structure Ontologie OWL

4- déclaration des propriétés

6- propriété inverse

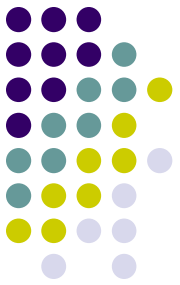


- Permet de déclarer qu'une propriété est l'inverse d'une autre
 - Une propriété P est l'inverse de la propriété P2 :
 - $$\text{Si } P1(x,y) \rightarrow P2(y,x)$$
 - On owl on utilise **owl:inverseOf**
 - Exemple
- ```
< Owl:functionalProperty rdf:ID="filsDe">
<owl:inverseOf rdf:resource="#pereDe" />
</owl:ObjectProperty
```

# Structure Ontologie OWL

## 4- déclaration des propriétés

### 7- sous propriété



- si  $P1$  est une sous-propriété de  $P2$ , alors l'extension de propriété de  $P1$  (un ensemble de couples) devrait être un sous-ensemble de l'extension de propriété  $P2$  (un ensemble de couples aussi)
- On owl on utilise `owl:ObjectProperty`
- Exemple
- `<owl:ObjectProperty rdf:ID="aPourPère">`  
`<rdfs:subPropertyOf rdf:resource="#aPourParent"/>`  
`</owl:ObjectProperty>`

# Structure Ontologie OWL

## 5- déclaration des instances (faits)

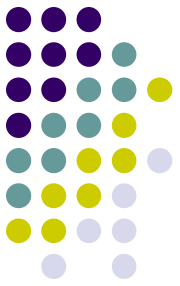


- La définition d'un individu consiste à énoncer un « fait », encore appelé « axiome d'individu ».
- On peut distinguer deux types de faits :
  1. Les faits concernant l'appartenance à une classe
  2. Les faits concernant l'identité des individus



# Structure Ontologie OWL

## 5- déclaration des instances (faits)



- faits d'appartenance

- concerne généralement la déclaration de l'appartenance à une classe d'individu et les valeurs de propriété de cet individu
- Un fait s'exprime de la manière suivante :

**<Homain rdf:ID="Pierre">**

**<nom>pierre</nom>**

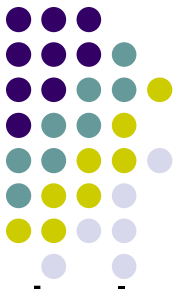
**<aPourPere rdf:resource="#Jacques" />**

**<aPourFrere rdf:resource="#Paul" />**

**</Homain>**

# Structure Ontologie OWL

## 5- déclaration des instances (faits)



- faits d'appartenance

- On peut déclarer un fait (instance) anonyme de la manière suivante :

**<Homain>**

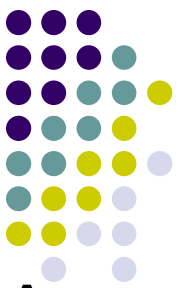
**<nom> pierre </nom>**

**<aPourPere rdf:resource="#Jacques" />**

**<aPourFrere rdf:resource="#Paul" />**

**</Homain>**

# Structure Ontologie OWL



## 5- déclaration des instances (faits)

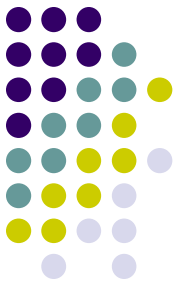
- **les faits d'identité des individus**

- Une difficulté qui peut éventuellement apparaître dans le nommage des individus concerne la non-unicité éventuelle des noms attribués aux individus. Par exemple, un même individu pourrait être désigné de plusieurs façons différentes.
- OWL propose un mécanisme permettant de lever cette ambiguïté, à l'aide des propriétés
- owl:sameAs
- owl:diffrentFrom
- owl:allDifferent.

# Structure Ontologie OWL

## 5- déclaration des instances (faits)

- les faits d'identité des individus



- Exemple

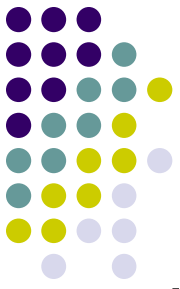
```
<Homain rdf:about="#Algerie">
<owl:sameAs rdf:resource="#paysDeMartyrs" />
</ Homain >
```

- Exemple

```
<Color rdf:about="#red">
<owl:differentFrom rdf:resource="#blue" />
</color>
```

# Structure Ontologie OWL

## 5- déclaration des instances (faits)



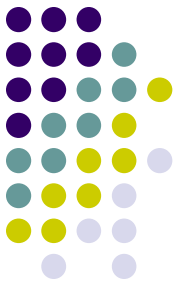
- les faits d'identité des individus

- **owl:differentFrom** conduira probablement à un grand nombre de déclarations, car tous les individus doivent être déclarés disjoints deux-à-deux.
- OWL offre un idiome spécial sous la forme de la structure **owl:AllDifferent**
- **owl:AllDifferent** utilise La propriété **owl:distinctMembers** qui est une structure syntaxique spéciale, ajoutée par commodité.

# Structure Ontologie OWL

## 5- déclaration des instances (faits)

- les faits d'identité des individus



- Exemple

**<owl:AllDifferent>**

```
<owl:distinctMembers rdf:parseType="Collection">
```

```
<Color rdf:about="#Red" />
```

```
<Color rdf:about="#White" />
```

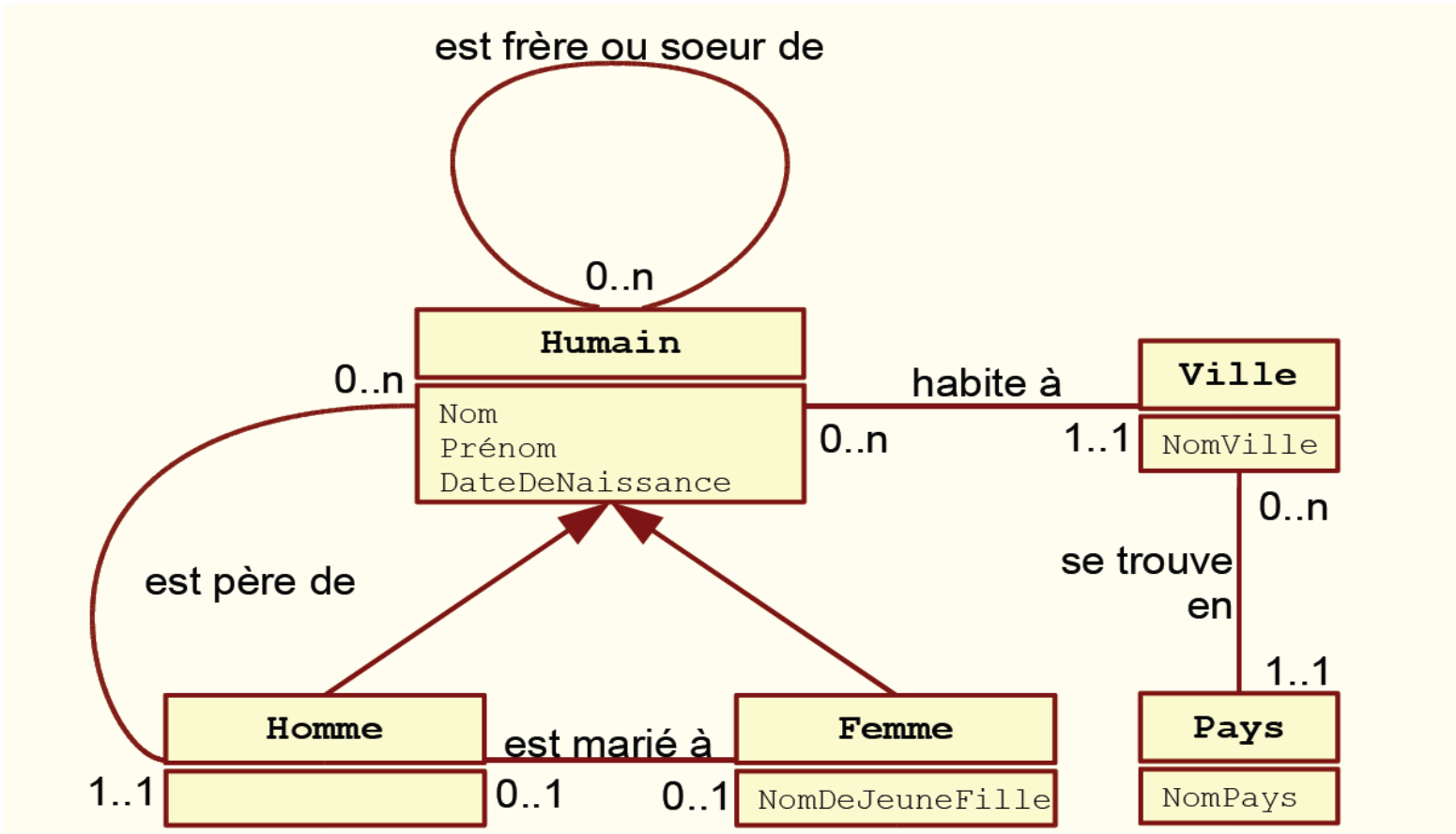
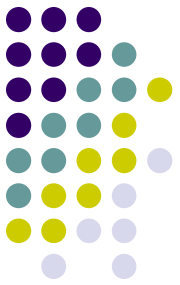
```
<Color rdf:about="#Rose" />
```

```
</owl:distinctMembers>
```

**</owl:AllDifferent>**

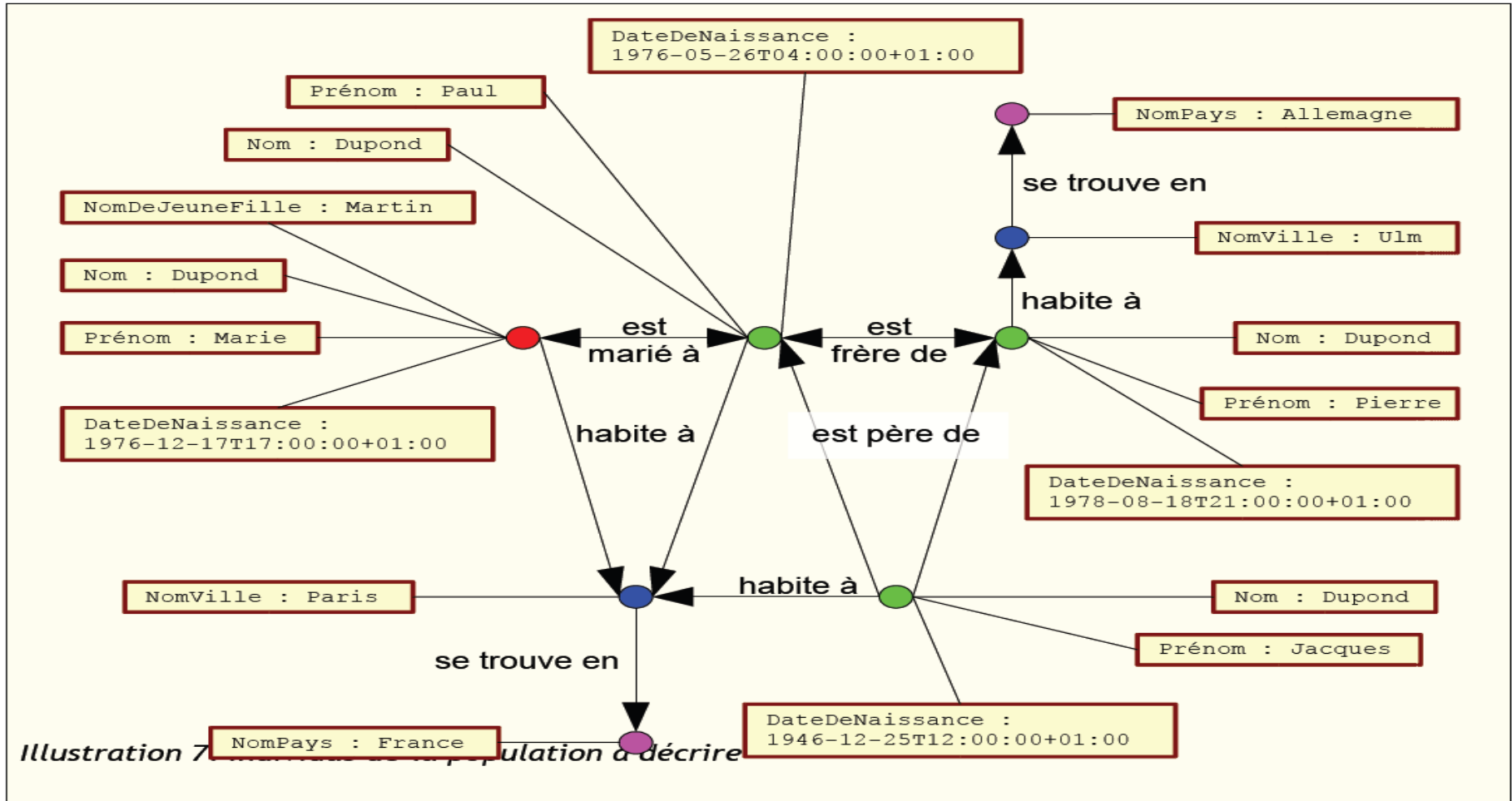
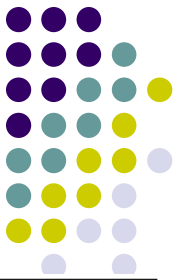
# Exemple écrire une ontologie complete

- Les classes (concepts)



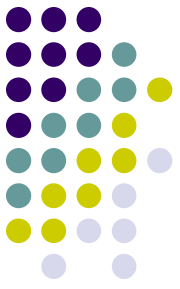
# Exemple écrire une ontologie complete

- Les les fais()les instanes





# conclusion



- Owl est un langage
  - Fondé sur la syntaxe RDF
  - doté d'une sémantique formelle
  - Trois sous types de langage: owl light ,owl dl,owl full
  - très expressif :vacataire plus large pour la description fine des concepts , relations ,instances .
- **OWL est un langage pour décrire tout type d'ontologie**