

Semantic Web

— to Artificial Intelligence

Yue Ma

Laboratoire Interdisciplinaire des Sciences du Numérique
(LISN)

Université Paris-Saclay

yue.ma@universite-paris-saclay.fr

Project : Query answering over Linked Data

Q1 : Can you find the answers to (one of) the questions proposed in cours1.pdf on eCampus (page 7) or the question mentioned in Tim Berners-Lee's TED talk (2009)¹ between 11'20 and 12'30 ?

Q2 : Propose another question in natural language and give your answer by checking Linked Data (and combined with the help of Chatgpt or LLM–large language model–based methods).

Q3 : Can you explain the answers ? For answers from Linked Data, you can use "CONSTRUCT" to return the "justifications" of your answers. And how about the hybride approach by combing Linked Data and Chatgpt ?

You may refer to existing datasets, such as DBpedia, Wikidata, or more from

- ▶ <https://lod-cloud.net>
- ▶ <http://linkedlifedata.com>

1. https://www.ted.com/talks/tim_berners_lee_on_the_next_web

SPARQL example

Q : scientists who are born in Europe and received a Nobel Prize ?

SPARQL example

Q : scientists who are born in Europe and received a Nobel Prize?

SPARQL query (pseudocode) :

```
select x where {  
    <y, hasName , x> ;  
    <y, type, Scientist> ;  
    <y, hasWonPrize, NobelPrize> ;  
    <y, bornIn, Europe> .}
```

SPARQL example

Q : scientists who are born in Europe and received a Nobel Prize?

SPARQL query (pseudocode) :

```
select x where {  
    <y, hasName , x> ;  
    <y, type, Scientist> ;  
    <y, hasWonPrize, NobelPrize> ;  
    <y, bornIn, Europe> .}
```

RDF triple store :

```
<http://.../marie-curie, hasName, "Marie Curie">  
<http://.../marie-curie, type, Scientist>  
<http://.../marie-curie, hasWonPrize, NobelPrize>  
<http://.../marie-curie, bornIn, Europe>  
<http://.../einstein, hasName, "Albert Einstein">  
<http://.../einstein, type, Physicist>  
<http://.../einstein, hasWonPrize, NobelPrize>  
<http://.../einstein, birthPlace, Ulm>  
<Ulm, locatedIn, Germany>  
<Germany, partOf, Europe>
```

SPARQL example

Q : scientists who are born in Europe and received a Nobel Prize?

SPARQL query (pseudocode) :

```
select x where {  
    <y, hasName , x> ;  
    <y, type, Scientist> ;  
    <y, hasWonPrize, NobelPrize> ;  
    <y, bornIn, Europe> .}
```

A : "Marie Curie"

RDF triple store :

```
<http://.../marie-curie, hasName, "Marie Curie">  
<http://.../marie-curie, type, Scientist>  
<http://.../marie-curie, hasWonPrize, NobelPrize>  
<http://.../marie-curie, bornIn, Europe>  
<http://.../einstein, hasName, "Albert Einstein">  
<http://.../einstein, type, Physicist>  
<http://.../einstein, hasWonPrize, NobelPrize>  
<http://.../einstein, birthPlace, Ulm>  
<Ulm, locatedIn, Germany>  
<Germany, partOf, Europe>
```

SPARQL example

Q : scientists who are born in Europe and received a Nobel Prize ?

SPARQL query (pseudocode) :

```
select x where {  
    <y, hasName , x> ;  
    <y, type, Scientist> ;  
    <y, hasWonPrize, NobelPrize> ;  
    <y, bornIn, Europe> .}
```

A : "Marie Curie"

RDF triple store :

```
<http://.../marie-curie, hasName, "Marie Curie ">  
<http://.../marie-curie, type, Scientist>  
<http://.../marie-curie, hasWonPrize, NobelPrize>  
<http://.../marie-curie, bornIn, Europe>  
<http://.../einstein, hasName, "Albert Einstein">  
<http://.../einstein, type, Physicist>  
<http://.../einstein, hasWonPrize, NobelPrize>  
<http://.../einstein, birthPlace, Ulm>  
<Ulm, locatedIn, Germany>  
<Germany, partOf, Europe>
```

To get "Albert Einstein" in the answer, we need to know :

- ▶ Physicist is special type of Scientist (SubclassOf)

SPARQL example

Q : scientists who are born in Europe and received a Nobel Prize ?

SPARQL query (pseudocode) :

```
select x where {  
    <y, hasName , x>;  
    <y, type, Scientist>;  
    <y, hasWonPrize, NobelPrize>;  
    <y, bornIn, Europe> .}
```

A : "Marie Curie"

RDF triple store :

```
<http://.../marie-curie, hasName, "Marie Curie ">  
<http://.../marie-curie, type, Scientist>  
<http://.../marie-curie, hasWonPrize, NobelPrize>  
<http://.../marie-curie, bornIn, Europe>  
<http://.../einstein, hasName, "Albert Einstein">  
<http://.../einstein, type, Physicist>  
<http://.../einstein, hasWonPrize, NobelPrize>  
<http://.../einstein, birthPlace, Ulm>  
<Ulm, locatedIn, Germany>  
<Germany, partOf, Europe>
```

To get "Albert Einstein" in the answer, we need to know :

- ▶ Physicist is special type of Scientist (SubclassOf)
- ▶ <x birthPlace y>, <y locatedIn z>, <z partOf s>
→ <x bornIn s>

SPARQL example

Q : scientists who are born in Europe and received a Nobel Prize ?

SPARQL query (pseudocode) :

```
select x where {  
    <y, hasName , x>;  
    <y, type, Scientist>;  
    <y, hasWonPrize, NobelPrize>;  
    <y, bornIn, Europe> .}
```

A : "Marie Curie"

RDF triple store :

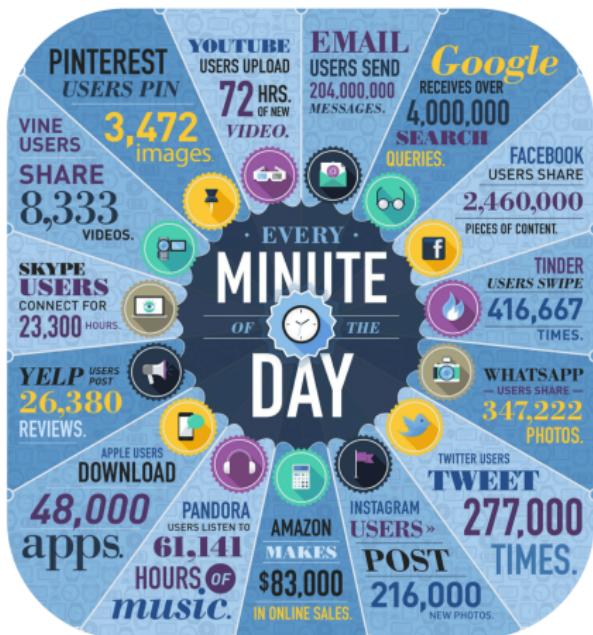
```
<http://.../marie-curie, hasName, "Marie Curie ">  
<http://.../marie-curie, type, Scientist>  
<http://.../marie-curie, hasWonPrize, NobelPrize>  
<http://.../marie-curie, bornIn, Europe>  
<http://.../einstein, hasName, "Albert Einstein">  
<http://.../einstein, type, Physicist>  
<http://.../einstein, hasWonPrize, NobelPrize>  
<http://.../einstein, birthPlace, Ulm>  
<Ulm, locatedIn, Germany>  
<Germany, partOf, Europe>
```

To get "Albert Einstein" in the answer, we need to know :

- ▶ Physicist is special type of Scientist (SubclassOf)
- ▶ <x birthPlace y>, <y locatedIn z>, <z partOf s>
→ <x bornIn s>

~ Abstract such implicit knowledge from queries !!!

We are Living in the Era of Big Data

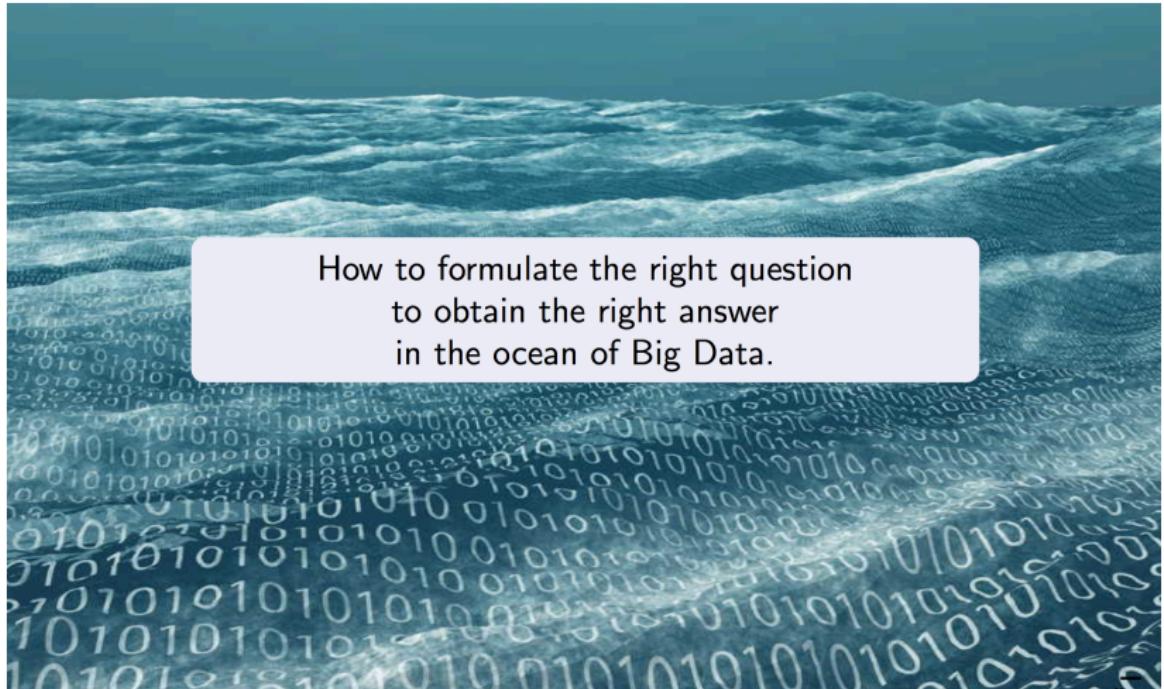


Data Never
Sleeps 2.0

The Problem : information access



The Problem : information access



[G. Xiao, 2015]

How much time is spent searching for data?



Engineers in industry spend a significant amount of their time searching for data that they require for their core tasks.

For example, in the oil&gas industry, 30–70% of engineers' time is spent looking for data and assessing its quality (Crompton, 2008).

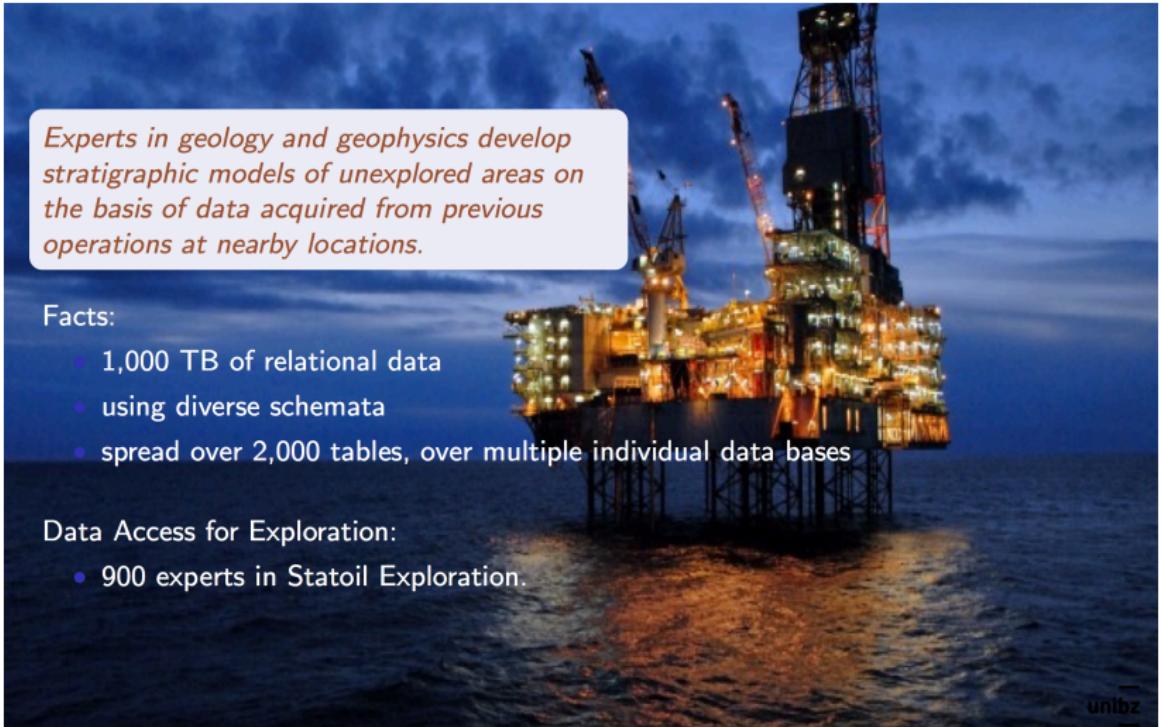
Example : Statoil Exploration



unibz

[G. Xiao, 2015]

How much time/money is spent searching for data ?



Experts in geology and geophysics develop stratigraphic models of unexplored areas on the basis of data acquired from previous operations at nearby locations.

Facts:

- 1,000 TB of relational data
- using diverse schemata
- spread over 2,000 tables, over multiple individual data bases

Data Access for Exploration:

- 900 experts in Statoil Exploration.

How much time/money is spent searching for data ?



A user query at Statoil

Show all norwegian wellbores with some additional attributes (wellbore id,). Limit to all wellbores with ... and show attributes like Limit to all wellbores with ... in and show key attributes in a table. After connecting to ... we could for instance limit further to cores in ... with and where it is larger than a given value, for instance We could also find out whether there are cores in which are not stored in (based on) and where there could be value. Some of the missing data we possibly own, other not.

How much time/money is spent searching for data ?

A use
Show
.....
.....
and s
limit
for in
not s
of the

```
SELECT [...]          table2a.attr1='keyword' AND
FROM                 table3a.attr2=table10c.attr1 AND
db_name.table1 table1, table3a.attr6=table6a.attr3 AND
db_name.table2 table2b, table3a.attr9='keyword' AND
db_name.table2 table2, table4a.attr10 IN ('keyword') AND
db_name.table3 table3a, table4a.attr1 IN ('keyword') AND
db_name.table3 table3b, table5a.kinds=table4a.attr13 AND
db_name.table3 table3c, table5b.kinds=table4c.attr74 AND
db_name.table3 table3d, table5b.name='keyword' AND
db_name.table4 table4a, (table6a.attr19=table10c.attr17 OR
db_name.table4 table4b, (table6a.attr2 IS NULL AND
db_name.table4 table4c, table10c.attr4 IS NULL)) AND
db_name.table4 table4d, table6a.attr14=table5b.attr14 AND
db_name.table4 table4e, table6a.attr2='keyword' AND
db_name.table4 table4f, (table6b.attr14=table10c.attr8 OR
db_name.table5 table5a, (table6b.attr4 IS NULL AND
db_name.table5 table5b, table10c.attr7 IS NULL)) AND
db_name.table6 table6a, table6b.attr19=table5a.attr55 AND
db_name.table6 table6b, table6b.attr2='keyword' AND
db_name.table7 table7a, table6b.attr2=table2b.attr9 AND
db_name.table7 table7b, table7a.attr19=table2b.attr19 AND
db_name.table8 table8b, table7a.attr17=table15.attr19 AND
db_name.table8 table8b, table4b.attr18='keyword' AND
db_name.table9 table9b, table8b.attr17=table7a.attr80 AND
db_name.table10 table10a, table8.attr19=table13.attr20 AND
db_name.table10 table10b, table8.attr19=table13.attr20 AND
db_name.table10 table10c, table9.attr10=table16.attr11 AND
db_name.table11 table11, table3b.attr19=table10c.attr18 AND
db_name.table12 table12, table3b.attr22=table12.attr63 AND
db_name.table13 table13, table3b.attr66='keyword' AND
db_name.table14 table14, table10a.attr54=table7a.attr8 AND
db_name.table15 table15, table10a.attr70=table10c.attr10 AND
db_name.table16 table16, table10a.attr16=table4d.attr11 AND
WHERE [...]          table4c.attr99='keyword' AND
                     table4c.attr1='keyword' AND
                     table11.attr10=table5a.attr10 AND
                     table11.attr40='keyword' AND
                     table11.attr50='keyword' AND
                     table2b.attr1=table1.attr8 AND
                     table2b.attr9 IN ('keyword') AND
                     table12.attr9 IN ('keyword') AND
                     table7b.attr1=table2a.attr10 AND
                     table3c.attr13=table10c.attr1 AND
                     table3c.attr10=table6b.attr20 AND
                     table3c.attr13='keyword' AND
                     table10b.attr16=table10a.attr7 AND
                     table10b.attr11=table7b.attr8 AND
                     table10b.attr13=table4b.attr89 AND
                     table13.attr1=table2b.attr10 AND
                     table13.attr20='keyword' AND
                     table13.attr15='keyword' AND
                     table3d.attr49=table12.attr18 AND
                     table3d.attr18=table10c.attr11 AND
                     table3d.attr14='keyword' AND
                     table4d.attr17 IN ('keyword') AND
                     table4d.attr19 IN ('keyword') AND
                     table16.attr28=table11.attr56 AND
                     table16.attr16=table10b.attr78 AND
                     table16.attr5=table14.attr56 AND
                     table4e.attr34 IN ('keyword') AND
                     table4e.attr48 IN ('keyword') AND
                     table4f.attr89=table5b.attr7 AND
                     table4f.attr45 IN ('keyword') AND
                     table4f.attr1='keyword' AND
                     table10c.attr2=table4e.attr19 AND
                     (table10c.attr78=table12.attr56 OR
                      (table10c.attr55 IS NULL AND
                       table12.attr17 IS NULL))
```

Challenges Accessing Big Data

```
SELECT [...]          table2a.attr1='keyword' AND      table11.attr10=table5a.attr10 AND  
FROM                 table3a.attr2=table10c.attr1 AND      table11.attr40='keyword' AND  
db_name.table1 table1,    table3a.attr6=table6a.attr3 AND      table11.attr50='keyword' AND  
db_name.table2 table2a,   table3a.attr9='keyword' AND      table2b.attr1=table1.attr8 AND  
db_name.table2 table2b,   table4a.attr10 IN ('keyword') AND      table2b.attr9 IN ('keyword') AND  
db_name.table3 table3a,   table4a.attr1 IN ('keyword') AND      table12.attr9 IN ('keyword') AND  
db_name.table3 table3b,   table5a.kinds=table4a.attr13 AND      table7b.attr1=table2a.attr10 AND  
db_name.table3 table3c,   table5b.kinds=table4c.attr74 AND      table3c.attr13=table10c.attr1 AND  
db_name.table3 table3d,   table5b.name='keyword' AND      table3c.attr10=table6b.attr20 AND  
db_name.table4 table4a,   (table6a.attr19=table10c.attr17 OR      table3c.attr13='keyword' AND  
db_name.table4 table4b,   (table6a.attr2 IS NULL AND      table10b.attr16=table10a.attr7 AND  
db_name.table4 table4c,   table10c.attr4 IS NULL)) AND
```

A use

Cl ...

At Statoil, it takes up to 4 days to formulate a query in SQL.

Statoil loses up to **50.000.000€** per year because of this!!

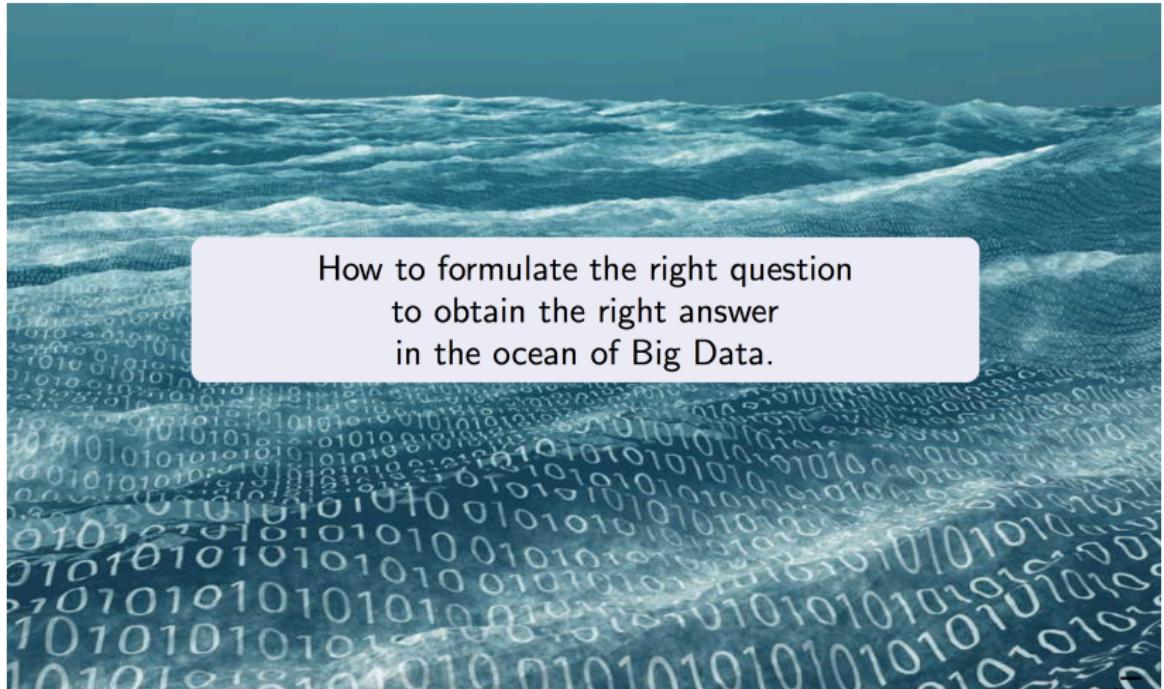
not s' of the

```
db_name.table9 table9,      table8.attr19=table7a.attr80 AND      table16.attr28=table11.attr56 AND  
db_name.table10 table10a,   table8.attr19=table13.attr20 AND      table16.attr16=table10b.attr78 AND  
db_name.table10 table10b,   table8.attr4='keyword' AND      table16.attr5=table14.attr56 AND  
db_name.table10 table10c,   table9.attr10=table16.attr11 AND      table4e.attr34 IN ('keyword') AND  
db_name.table11 table11,    table3b.attr19=table10c.attr18 AND      table4e.attr48 IN ('keyword') AND  
db_name.table12 table12,    table3b.attr22=table12.attr63 AND      table4f.attr89=table5b.attr7 AND  
db_name.table13 table13,    table3b.attr66='keyword' AND      table4f.attr45 IN ('keyword') AND  
db_name.table14 table14,    table10a.attr54=table7a.attr8 AND      table4f.attr1='keyword' AND  
db_name.table15 table15,    table10a.attr70=table10c.attr10 AND      table10c.attr2=table4e.attr19 AND  
db_name.table16 table16,    table10a.attr16=table4d.attr11 AND      (table10c.attr78=table12.attr56 OR  
WHERE [...]           table4c.attr99='keyword' AND      (table10c.attr55 IS NULL AND  
                           table4c.attr1='keyword' AND      table12.attr17 IS NULL))
```

ome

unibz

The Problem : information access



[G. Xiao, 2015]

Need for Abstraction

We need to facilitate access to Data

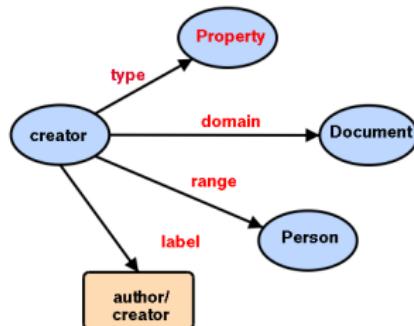
- ▶ by abstracting away from how the data is stored, and
- ▶ by making use of high level views on the data, so called ontologies.

RDF Schema (RDFs) : example

In English	The graph
<ul style="list-style-type: none">Dog1 is an animalCat1 is a catCats are animalsZoos host animalsZoo1 hosts the Cat2	<p>The graph illustrates the relationships between the following entities:</p> <ul style="list-style-type: none">ex:dog1 is of type ex:animal (<code>rdf:type</code>).ex:cat1 is of type ex:cat (<code>rdf:type</code>).ex:cat is a subclass of ex:animal (<code>rdfs:subClassOf</code>).ex:zool hosts ex:cat2 (<code>zoo:host</code>).ex:zool has ex:animal as its range (<code>rdfs:range</code>). <p>Legend: RDF special terms (green), RDFS special terms (orange)</p>
RDF/turtle	
<pre>@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> . @prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> . @prefix ex: <http://example.org/> . @prefix zoo: <http://example.org/zoo/> . ex:dog1 rdf:type ex:animal . ex:cat1 rdf:type ex:cat . ex:cat rdfs:subClassOf ex:animal . zoo:host rdfs:range ex:animal . ex:zool zoo:host ex:cat2 .</pre>	

RDFS Vocabulary vs. implicite information

Unlike sparql query which only matches graph patterns, RDFS vocabularies can help to deduce implicite information.
~> Metadata reasoning !

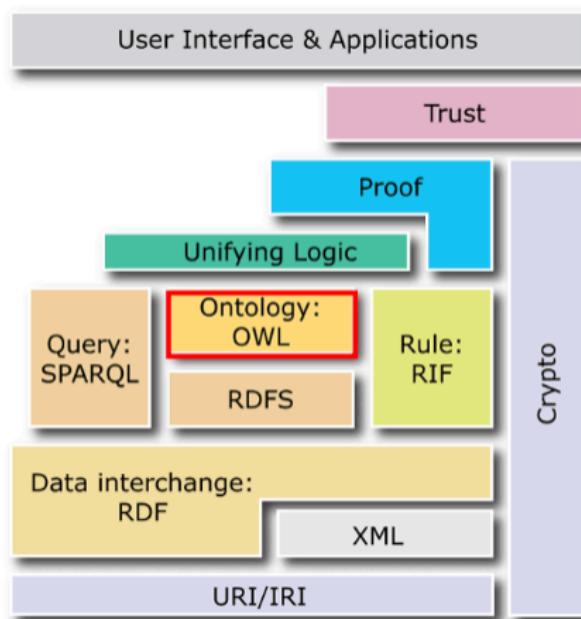


```
@prefix ex : <http://www.lri.fr/sw/example> .  
ex :LesMiserables ex :creator ex :Hugo .  
ex :creator rdf:type rdf:Property .  
ex :creator rdfs:domain ex :Document .  
ex :creator rdfs:range ex :Person .  
ex :creator rdfs:label "author/creator" .
```

So we can deduce that Hugo is a person and that LesMiserables is a document.

Are there some other
interesting vocabularies ?

Semantic Web Cake Layer



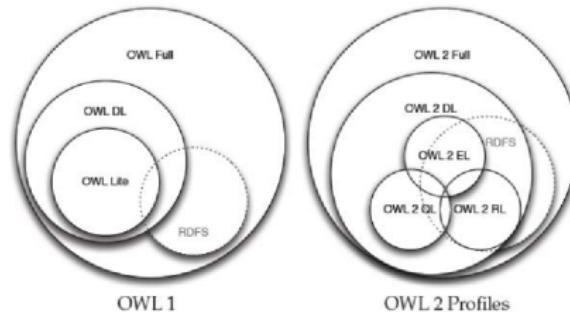
Ontology and Semantic Web

OWL Web Ontology Language

Status	Published
Year started	2004
Editors	Mike Dean, Guus Schreiber
Base standards	Resource Description Framework, RDFS
Domain	Semantic Web
Abbreviation	OWL
Website	OWL Reference ↗

OWL 2 Web Ontology Language

Status	Published
Year started	2009
Editors	W3C OWL Working Group
Base standards	Resource Description Framework, RDFS
Domain	Semantic Web
Abbreviation	OWL 2
Website	OWL2 Overview ↗



OWL (Ontology Web Language) ... and DLs

```
:Mary rdf:type :Woman .
```

```
:John :hasWife :Mary .
```

```
:John owl:differentFrom :Bill .
```

{John} ⊓ {Bill} ⊑ ⊥

```
:James owl:sameAs :Jim.
```

{John} ≡ {Jim}

```
:John :hasAge "51"^^xsd:nonNegativeInteger .
```

```
[] rdf:type owl:NegativePropertyAssertion ;  
owl:sourceIndividual :Bill ;  
owl:assertionProperty :hasWife ;  
owl:targetIndividual :Mary .
```

¬hasWife(Bill,Mary)

```
[] rdf:type owl:NegativePropertyAssertion ;  
owl:sourceIndividual :Jack ;  
owl:assertionProperty :hasAge ;  
owl:targetValue 53 .
```

Inconsistent information ???

:John owl :differentFrom :Bill .

:John owl :sameAs :Bill .

~> Inconsistency !

:John :hasWife :Mary .

$\neg hasWife(John, Mary)$.

~> Inconsistency !

OWL (Ontology Web Language) ... and DLs

```
:Woman rdfs:subClassOf :Person .  
:Person owl:equivalentClass :Human .
```

```
[] rdf:type owl:AllDisjointClasses ;  
owl:members ( :Woman :Man ) .
```

Woman \sqcap **Man** $\sqsubseteq \perp$

```
:hasWife rdfs:subPropertyOf :hasSpouse .
```

```
:hasWife rdfs:domain :Man ;  
rdfs:range :Woman .
```

Incoherent information ???

:x rdfs :subClassOf :Man
:x rdfs :subClassOf :Woman

~> :x is an incoherent concept!

If there is an incoherent concept, we say the ontology is incoherent.

OWL (Ontology Web Language) ... and DLs

```
:Mother owl:equivalentClass [  
    rdf:type owl:Class ;  
    owl:intersectionOf ( :Woman :Parent )  
] .
```

$$\text{Mother} \equiv \text{Woman} \sqcap \text{Parent}$$

```
:Parent owl:equivalentClass [  
    rdf:type owl:Class ;  
    owl:unionOf ( :Mother :Father )  
] .
```

$$\text{Parent} \equiv \text{Mother} \sqcup \text{Father}$$

```
:ChildlessPerson owl:equivalentClass [  
    rdf:type owl:Class ;  
    owl:intersectionOf ( :Person [ owl:complementOf :Parent ] )  
] .
```

$$\text{ChildlessPerson} \equiv \text{Person} \sqcap \neg \text{Parent}$$

```
:Grandfather rdfs:subClassOf [  
    rdf:type owl:Class ;  
    owl:intersectionOf ( :Man :Parent )  
] .
```

OWL (Ontology Web Language) ... and DLs

```
:Jack rdf:type [  
    rdf:type owl:Class ;  
    owl:intersectionOf ( :Person  
        [ rdf:type owl:Class ;  
          owl:complementOf :Parent ]  
    )  
] .
```

Person ⊓ ¬Parent (Jack)

OWL (Ontology Web Language) ... and DLs

```
:Parent owl:equivalentClass [  
    rdf:type          owl:Restriction ;  
    owl:onProperty   :hasChild ;  
    owl:someValuesFrom :Person  
] .
```

Parent $\equiv \exists \text{hasChild}.\text{Person}$

```
:Orphan owl:equivalentClass [  
    rdf:type          owl:Restriction ;  
    owl:onProperty   [ owl:inverseOf :hasChild ] ;  
    owl:allValuesFrom :Dead  
] .
```

Orphan $\equiv \forall \text{hasChild}^-.\text{Dead}$

OWL (Ontology Web Language) ... and DLs

```
:JohnsChildren owl:equivalentClass [  
    rdf:type          owl:Restriction ;  
    owl:onProperty   :hasParent ;  
    owl:hasValue     :John  
] .
```

JohnsChildren $\equiv \exists \text{hasParent}.\{\text{John}\}$

```
:NarcisticPerson owl:equivalentClass [  
    rdf:type          owl:Restriction ;  
    owl:onProperty   :loves ;  
    owl:hasSelf      "true"^^xsd:boolean .  
] .
```

NarcisticPerson $\equiv \exists \text{loves}.\text{Self}$

OWL (Ontology Web Language) ... and DLs

```
:John rdf:type [  
    rdf:type owl:Restriction ;  
    owl:maxQualifiedCardinality "4"^^xsd:nonNegativeInteger ;  
    owl:onProperty :hasChild ;  
    owl:onClass :Parent  
] .  
  
:John rdf:type [  
    rdf:type owl:Restriction ;  
    owl:minQualifiedCardinality "2"^^xsd:nonNegativeInteger ;  
    owl:onProperty :hasChild ;  
    owl:onClass :Parent  
] .  
  
:John rdf:type [  
    rdf:type owl:Restriction ;  
    owl:qualifiedCardinality "3"^^xsd:nonNegativeInteger ;  
    owl:onProperty :hasChild ;  
    owl:onClass :Parent  
] .
```

$\leq 4 \text{ hasChild.} \text{Parent} \text{ (John)}$

$\geq 2 \text{ hasChild.} \text{Parent} \text{ (John)}$

$= 3 \text{ hasChild.} \text{Parent} \text{ (John)}$

OWL (Ontology Web Language) ... and DLs

```
:John rdf:type [  
    rdf:type owl:Restriction ;  
    owl:cardinality "5^^xsd:nonNegativeInteger ;  
    owl:onProperty :hasChild  
] .
```

=5 hasChild.T (John)

```
:MyBirthdayGuests owl:equivalentClass [  
    rdf:type owl:Class ;  
    owl:oneOf ( :Bill :John :Mary )  
] .
```

MyBirthdayGuests ≡ {Bill, John, Mary}

OWL (Ontology Web Language) ... and DLs

```
:hasParent owl:inverseOf :hasChild .  
  
:Orphan owl:equivalentClass [  
    rdf:type owl:Restriction ;  
    owl:onProperty [ owl:complementOf :hasChild ] ;  
    owl:allValuesFrom :Dead  
] .  
  
:hasSpouse rdf:type owl:SymmetricProperty .  
:hasChild rdf:type owl:AsymmetricProperty .  
:hasParent owl:propertyDisjointWith :hasSpouse .  
:hasRelative rdf:type owl:ReflexiveProperty .  
:parentOf rdf:type owl:IrreflexiveProperty .  
:hasHusband rdf:type owl:FunctionalProperty .  
:hasHusband rdf:type owl:InverseFunctionalProperty .  
:hasAncestor rdf:type owl:TransitiveProperty .
```

$$\text{Orphan} \equiv \forall \text{hasChild} . \text{Dead}$$

OWL (Ontology Web Language) ... and DLs

```
:hasGrandparent owl:propertyChainAxiom ( :hasParent :hasParent ).
```

hasParent \circ hasParent \sqsubseteq hasGrandParent

```
:Person owl:hasKey ( :hasSSN ) .
```

In OWL 2 a collection of (data or object) properties can be assigned as a key to a class expression. This means that each named instance of the class expression is uniquely identified by the set of values which these properties attain in relation to the instance.

That is, if two named instances of the class coincide on values for each of key properties, then these two individuals are the same.

OWL (Ontology Web Language) ... and DLs

```
:personAge  owl:equivalentClass
[ rdf:type rdfs:Datatype;
  owl:onDatatype xsd:integer;
  owl:withRestrictions (
    [ xsd:minInclusive "0"^^xsd:integer ]
    [ xsd:maxInclusive "150"^^xsd:integer ]
  )
]
.

:majorAge  owl:equivalentClass
[ rdf:type rdfs:Datatype;
  owl:intersectionOf (
    :personAge
    [ rdf:type rdfs:Datatype;
      owl:datatypeComplementOf :minorAge ]
  )
]
```

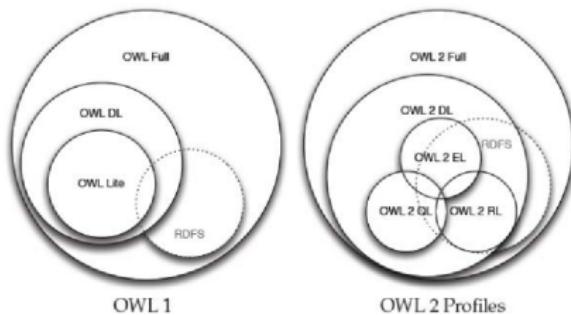
Datatype facets

OWL (Ontology Web Language) ... and DLs

Feature	Related OWL vocabulary	FOL	DL
top/bottom class	<code>owl:Thing/owl:Nothing</code>	(axiomatise)	\top/\perp
Class intersection	<code>owl:intersectionOf</code>	\wedge	\sqcap
Class union	<code>owl:unionOf</code>	\vee	\sqcup
Class complement	<code>owl:complementOf</code>	\neg	\neg
Enumerated class	<code>owl:oneOf</code>	(ax. with \approx)	{a}
Property restrictions	<code>owl:onProperty</code>		
Existential	<code>owl:someValueFrom</code>	$\exists y \dots$	$\exists R.C$
Universal	<code>owl:allValuesFrom</code>	$\forall y \dots$	$\forall R.C$
Min. cardinality	<code>owl:minQualifiedCardinality</code> <code>owl:onClass</code>	$\exists y_1 \dots y_n \dots$	$\geq n \text{ S.C}$
Max. cardinality	<code>owl:maxQualifiedCardinality</code> <code>owl:onClass</code>	$\forall y_1 \dots y_{n+1} \dots \rightarrow \dots$	$\leq n \text{ S.C}$
Local reflexivity	<code>owl:hasSelf</code>	$R(x,x)$	$\exists R.\text{Self}$

Wrap-up

- ▶ Presentation of W3C Semantic Web vocabulary standards (RDF, RDFS, OWL, OWL2).
Well, not yet for the distinction between OWL and OWL2 !



- ▶ Next : the formal definition of their semantics (DL)
 - ▶ Lab : create your first ontology