



WEB SÉMANTIQUE ET ONTOLOGIES  
WEB DES DONNÉES  
DONNÉES LIÉES (LINKED DATA)

## 5 - MODÉLISATION SÉMANTIQUE VOCABULAIRES CONTRÔLÉS & ONTOLOGIES

Philippe GENOUD – Danielle ZIEBELIN - LIG-STEAMER

[Prénom.Nom@imag.fr](mailto:Prénom.Nom@imag.fr)



# Outline

- Introduction
- Distributing Data on the web with RDF
  - Naming the Data : URIs (Uniform Resources Identifiers)
  - The RDF Data model
- Querying Linked Data with SPARQL
- Semantic modelling
  - RDFS
  - OWL
- Conclusion

# RDF limitations

- RDF provides a standard way to express simple statements about resources, using named properties and values...

But you can't express knowledge about the properties and types of the resources

- what are the types allowed for resources ?
- what are the properties allowed for a given type of resource ?
- what are the allowed values for a given property ?
- what are the relations between types of resources (generalization/specialization) ?
- ...

# Resource Description Framework

- RDF a simple model (labelled graph) for data exchange

[http://dbpedia.org/page/Rin\\_Tin\\_Tin](http://dbpedia.org/page/Rin_Tin_Tin)



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

<http://dbpedia.org/class/yago/AnimalActors>



*"On the Internet, nobody knows you're a dog."*

But...

- Which labels for the nodes and edges??
- How to interpret them ?

**Semantic Web** : formally capture some aspects of the meaning of these labels.

Controlled vocabularies

Ontologies

# What is a vocabulary ?

- A person's **vocabulary** is the set of words within a language that are familiar to that person.  
(*Wikipedia*)
- = “all the words known and used by a particular person”  
(*Cambridge Advanced Learners Dictionary*)

On the Semantic Web, **vocabularies** define the **concepts and relationships** (also referred to as “terms”) used to describe and represent an area of concern.

**Vocabularies** are used  
to **classify the terms** that can be used in a particular application,  
**characterize** possible **relationships**,  
and **define** possible **constraints** on using those terms.

In practice, vocabularies can be very complex (with several thousands of terms) or very simple (describing one or two concepts only).

<http://www.w3.org/standards/semanticweb/ontology>

# What is an Ontology ?

- "ontology is the philosophical study of the nature of being, becoming, existence, or reality, as well as the basic categories of being and their relations. Traditionally listed as a part of the major branch of philosophy known as **metaphysics**, ontology deals with questions concerning what entities exist or can be said to exist, and how such entities can be grouped, related within a hierarchy, and subdivided according to similarities and differences."

<http://en.wikipedia.org/wiki/Ontology>

- "In computer science ..., an ontology is a formal framework for representing knowledge. This framework names and defines the types, properties, and interrelationships of the entities in a domain of discourse. The entities are conceptualizations (limited abstractions) of phenomena."

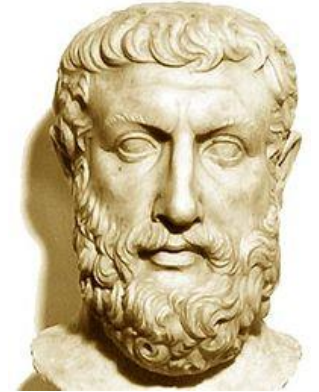
[http://en.wikipedia.org/wiki/Ontology\\_%28information\\_science%29](http://en.wikipedia.org/wiki/Ontology_%28information_science%29)

*An ontology is an explicit specification of a conceptualization. [...] A conceptualization is an abstract, simplified view of the world that we wish to represent for some purpose.*

Thomas R. Gruber, *Towards Principles for the Design of Ontologies Used for Knowledge Sharing in Formal Ontology in Conceptual Analysis and Knowledge Representation*, Kluwer Academic Publishers, 1993

<http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.91.6025&rep=rep1&type=pdf>

<http://tomgruber.org/writing/ontology-definition-2007.htm>



Parmenides  
(c. 515 BCE - c. 460 BCE)



Thomas R. Gruber  
(1959 - )

# The Concept of Ontology in Computer Science

"An ontology is an **explicit, formal specification of a shared conceptualization**. The term is borrowed from philosophy, where an Ontology is a systematic account of Existence. For AI systems, what 'exists' is that which can be represented."

*Thomas R. Gruber: A Translation Approach to Portable Ontology Specifications. Knowledge Acquisition, 5(2):199-220, 1993.*

<b>conceptualization:</b>	abstract model (domain, identified relevant concepts, relations)
<b>explicit:</b>	meaning of all concepts must be defined
<b>formal:</b>	machine understandable
<b>shared:</b>	consensus about ontology

# Vocabulary vs. Ontology

- There is no clear division between what is referred to as “vocabularies” and “ontologies”.

The trend is to use the word “ontology” for more complex, and possibly quite formal collection of terms, whereas “vocabulary” is used when such strict formalism is not necessarily used or only in a very loose sense. Vocabularies are the basic building blocks for inference techniques on the Semantic Web.

<http://www.w3.org/standards/semanticweb/ontology>



# RDF Schema (RDFS)

Officially called “RDF Vocabulary Description Language”  
“Schema” is retained for historical reasons (XMLmania...)  
(not a piece of cake...)

→ provides user communities this ability to define the **vocabularies** (terms) they intend to use in their RDF statements

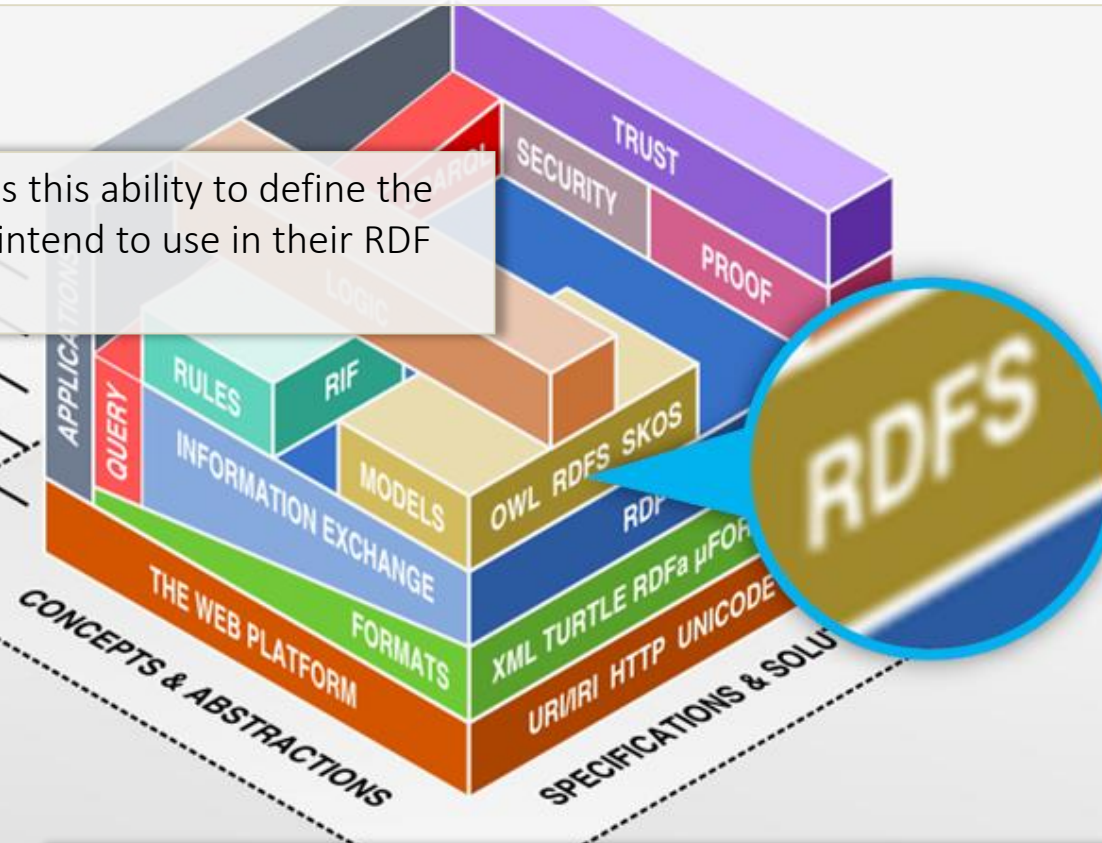
Standardized information exchange is key

Formats are necessary, but not too important

The Semantic Web is based on the Web

Linked Data uses a small selection of technologies

LINKED DATA



→ a **RDF vocabulary** : a specialized set of predefined RDF resources with their own special meanings (semantics)

# RDF Schema

- Introduction
- RDFS Classes
- RDFS Properties
- Interpreting RDFS Schema Declarations
- More RDFS properties
- Conclusion

# RDF Schema - RDFS

- RDFS extends RDF with a *schema vocabulary* (*Resource, Class, Property, subclassOf, subPropertyOf, range, domain ...*) that allows us to define basic vocabulary terms and the relations between them.
  - W3C recommendation RDF Vocabulary Description Language –RDF 1.1 (Feb. 2014)  
<http://www.w3.org/TR/rdf-schema/> (previous rec. RDF 1.0 Feb. 2004)
- A well-defined **semantics** gives “extra meaning” to these particular RDF predicates and resources
  - specifies how terms should be interpreted
  - allows us to draw simple inferences (*entailments*)
- the RDF(S) schema vocabulary is itself provided in the form of an RDF vocabulary
  - resources in the RDF Schema vocabulary have URIs with the prefix  
<http://www.w3.org/2000/01/rdf-schema#> (conventional abbreviation **rdfs:**)

# RDF Schema - RDFS

- Vocabulary descriptions (schemas) written in the RDF(S) language are legal RDF graphs.
  - any software that can process RDF can also interpret a RDF(S) schema as a legal RDF graph
  - ... but must be extended to understand the additional built-in meanings of the RDF Schema terms.

# RDF Schema - RDFS

- RDFS provides a *type system* for RDF
  - similar in some respects to the type systems of object-oriented programming languages such as Java
    - Classes  $\leftrightarrow$  Types
    - Attributes  $\leftrightarrow$  Properties
    - Instances  $\leftrightarrow$  Resources
    - Inheritance  $\leftrightarrow$  Hierarchical organization of types
  - but
    - RDFS is a description language not a programming language : no methods
    - RDFS has a property-centric approach
      - Class based programming languages define a class in terms of the properties its instance may have
      - RDFS describes properties in terms of the classes of resources to which they apply

# RDF Schema

- Introduction
- **RDFS Classes**
- RDFS Properties
- Interpreting RDFS Schema Declarations
- More RDFS properties
- Conclusion

# RDFS - Classes

- **classes** : sets of resources sharing certain characteristics
  - **instances** : members of a class (individuals/objects)
  - **class extension**: the set of instances of the class
  - in RDFS
    - **rdf:type** the property to indicate that a resource is an instance of a class.
    - **rdfs:Class** the class of classes.
- a class is any resource having an **rdf:type** property whose value is the resource **rdfs:Class**.



# RDFS - Classes

## Examples in Turtle

- classes definitions

```
ex:MotorVehicle rdf:type rdfs:Class .
```

```
ex:BlackThing a rdfs:Class .
```

a shortcut for **rdf:type**

- instances definitions

```
exthings:companyCar1 rdf:type ex:MotorVehicle .
```

```
exthings:companyCar2 rdf:type ex:MotorVehicle , ex:BlackThing .
```

```
rdfs:Class rdf:type rdfs:Class.
```

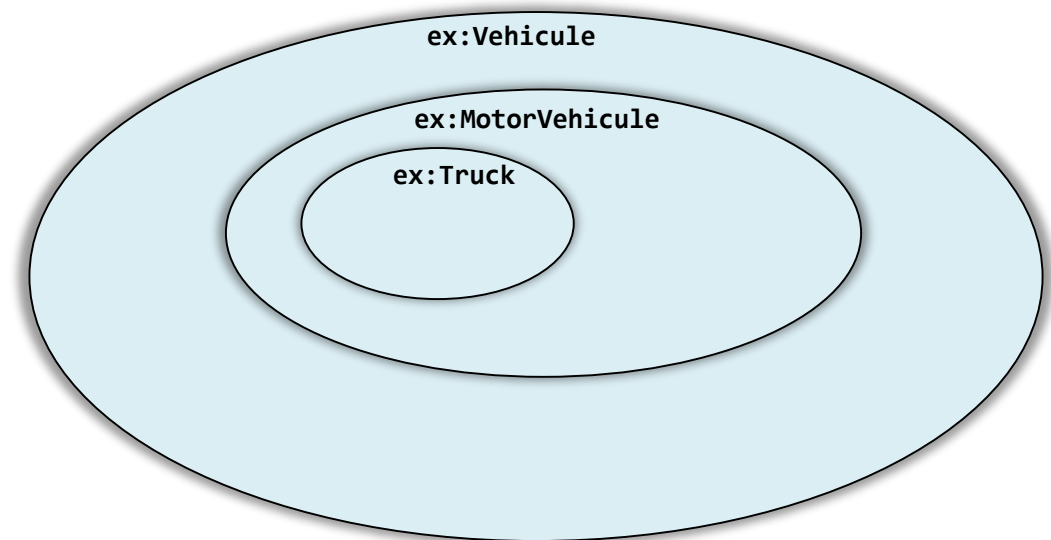
- A resource may be an instance of more than one class.



# RDFSchema: Classes

## Hierarchical relationships

- **class** : set of resources sharing certain characteristics
- **class extension**: the set of resources belonging to the class
- **subclass of class **ClassA****: a class whose extension is a subset of **ClassA** extension
- **superclass of class **ClassA****: a class whose extension is a superset of **ClassA** extension
- subclass and superclass properties defines **hierarchical relationships**
  - subclass : **specialization**
  - superclass : **generalization**
- superclass property is the inverse property of subclass



# RDFS - Classes `rdfs:subClassOf`

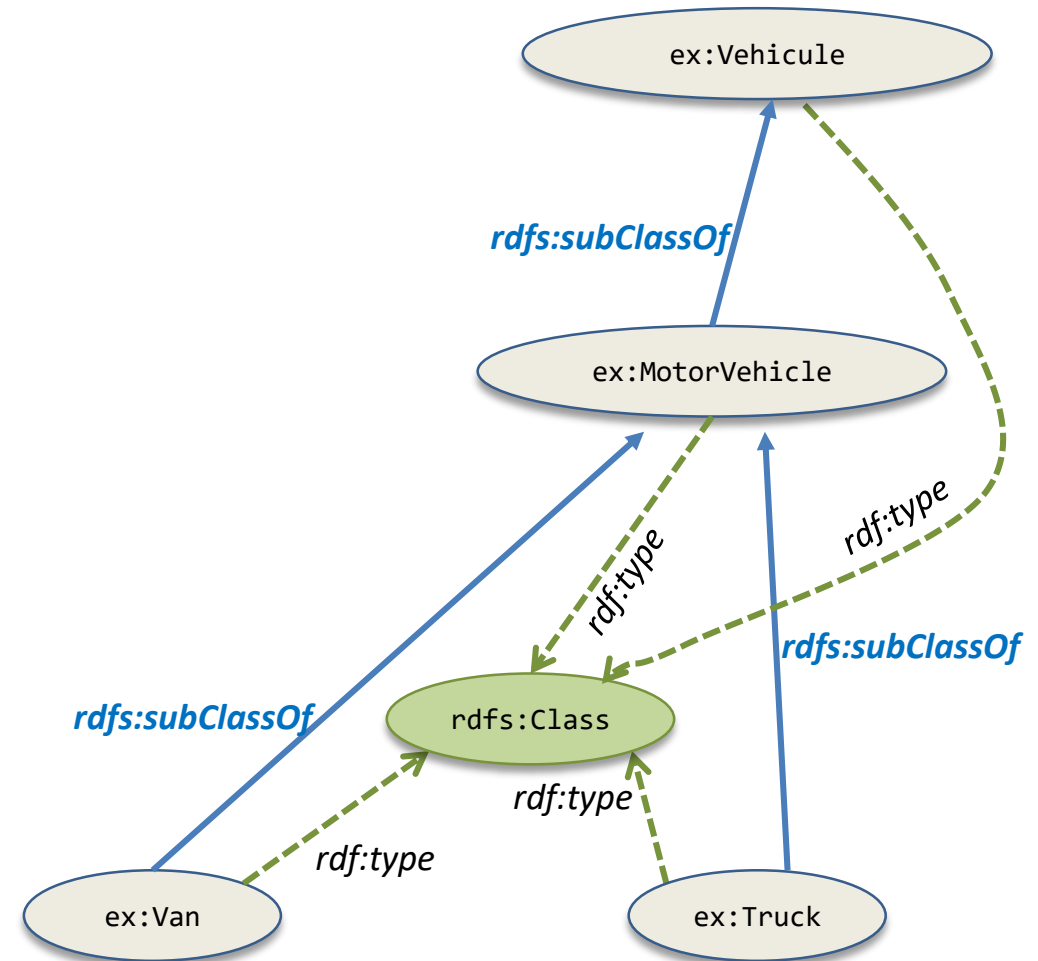
- Specialization relationship between two classes is described using the predefined `rdfs:subClassOf` property

```
ex:Vehicle rdf:type rdfs:Class.
```

```
ex:MotorVehicle rdf:type rdfs:Class;  
  rdfs:subClassOf ex:Vehicle.
```

```
ex:Truck rdf:type rdfs:Class;  
  rdfs:subClassOf ex:MotorVehicle.
```

```
ex:Van rdf:type rdfs:Class;  
  rdfs:subClassOf ex:MotorVehicle.
```



# RDFS - Classes

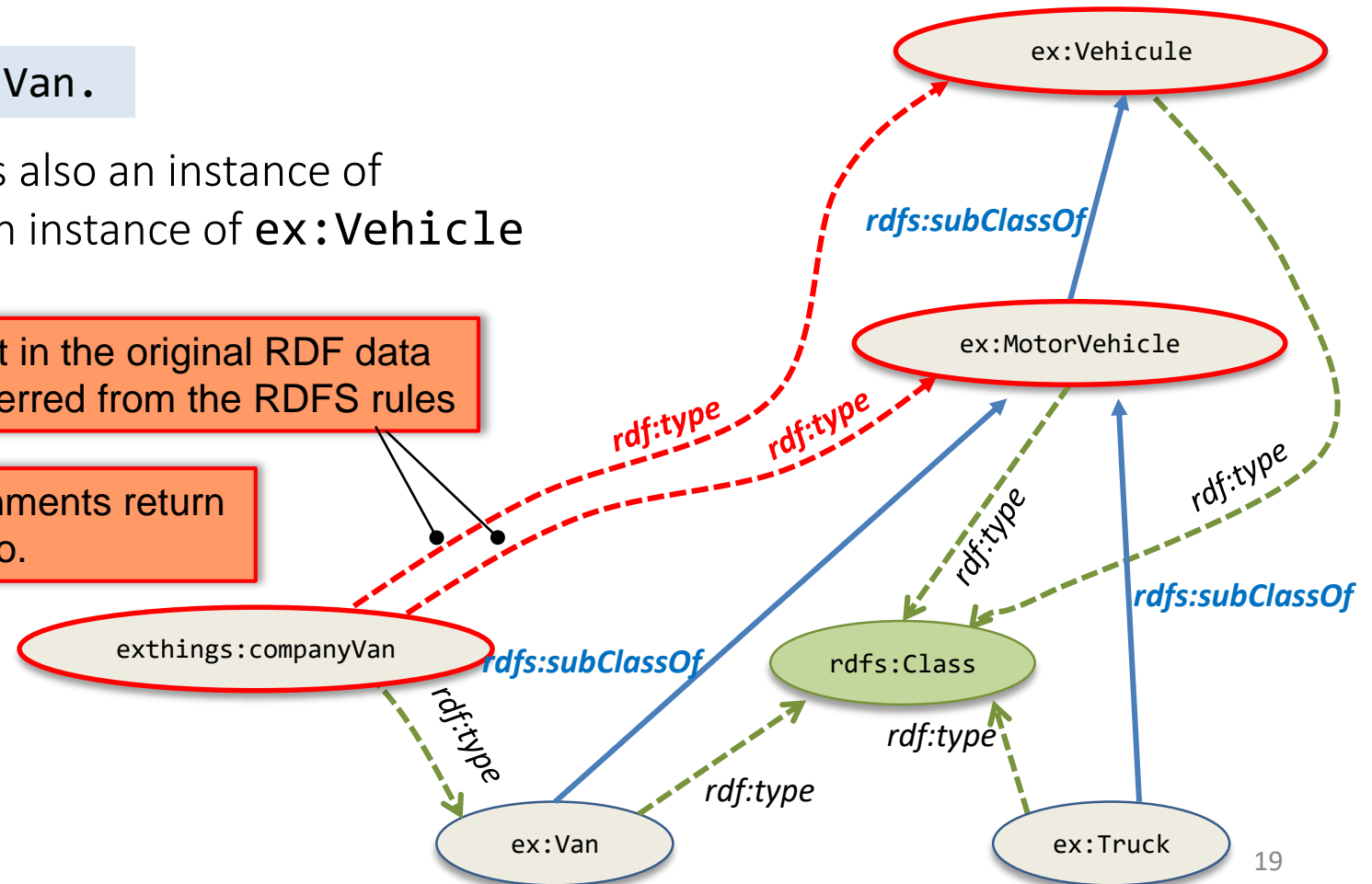
- **subClassOf** semantics (meaning) : any instance of a subclass is also an instance of all the superclasses

```
exthings:companyVan a ex:Van.
```

→ `exthings:companyVan` is also an instance of `ex:MotorVehicle` and an instance of `ex:Vehicle` (transitivity)

Triples are not in the original RDF data but can be inferred from the RDFS rules

RDFS environments return that triples, too.



# RDFS - Inferences

- what magic are these new triples coming from?

YOU KNOW, I DON'T  
THINK MATH IS A SCIENCE.  
I THINK IT'S A RELIGION.  
ALL THESE EQUATIONS  
ARE LIKE MIRACLES YOU  
TAKE TWO NUMBERS AND WHEN  
YOU ADD THEM, THEY MAGICALLY  
BECOME ONE *NEW* NUMBER!  
NO ONE CAN SAY HOW IT  
HAPPENS. YOU EITHER BELIEVE  
IT OR YOU DON'T.



# RDFS - Inferences

- what magic are the inferences coming from?

**TABLE OF CONTENTS**

1. Introduction
2. Conformance
3. Semantic extensions and entailment regimes
4. Notation and terminology
5. Simple Interpretations
  - 5.1 Blank Nodes
  - 5.2 Intuitive summary
6. Simple Entailment
  - 6.1 Properties of simple entailment.
7. Skolemization
8. Literals and datatypes
9. D-interpretations
10. Datatype entailment
  - 10.1 Patterns of datatype entailment
11. RDF Interpretations
12. RDF entailment
  - 12.1 Patterns of RDF entailment
13. RDFS Interpretations
  - 13.1 A note on rdfs:Literal
14. RDFS Entailment
  - 14.1 Patterns of RDFS entailment.
15. RDF Datasets
- A. Entailment rules (Informative)

**RDF 1.1 Semantics**  
W3C Editor's Draft 17 October 2018

**This version:**  
<https://dvcs.w3.org/hg/rdf/raw-file/default/rdf-mt/index.html>

**Latest published version:**  
<https://www.w3.org/TR/rdf11-mt/>

**Latest editor's draft:**  
<https://dvcs.w3.org/hg/rdf/raw-file/default/rdf-mt/index.html>

**Latest Recommendation:**  
<https://www.w3.org/TR/rdf-mt/>

**Editors:**  
Patrick J. Hayes (Florida IHMC)  
Peter F. Patel-Schneider (Nuance Communications)

Copyright © 2004-2018 W3C® (MIT, ERCIM, Keio, Beihang). W3C liability, trademark and permissive document license rules apply.

**Abstract**

This document describes a precise semantics for the Resource Description Framework 1.1 [RDF-PRIMER] and RDF Schema [RDF-SCHEMA]. It defines a number of distinct entailment regimes and corresponding systems of inference rules. It is part of a suite of documents which comprise the full specification of RDF 1.1.

**Status of This Document**

*This section describes the status of this document at the time of its publication. Other documents may supersede this document. A list of current W3C publications and the latest revision of this technical report can be found in the W3C technical reports index at <https://www.w3.org/TR/>.*

This is a revision of the 2004 Semantics specification for RDF [RDF-MT] and supersedes that document.

This document was published by the [RDF Working Group](#) as an Editor's Draft.

Comments regarding this document are welcome. Please send them to [public-rdf-comments@w3.org](mailto:public-rdf-comments@w3.org) ([archives](#)).

Publication as an Editor's Draft does not imply endorsement by the W3C Membership. This is a draft document and may be updated, replaced or obsoleted by other documents at any time. It is inappropriate to cite this document as a standard.

- The RDF 1.1 Semantics document (<https://www.w3.org/TR/rdf11-mt/>) (RDF 1.1 Semantics - W3C Recommendation 25 February 2014) formally defines the meaning of RDF and RDFS vocabularies.
- Formal semantics enables RDFS to draw **valid** and **sound logical** inferences

ex:Van **rdfs:subClassOf** ex:MotorVehicle.  
exthings:companyVan **rdf:type** ex:Van.

ex:Van  $\subseteq$  ex:MotorVehicle      **Model-theoretic semantics**  
exthings:companyVan  $\in$  ex:Van

↓ entailment

exthings:companyVan  $\in$  ex:MotorVehicle

based on the model theoretic semantics new facts (triples) can be deduced from the asserted facts (existing) triples

exthings:companyVan **rdf:type** ex:MotorVehicle.

# RDFS - Inferences

- The RDF 1.1 Semantics document (<https://www.w3.org/TR/rdf11-mt/> RDF 1.1 Semantics - W3C Recommendation 25 February 2014) gives a list of entailment rules (33) that conforms to RDF/RDFS formal semantics
- entailment rules are of the form
  - “if such and such triples are in the graph, add this and this”
  - do that recursively until the graph does not change
- The relevant rule for our example (type propagation):

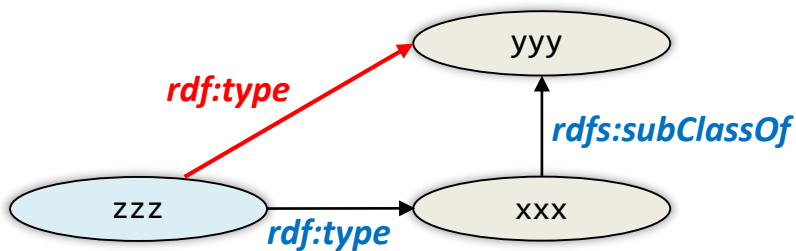
§ 14.1 Patterns of RDFS entailment.

RDFS entailment holds for all the following patterns, which correspond closely to the RDFS semantic conditions:

RDFS entailment patterns.

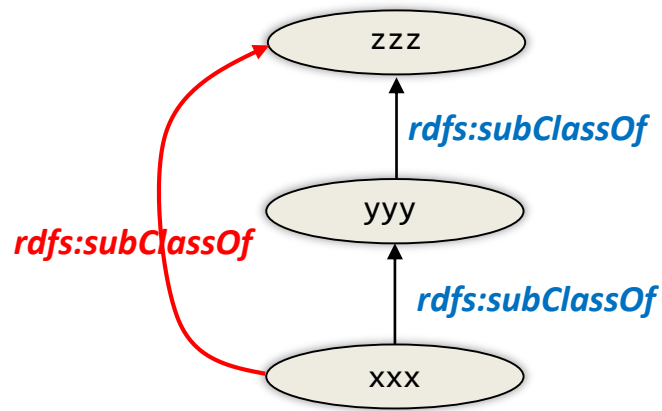
	If S contains:	then S RDFS entails recognizing D:
<i>rdfs:1</i>	any IRI aaa in D	aaa <i>rdf:type</i> <i>rdfs:Datatype</i> .

<i>rdfs:9</i>	<i>xxx rdfs:subClassOf</i> <i>yyy</i> . <i>zzz rdf:type</i> <i>xxx</i> .	<i>zzz rdf:type</i> <i>yyy</i> .
---------------	---	----------------------------------



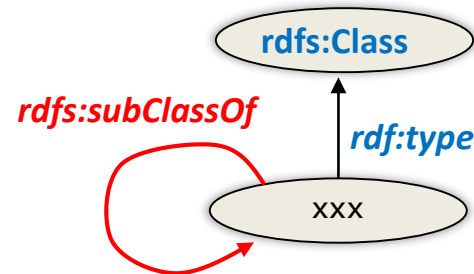
# RDFS: Classes entailments

- Subsumption Transitivity



<b>rdfs11</b>	<code>xxx rdfs:subClassOf yyy .</code> <code>yyy rdfs:subClassOf zzz .</code>	<code>xxx rdfs:subClassOf zzz .</code>
---------------	--	--

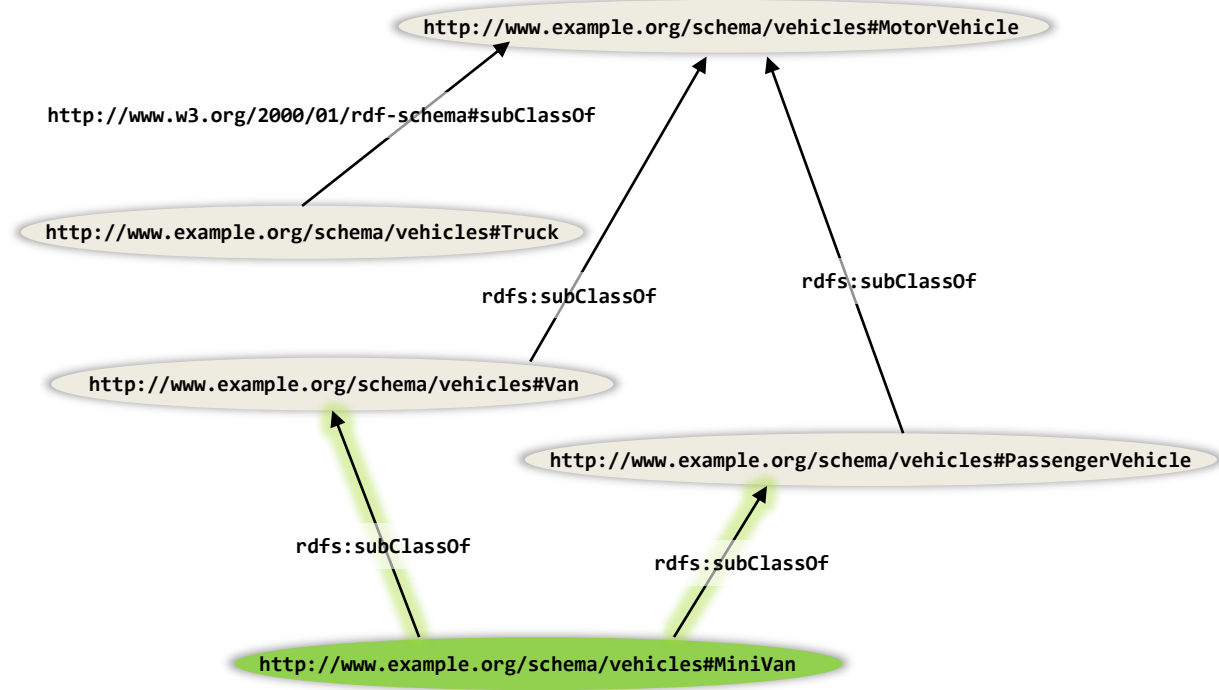
- Subsumption Reflexivity



<b>rdfs10</b>	<code>xxx rdf:type rdfs:Class .</code>	<code>xxx rdfs:subClassOf xxx .</code>
---------------	--	--

# RDFS - Classes

- A class may be a subclass of more than one class

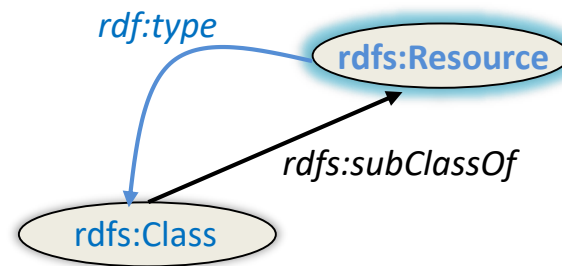


```
ex:MotorVehicle    rdf:type    rdfs:Class .
ex:PassengerVehicle  rdf:type    rdfs:Class ;
                   rdfs:subClassOf  ex:MotorVehicle .
ex:Van              rdf:type    rdfs:Class ,
                   rdfs:subClassOf  ex:MotorVehicle .
ex:Truck             rdf:type    rdfs:Class;
                   rdfs:subClassOf  ex:MotorVehicle .
ex:MiniVan           rdf:type    rdfs:Class ;
                   rdfs:subClassOf  ex:Van, ex:PassengerVehicle .
```

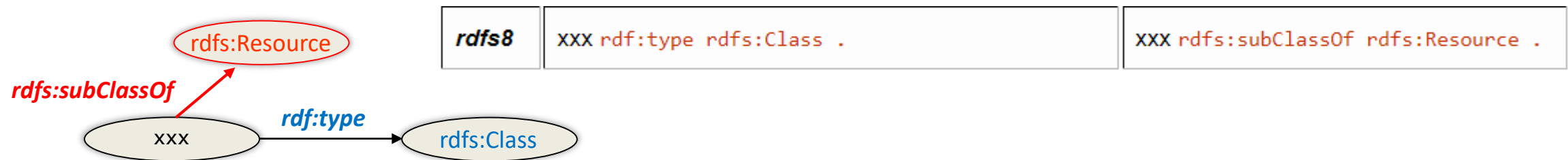


# RDFS - Resource type

- RDFS defines a class named **rdfs:Resource**



- RDFS semantics says that
  - everything is a resource
  - **rdfs:Resource** is a superclass of all classes (aka. **Object** in Java)
  - entailment rule



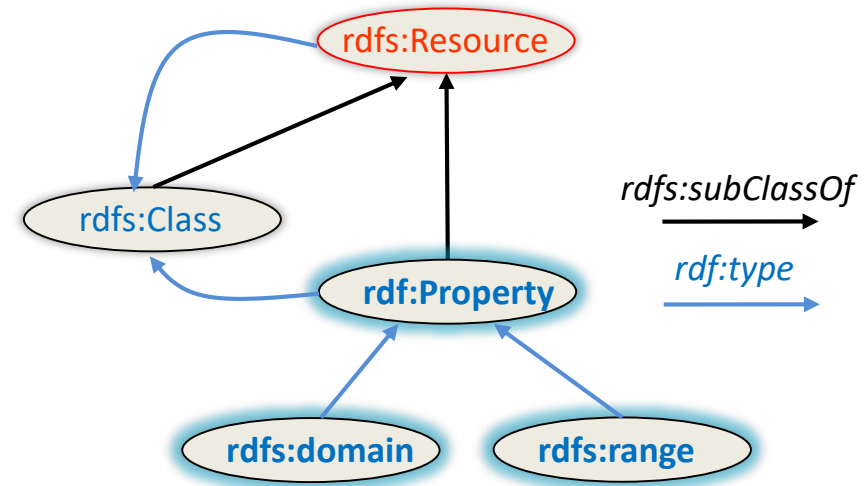
# RDFSchema

- Introduction
- RDFSClasses
- **RDFS Properties**
- Interpreting RDFS Schema Declarations
- More RDFS properties
- Conclusion

# RDFS Properties

- All properties in RDF are described as instances of class **rdf:Property**.

	if S contains	then S RDF entails, recognizing D
rdfD2	xxx aaa yyy .	aaa rdf:type rdf:Property .



- RDFS defines two properties **rdfs:domain**, **rdfs:range** to describe how properties and classes are intended to be used together in RDF data.
  - rdfs:domain** specifies the type of all individuals that are the subject of statements using the described property.
  - rdfs:range** specifies the type of all individuals or the datatype of all literals that are the object of statements using the described property.

# RDFS Properties

- examples of a properties

```
ex:Person    rdf:type    rdfs:Class .
ex:Book     rdf:type    rdfs:Class .
```

```
ex:author    rdf:type    rdf:Property ;
              rdfs:domain ex:Book ;
              rdfs:range  ex:Person .
```

```
ex:price     rdf:type    rdf:Property ;
              rdfs:domain rdfs:Books ;
              rdfs:range  xsd:Integer .
```

```
ex:SemanticWebForDummies ex:author ex:JeffPollock;
                          ex:price  "20"^^xsd:integer;
```

domain is a `rdfs:Class`

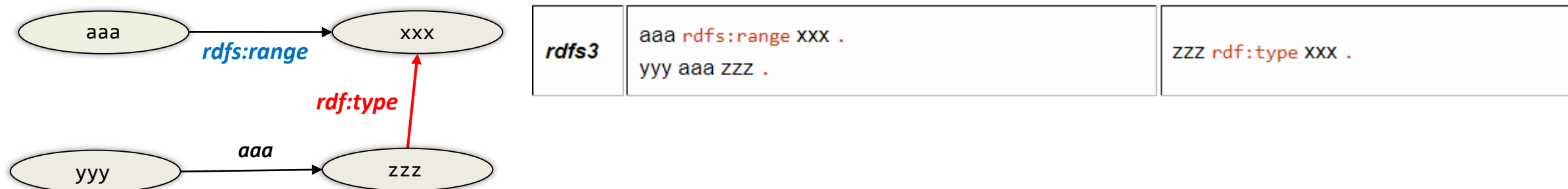
range can be `rdfs:Class`  
or a datatype (`rdfs:DataType`)

# RDFS - Properties

- RDFS semantics allows to deduce new facts from the domain and range properties
  - deduction of entity class membership from the domain of one of its properties



- deduction of entity class membership from the range of one of its properties



# RDFS Properties

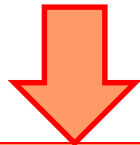
- examples of a inference

```
ex:Person    rdf:type    rdfs:Class .  
ex:Book     rdf:type    rdfs:Class .
```

```
ex:author   rdf:type    rdf:Property ;  
            rdfs:domain ex:Book ;  
            rdfs:range  ex:Person .
```

```
ex:price    rdf:type    rdf:Property ;  
            rdfs:domain rdfs:Books ;  
            rdfs:range  xsd:Integer .
```

```
ex:SemanticWebForDummies ex:author ex:JeffPollock;  
                        ex:price  "20"^^xsd:integer;
```



```
ex:SemanticWebForDummies rdf:type ex:Book.  
ex:JeffPollock          rdf:type ex:Person.
```

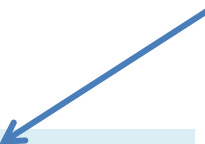
# RDFS Properties

- A property can have zero, one, or more than one range property
  - **0 range property** : nothing is said about the values of the property.  
Anything (resource or literal) can be used as an object of the property
  - **1 range property** : this says that the objects of the property are instances (literals) of the given class (datatype)
  - **more than 1 range property** : this says that the objects of the property are instances (literals) of **all** classes (datatypes) specified as a range

```
ex:hasMother    rdfs:range    ex:Female ;  
                rdfs:range    ex:Person .
```

```
exstaff:frank  ex:hasMother  exstaff:frances .
```

must be both an  
instance of `ex:Female`  
and of `ex:Person`.



# RDF(S) Properties

- A property can have zero, one, or more than one domain property
  - **0 domain property:** nothing is said about the subjects of the property.  
Any resource can be used as a subject of the property
  - **1 domain property :** this says that the subjects of the property are instances of the given class
  - **more than 1 domain property :** this says that the subjects of the property are instances of **all** classes specified as a range

```
ex:isMotherOf    rdfs:domain    ex:Female ;  
                 rdfs:domain    ex:Person .
```

```
exstaff:frances  ex:isMotherOf  exstaff:frank .
```



must be both an instance of `ex:Female` and of `ex:Person`.

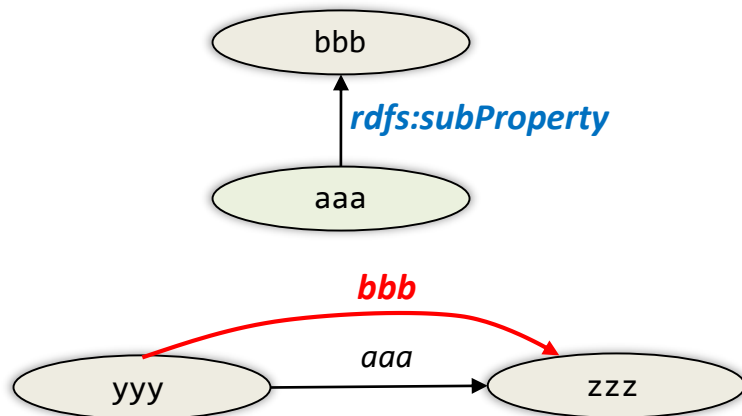


# RDFS Properties

- rdfs defines the **rdfs:subPropertyOf** property
- it allows to describe a specialization relationship between two properties

`property1 rdfs:subPropertyOf property2.`      property1 is a specialization of property2

- semantics : any two resources related using **property1** are implicitly related by **property2**



<i>rdfs7</i>	<code>aaa rdfs:subPropertyOf bbb .</code> <code>xxx aaa yyy .</code>	<code>xxx bbb yyy .</code>
--------------	---	----------------------------

# RDFS Properties

- example of deduction of new facts from subproperties relationships

```
ex:driver      rdf:type      rdf:Property .  
ex:primaryDriver  rdf:type      rdf:Property ;  
               rdfs:subPropertyOf  ex:driver .
```

```
exstaff:fred ex:primaryDriver ex:companyVan .
```



```
exstaff:fred ex:driver ex:companyVan
```

# RDFS Properties

- A property may be a subproperty of zero, one or more properties
- All RDF Schema `rdfs:range` and `rdfs:domain` properties that apply to an RDF property also apply to each of its subproperties.

```
ex:driver          rdf:type          rdf:Property .  
                  rdfs:range        ex:Person;  
                  rdfs:domain       ex:Vehicle.  
ex:primaryDriver  rdf:type          rdf:Property ;  
                  rdfs:subPropertyOf ex:driver .
```

```
exstaff:fred ex:primaryDriver ex:companyVan.
```



```
exstaff:fred ex:driver ex:companyVan.
```

```
exstaff:fred rdf:type ex:Person.
```

```
ex:companyVan rdf:type ex:Vehicle.
```

# RDF(S) Properties

- by default properties are defined independently of classes descriptions

```
ex:hasParent a rdf:Property.
```

```
ex:Human a rdfs:Class.
```

```
ex:John a          ex:Human;  
        ex:hasParent ex:Mary.
```

```
ex:Dog a rdfs:Class.
```

```
ex:Daisy a ex:Dog;  
        ex:hasParent ex:Amber.
```

```
ex:Book a rdfs:Class.
```

```
ex:SemwebForDummies a ex:Book;  
                    ex:hasParent ex:Jeff.
```

**hasParent** has a *global scope* and could then be used to describe instances of any class

this flexibility facilitates the use of property in situations that might not have been anticipated in the original description

but properties can be mis-applied in inappropriate situations

# RDF(S) properties

- with `rdfs:domain` and `rdfs:range` you can "constraint" properties...

```
ex:hasParent a rdf:Property;  
             rdfs:domain ex:Human;  
             rdfs:range ex:Human.
```

```
ex:John a ex:Human;  
        ex:hasParent ex:Mary.
```

- but this introduce inflexibility and should be used with caution

```
ex:Daisy a ex:Dog.
```

```
ex:Mary ex:hasParent ex:Daisy.
```

→ `ex:Daisy` a Dog **and** a Human ?

any range defined for an RDF property applies to all uses of the property

it is not possible in an RDF Schema to define a specific property as having locally-different ranges depending on the class of the resource it is applied to.

# RDF(S) Properties

```
ex:LivingOrganism a rdfs:Class.
```

```
ex:Human a rdfs:Class;  
  rdfs:subClassOf ex:LivingOrganism.
```

```
ex:Dog a rdfs:Class;  
  rdfs:subClassOf ex:LivingOrganism.
```

```
ex:Cat a rdfs:Class;  
  rdfs:subClassOf ex:LivingOrganism.
```

```
ex:hasParent a rdf:Property;  
  rdfs:domain ex:LivingOrganism;  
  rdfs:range ex:LivingOrganism.
```

```
ex:hasHumanParent a rdf:Property;  
  rdfs:subPropertyOf ex:hasParent;  
  rdfs:domain ex:Human;  
  rdfs:range ex:Human.
```

```
ex:hasCatParent a rdf:Property;  
  rdfs:subPropertyOf ex:hasParent;  
  ...
```



Not enough :  
A Dog can have a Cat as parent

Add new properties  
specializing hasParent



# RDF Schema

- Introduction
- RDFS Classes
- RDFS Properties
- Interpreting RDFS Schema Declarations
- More RDFS properties
- Conclusion

# RDFS - Interpreting Schema Declarations

- RDF(S) descriptions are not (necessarily) *prescriptive* like OOP languages are.
- RDF(S) doesn't prescribe how the statements should be used by an application

## OOP Language (Java)

```
public Class Person{  
    private String name;  
  
    Person(String name) {  
        this.name = name;  
    }  
}
```

every instance of Person has a property (attribute) name and this attribute is unique

this attribute is unique

instance of Person have no other property than name

## RDF(S)

```
ex:Person a rdfs:Class.
```

```
ex:hasName a rdf:Property;  
    rfs:domain ex:Person;  
    rdfs:range xsd:String;
```

the property is not mandatory for instances of Person

```
ex:John a ex:Person;  
    ex:hasName "Johnny Defool".
```

```
ex:Jane a rdfs:Person.
```

you can have multiple values

```
ex:John ex:hasName "Johnny".
```

you can have other properties

```
ex:John foaf:knows ex:Mary.
```



# RDFS - Interpreting Schema Declarations

- RDFS might be used in different ways according to applications

```
ex:hasParent a rdf:Property;  
             rdfs:domain ex:Human;  
             rdfs:range  ex:Human.
```

- Constraint

- a Human doesn't have a parent → application error
- a Human has a parent p but its type is unknown → application error
- a Human has a parent p with an other type than Human → application error

- Inferences

- a Human doesn't have a parent → application adds an unknown (Human) parent
- a Human has a parent p but its type is unknown → application infers that p is of Human type
- a Human has a parent p with an other type than Human → application infers that p is of Human type (in addition to other known types)
  - this can lead to inconsistencies that must be detected

# RDF Schema

- Introduction
- RDF(S) Classes
- RDF(S) Properties
- Interpreting RDF(S) Schema Declarations
- **More RDF(S) properties**
- Conclusion

# more RDF(S) properties

- *rdfs:Resource* **rdfs:label** *rdfs:Literal*
  - a human-readable version of a resource's name
- *rdfs:Resource* **rdfs:comment** *rdfs:Literal*
  - used to provide a human-readable description of a resource
- *rdfs:Resource* **rdfs:seeAlso** *rdfs:Resource*
  - used to indicate a resource that might provide additional information about the subject resource
- *rdfs:Resource* **rdfs:isDefinedBy** *rdfs:Resource*
  - to indicate a resource defining the subject resource. This property may be used to indicate an RDF vocabulary in which a resource is described
- ...

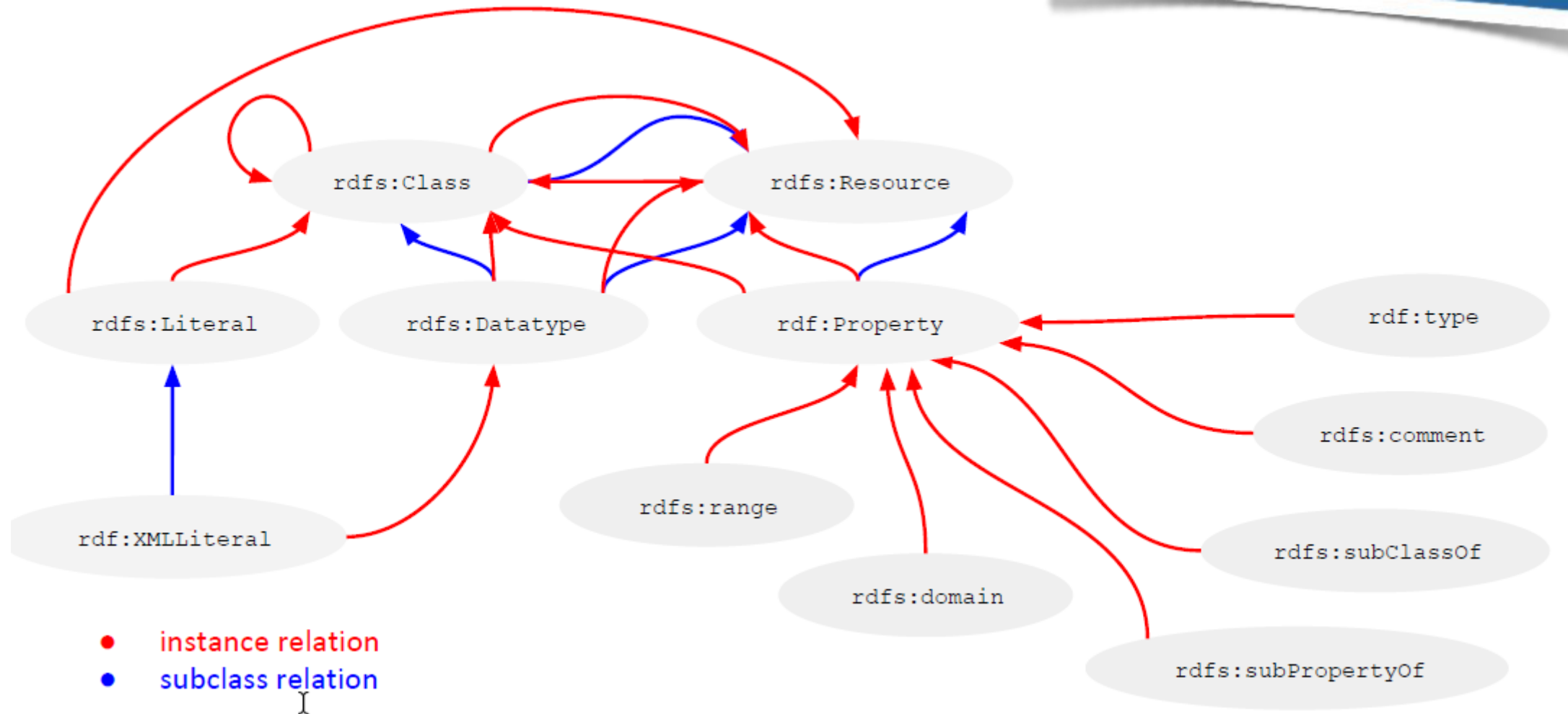
<http://www.w3.org/TR/rdf-schema/>

# RDF Schema (RDF(S))

- Introduction
- RDF(S) Classes
- RDF(S) Properties
- Interpreting RDF(S) Schema Declarations
- more RDF(S) properties
- Conclusion

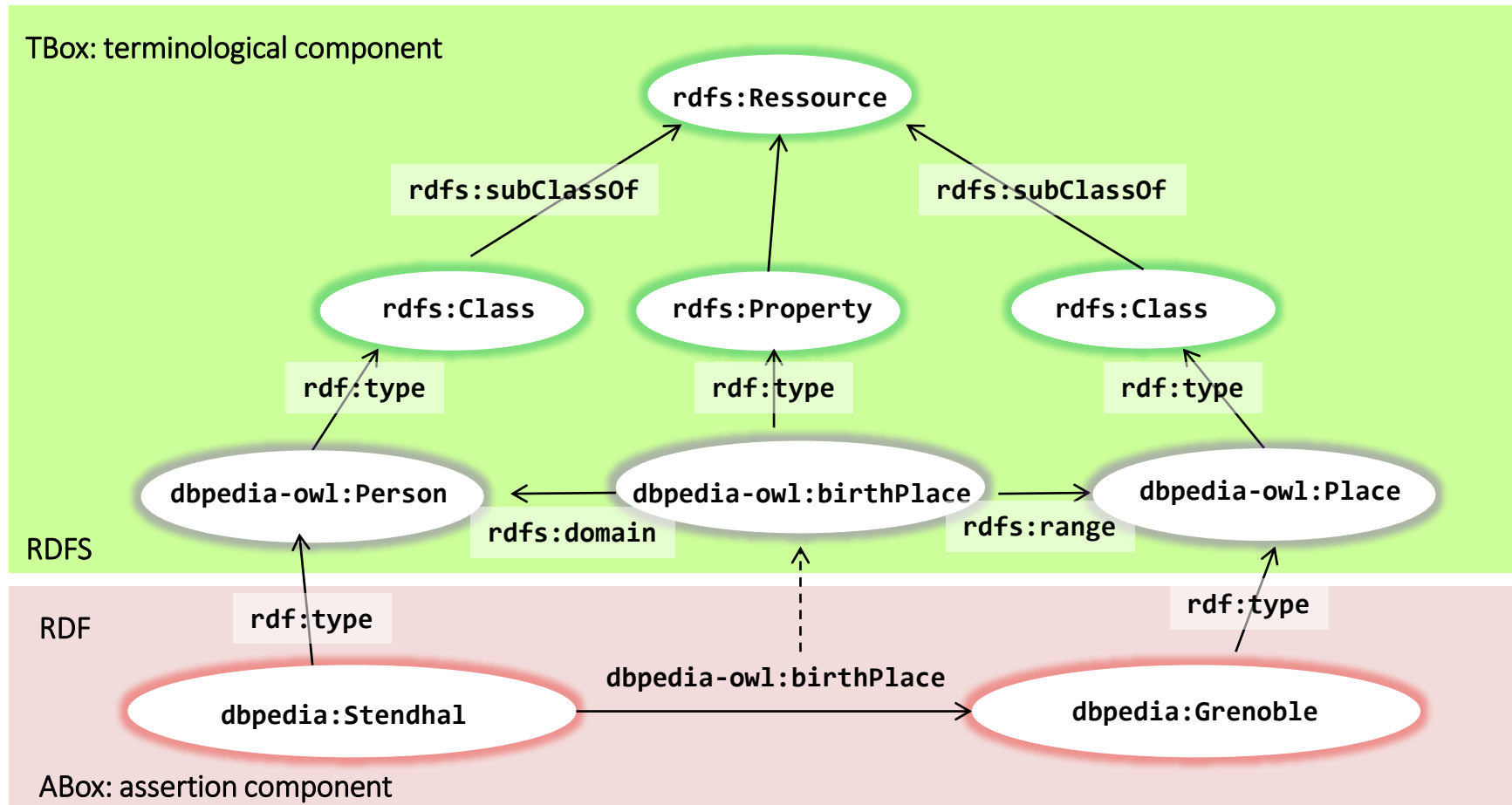
# RDFS Meta model

- RDFS is self-described in RDF



# RDF Schema

Prefixes dbpedia: http://dbpedia.org/resource/  
dbpedia-owl: http://dbpedia.org/ontology/  
rdfs: http://www.w3.org/2000/01/rdf-schema#



# What does RDFS give us ?

- RDF
  - A mechanism for publishing data.
  - Single (simple) data model.
  - Syntactic consistency between names (IRIs).
  - Low level integration of data.
  - Mash the graphs together and we're done.
- RDFS
  - Ability to use simple schema/vocabularies when describing our resources.
  - Consistent vocabulary use and sharing.
  - Basic inference

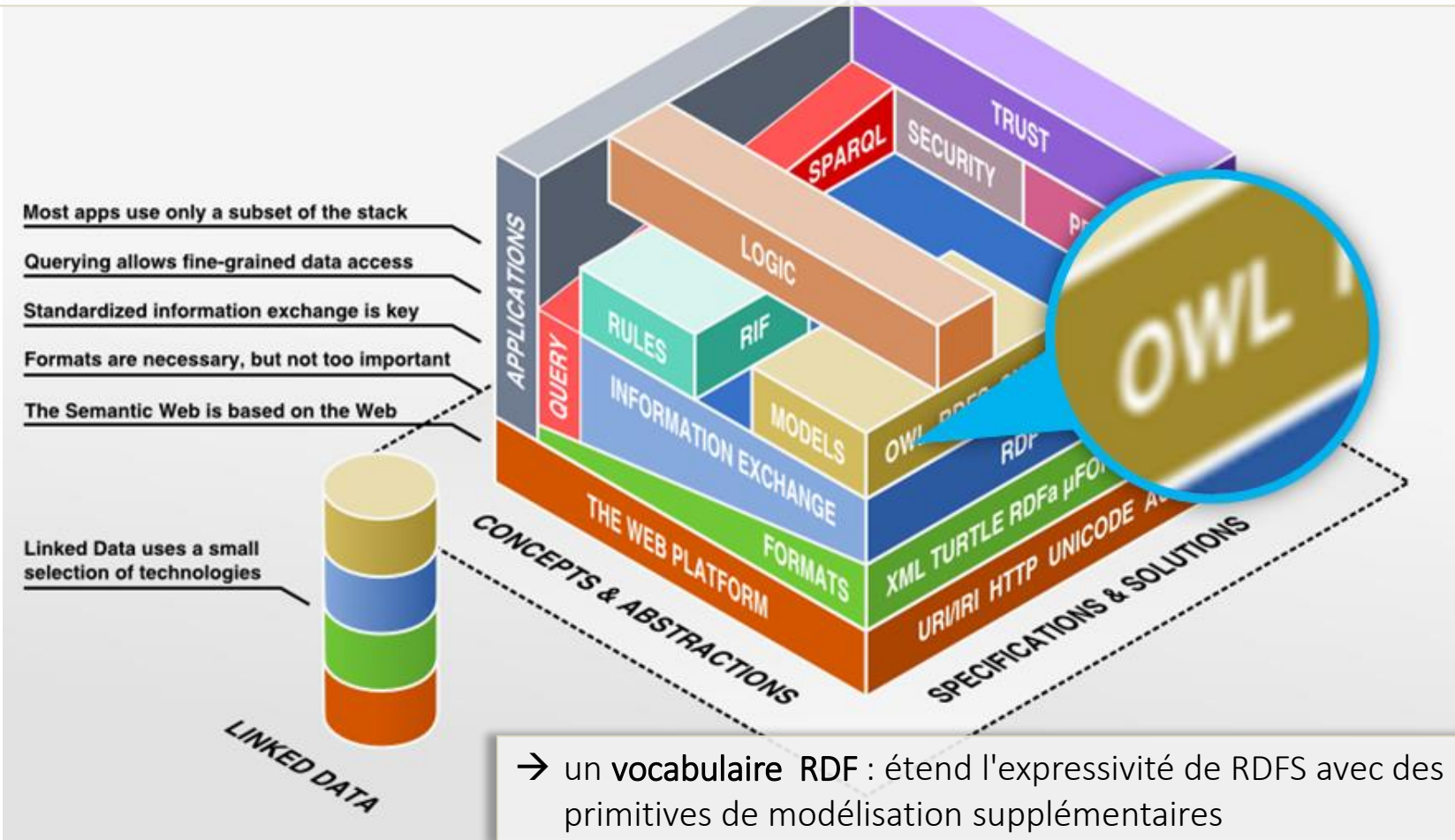
# Problems with RDF(S)

- RDF(S) is **too weak** to describe resources in sufficient detail
  - No **localized range and domain** constraints
    - Can't say that the range of **hasParent** is **Person** when applied to **Persons** and **Dog** when applied to **Dogs**
  - No **existence/cardinality** constraints
    - Can't say that all instances of **Person** have a mother that is also a **Person**, or that **Persons** have exactly 2 parents
  - No **transitive, inverse or symmetrical** properties
    - Can't say that **isPartOf** is a transitive property, that **hasPart** is the inverse of **isPartOf** or that **touches** is symmetrical
  - ...



# Web Ontology Language (OWL)

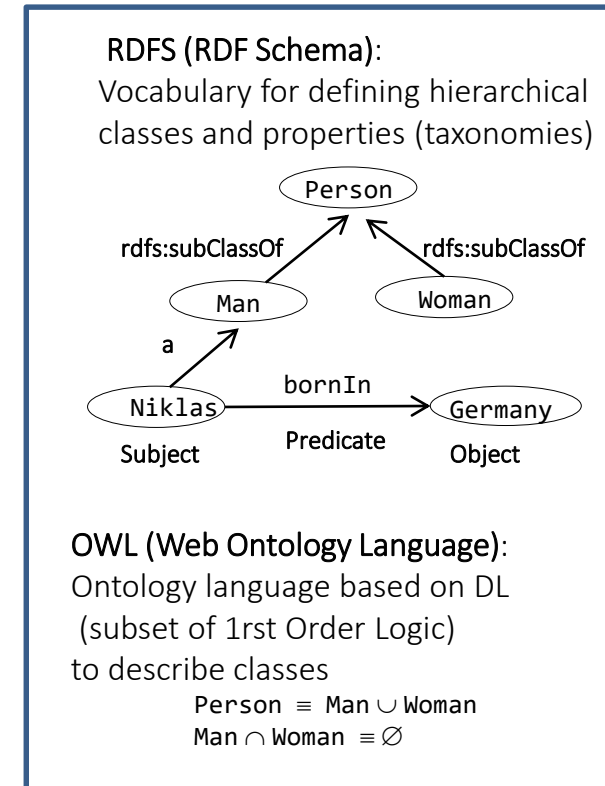
The Semantic Web Technology Stack  
(not a piece of cake...)



→ un vocabulaire RDF : étend l'expressivité de RDFS avec des primitives de modélisation supplémentaires

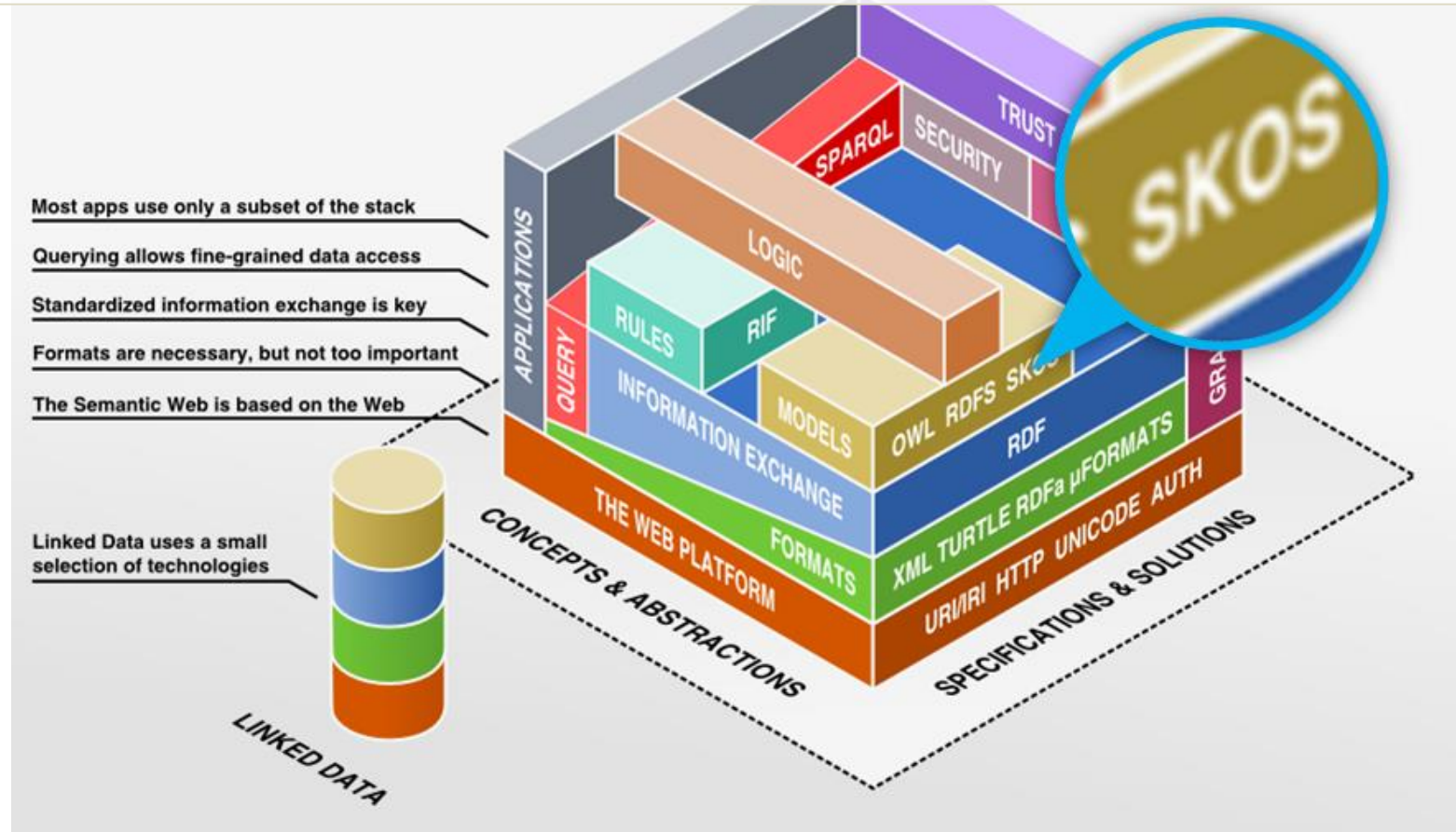
# Web Ontology Language (OWL)

- a W3C standard
  - OWL 1 : W3C recommendation 10 Feb. 2004 <http://www.w3.org/TR/owl-features/>
  - OWL 2 : W3C recommendation 11 Dec. 2012 <http://www.w3.org/TR/owl2-overview/>
- OWL vocabulary : a set of primitives described in RDF that extends RDFS vocabulary
  - namespace  
<http://www.w3.org/2002/07/owl#> ↔ **owl:**
- Far more expressive than RDFS
  - Classes can be describe by union, intersection, complement, properties restrictions.
  - notions of classes or properties equivalence, resources equality,
  - notions of inverse, symmetric, transitive ... properties
  - properties cardinality...
- Formal specification (based on Description Logics)
  - support for automated reasoning



# Simple Knowledge Organization System (SKOS)

modèle commun pour partager et lier (via RDF) sur le web différents systèmes d'organisation de connaissances tels thésaurus, taxinomies, systèmes de classification, système d'index.



# Simple Knowledge Organization System (SKOS)

- OWL → Ontologie      SKOS → thesaurus
  - *"l'objectif d'un thésaurus est de constituer des vocabulaires normalisés et d'organiser la liste des termes de ces vocabulaire sans forcément les définir, dans le but notamment d'indexer un corpus de ressources documentaires et de faciliter les recherches dans ce corpus"* Le web Sémantique, Fabien Gandon, Catherine Faron-Zucker, Olivier Corby, ed. Dunod 2012

(d'après <http://fr.wikipedia.org/wiki/Thésaurus>)

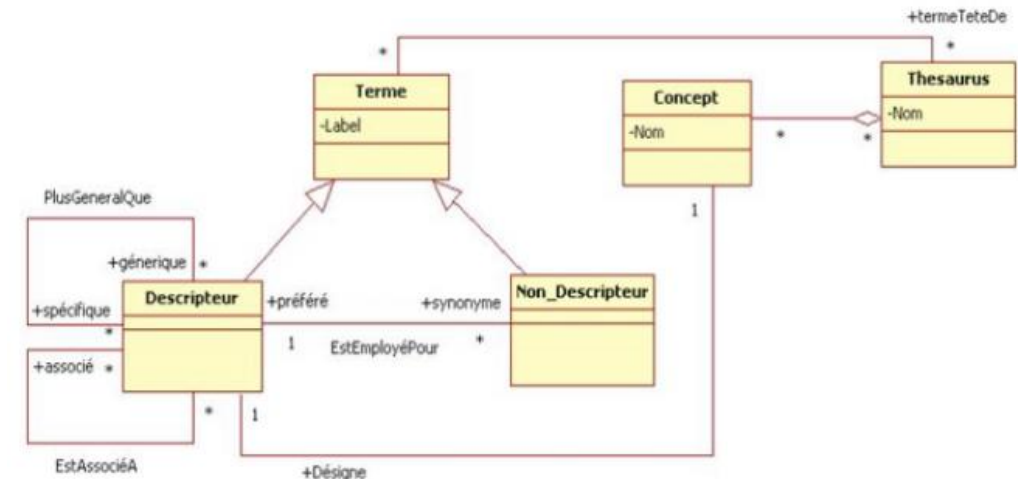
- Hierarchical relationships
  - Generic term (BT: broader term) , specific term (NT: narrower term).
  - Partitives relationships (whole-part relations) , instantiation relationships (to give examples)
- Associative relationships
  - RT: related term.
- Equivalent terms
  - ...

# Simple Knowledge Organization System (SKOS)

- W3C Standard (recommandation) : SKOS Simple Knowledge Organization System Reference (August 2009)  
<http://www.w3.org/TR/2009/REC-skos-reference-20090818/>

"SKOS a common data model for sharing and linking knowledge organization systems via the Web. Many knowledge organization systems, such as thesauri, taxonomies, classification schemes and subject heading systems, share a similar structure, and are used in similar applications. SKOS captures much of this similarity and makes it explicit, to enable data and technology sharing across diverse applications

The SKOS data model provides a standard, low-cost migration path for porting existing knowledge organization systems to the Semantic Web. SKOS also provides a lightweight, intuitive language for developing and sharing new knowledge organization systems. It may be used on its own, or in combination with formal knowledge representation languages such as the Web Ontology Language (OWL)"



- Technical notes:

- SKOS Simple Knowledge Organization System Primer <http://www.w3.org/TR/2009/NOTE-skos-primer-20090818/>
- SKOS Use Cases and Requirements <http://www.w3.org/TR/2009/NOTE-skos-ucr-20090818/>