Semantic Web

— to Artificial Intelligence

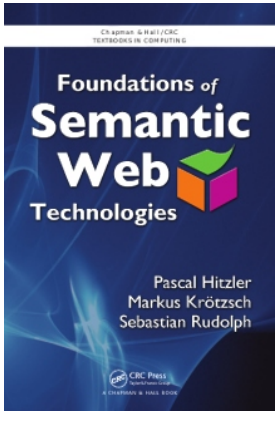
Yue Ma

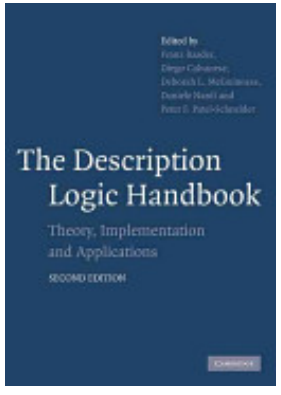
Laboratoire de Recherche en Informatique (LRI)
Université Paris Sud
ma@lri.fr

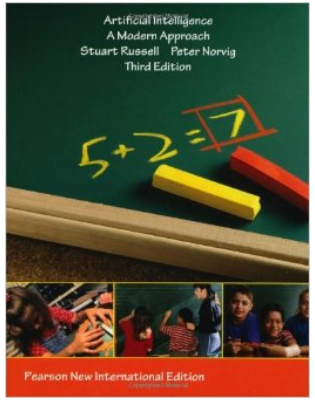
General Information

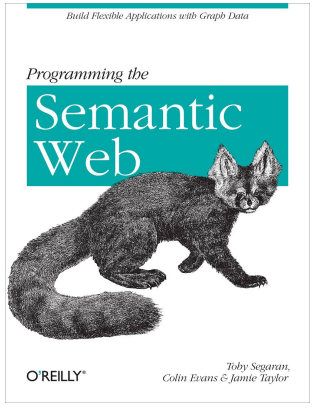
- This lecture is inspired by the following courses :
  - Knowledge Representation for the Semantic Web
    Pascal Hitzler, Wright State University, US
  - Foundations of Semantic Web Technologies
    Sebastian Rudolph, TU Dresden, Germany
  - Knowledge Graph Analysis
    Jens Lehmann, University of Bonn, Germany
  - Introduction à l'intelligence artificielle
    Laurent Simon, Polytech Paris Sud

- Reference books :









General Information

- Course materials : `www.lri.fr/~ma/M2DK`
- Organizational matters :
  - Initial plans :
    - Part 1 : Semantic Web Standards (RDF, RDFS, OWL, SPARQL)
    - Part 2 : Semantics and Reasoning for Semantic Web
    - Projects : Querying Semantic Data, Ontology Reasoning
  - Grading : average(Projects, Exam)

Project 1 : Query answering over Linked Data

Q1 : Can you find the answers to (one of) the questions proposed in https ://www.lri.fr/ ma/M2DK/cours1.pdf (pages 10-15) 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.

Use "CONSTRUCT" to return the "justifications" of your answers.

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 bitherPlace 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 :

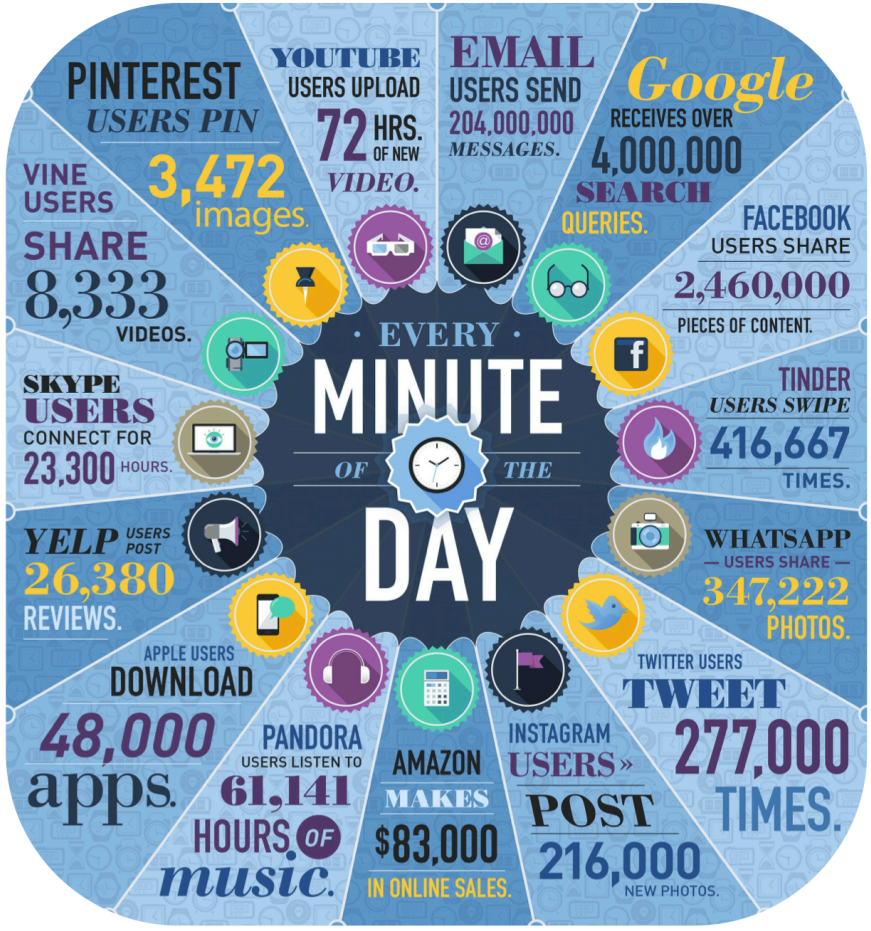
<http ://.../marie-curie, hasName, “Marie Curie ”>
<http ://.../marie-curie, type, Scientist>
<http ://.../marie-curie, hasWonPrize, NobelPrize>
<http ://.../marie-curie, bornIn, Europe>
<http ://.../einstein, hasName, “Albert Einstein”>
<http ://.../einstein, type, Physicist>
<http ://.../einstein, 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 bitherPlace y>, <y locatedIn z>, <z partOf s> $\rightarrow$ <x bornIn s>

$\rightsquigarrow$ Abstract such implicit knowledge from queries !!!

We are Living in the Era of Big Data

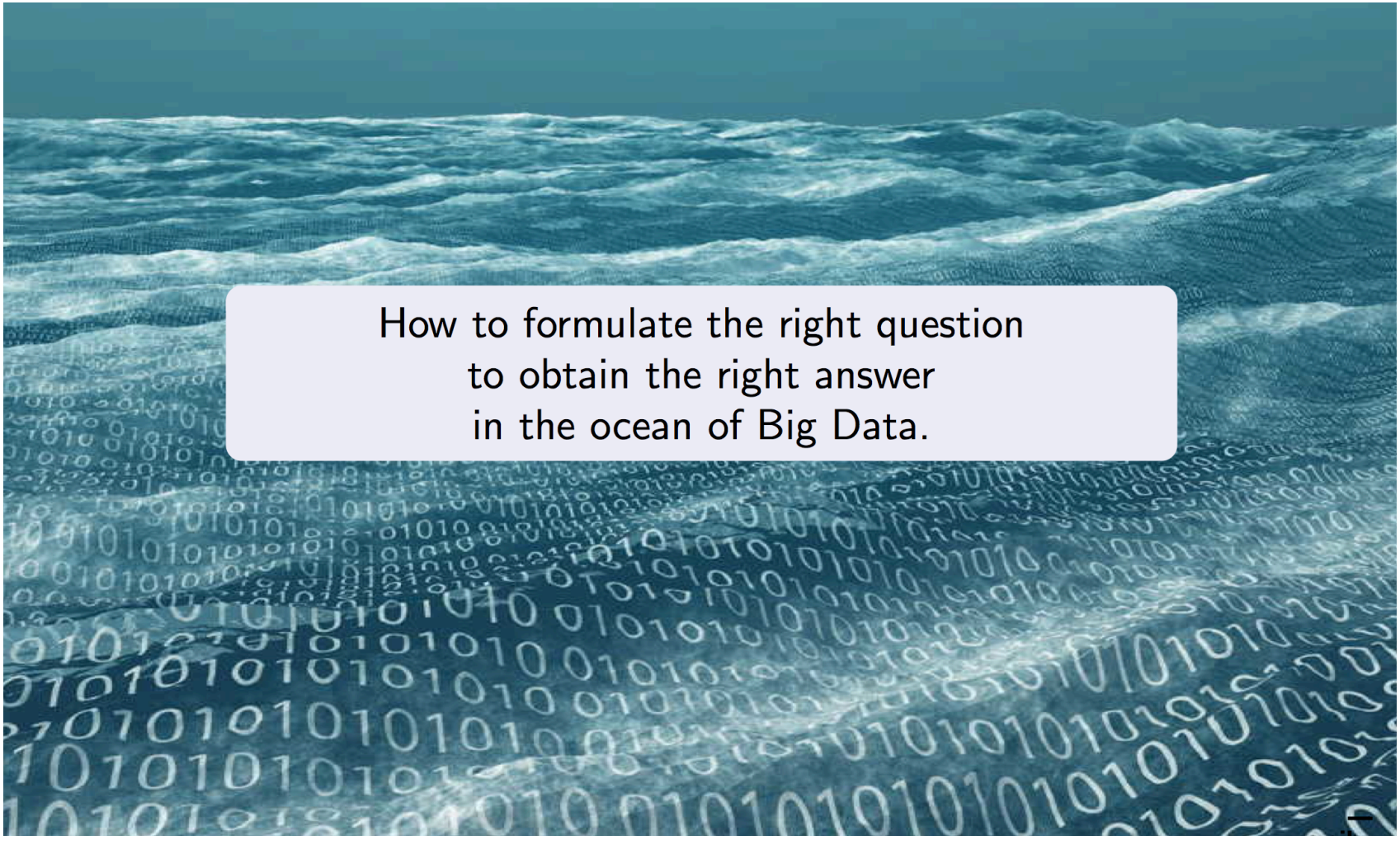


Data Never Sleeps 2.0

The Problem : information access

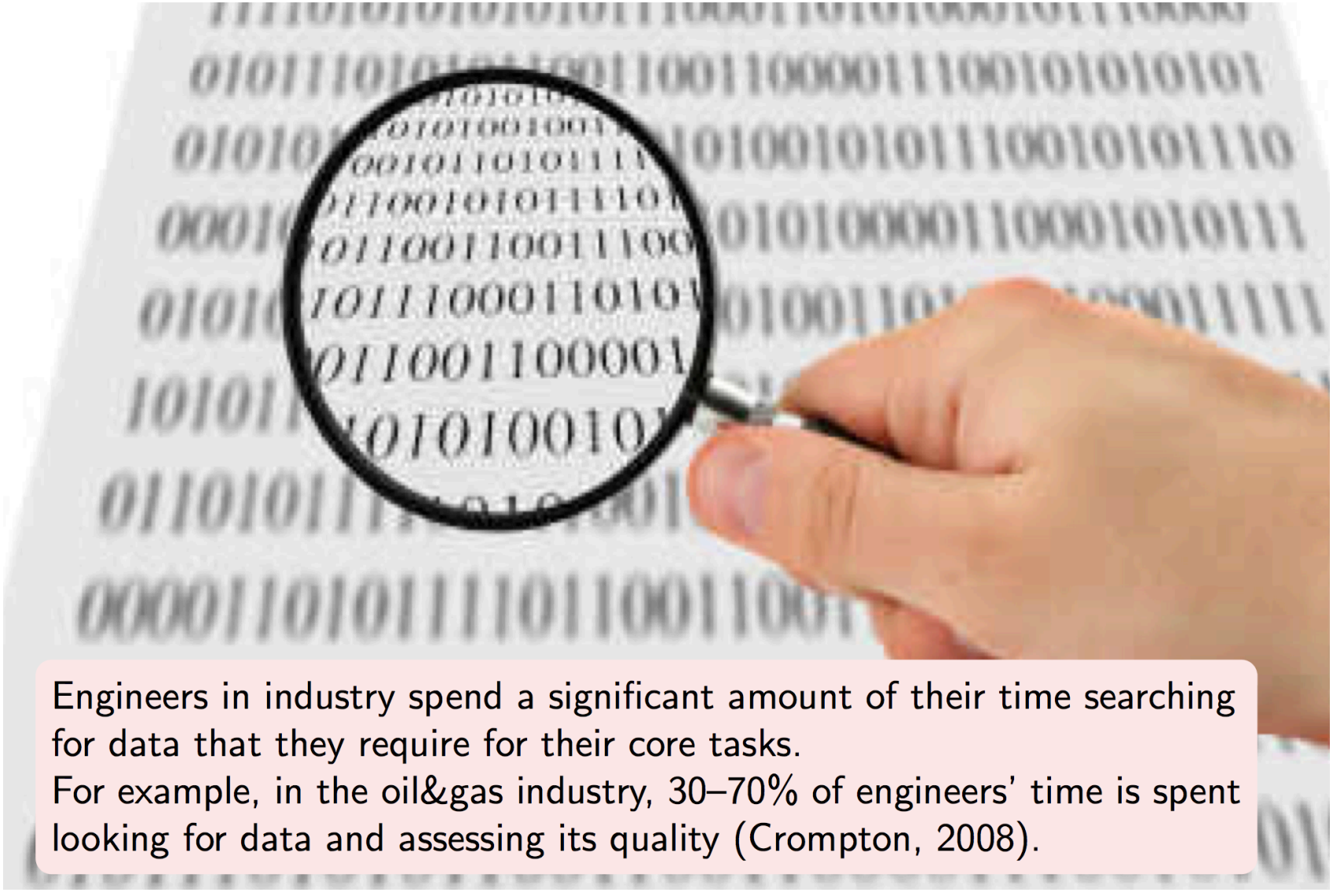
[G. Xiao, 2015]

The Problem : information access



[G. Xiao, 2015]

How much time is spent searching for data ?



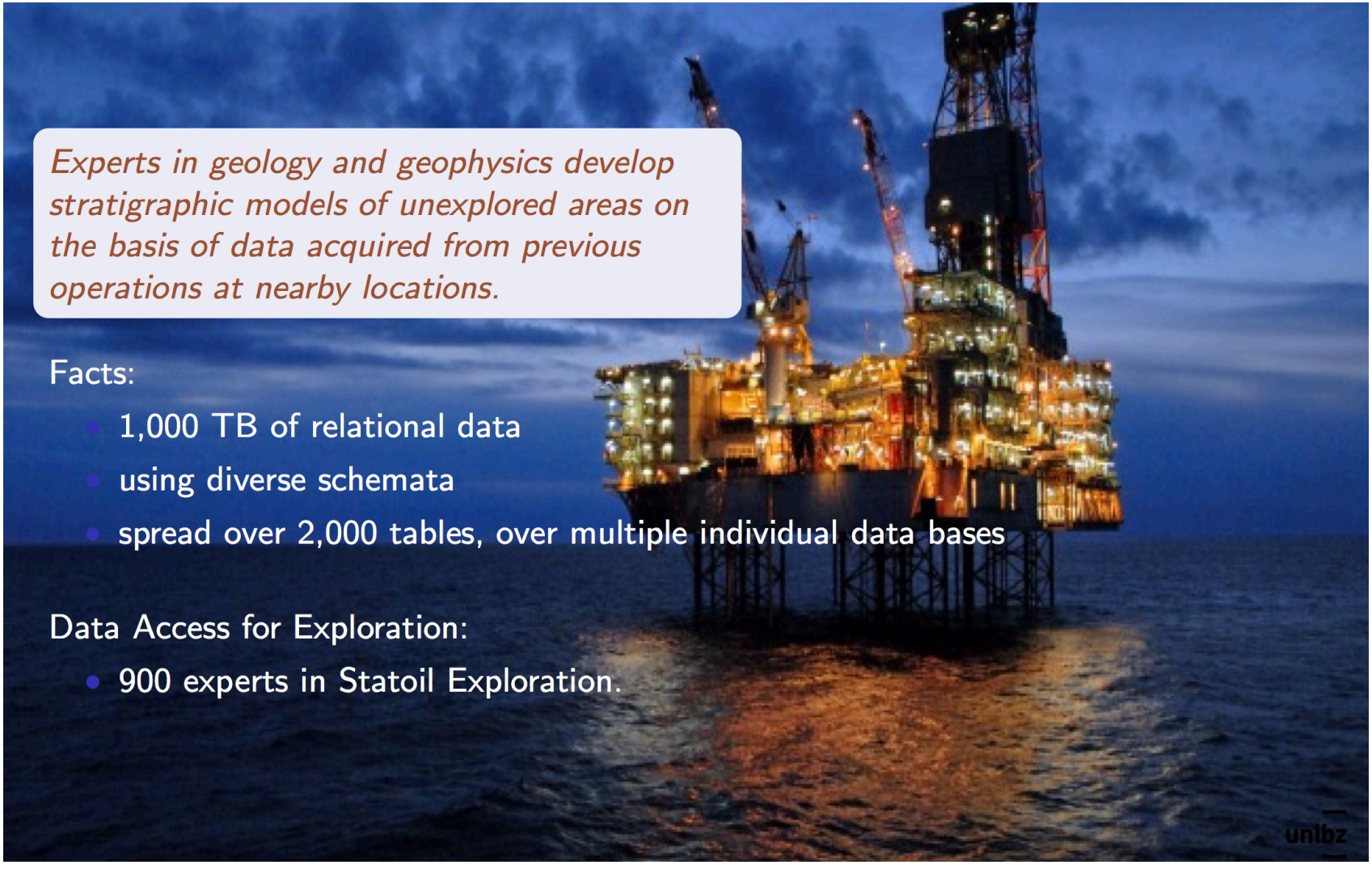
Engineers in industry spend a significant amount of their time searching for data that they require for their core tasks.
For example, in the oil&gas industry, 30–70% of engineers' time is spent looking for data and assessing its quality (Crompton, 2008).

[G. Xiao, 2015]

Example : Statoil Exploration

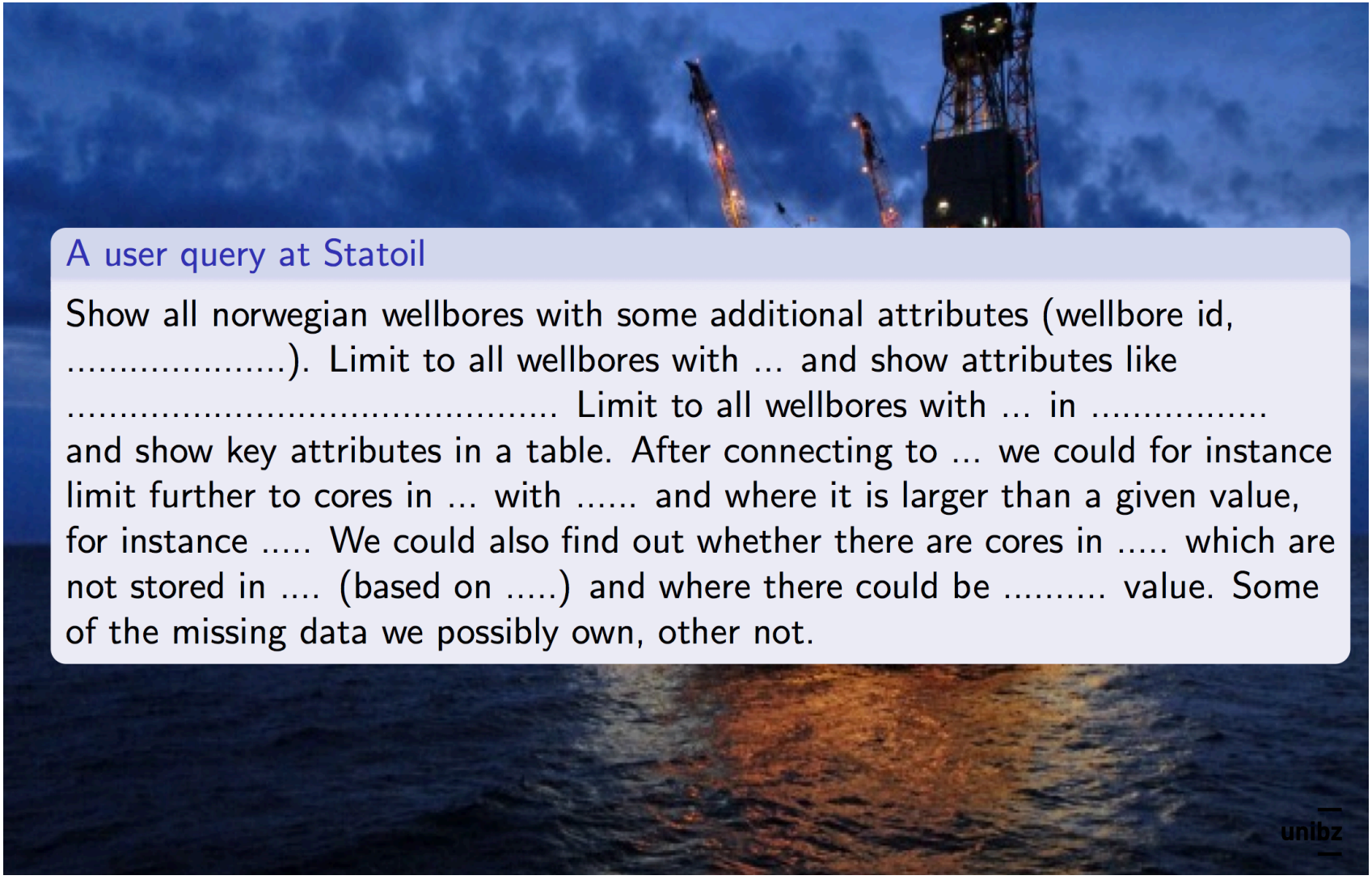
[G. Xiao, 2015]

How much time/money is spent searching for data ?



[G. Xiao, 2015]

How much time/money is spent searching for data ?



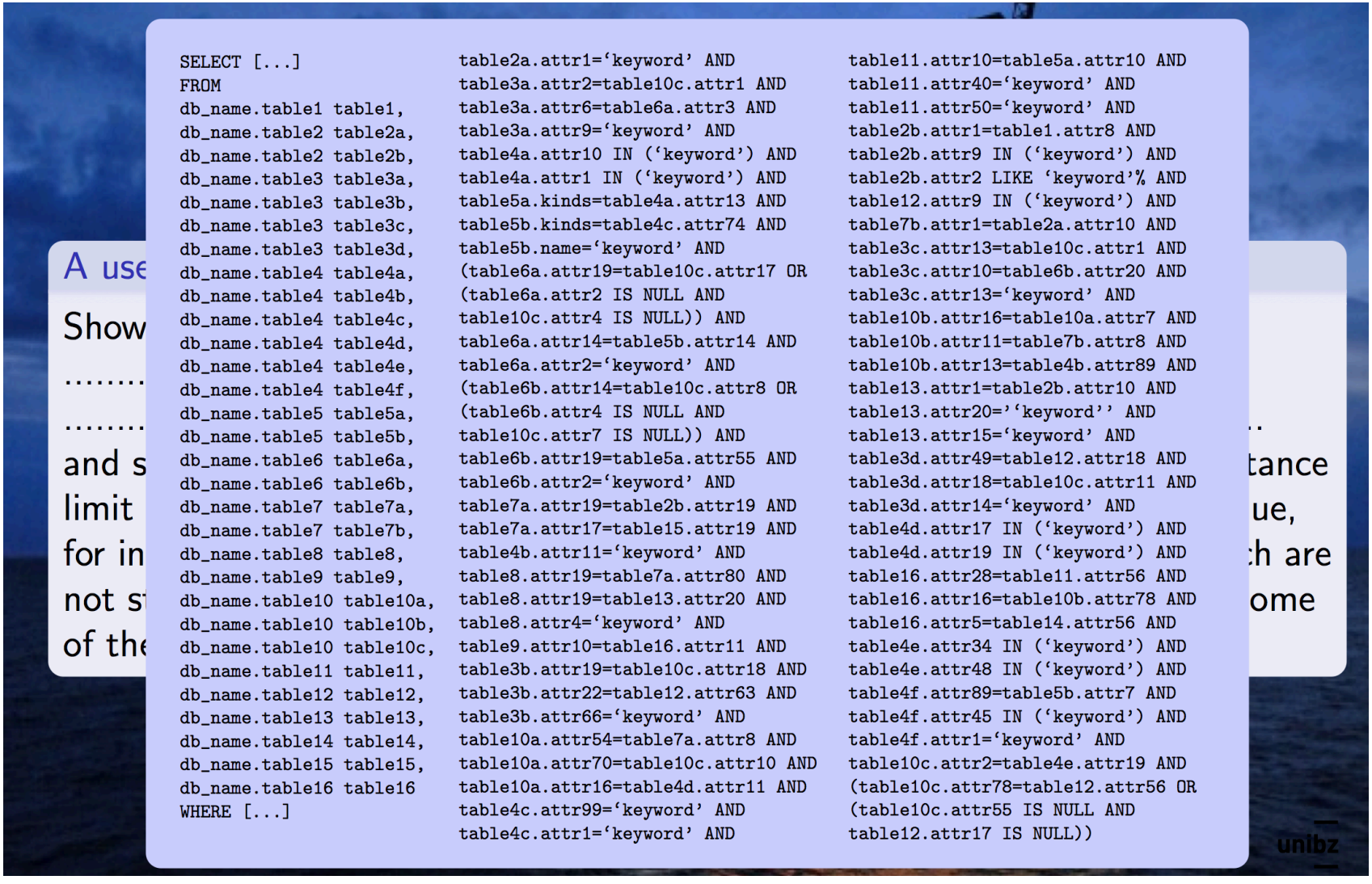
A user query at Statoil

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.

unibz

[G. Xiao, 2015]

How much time/money is spent searching for data ?



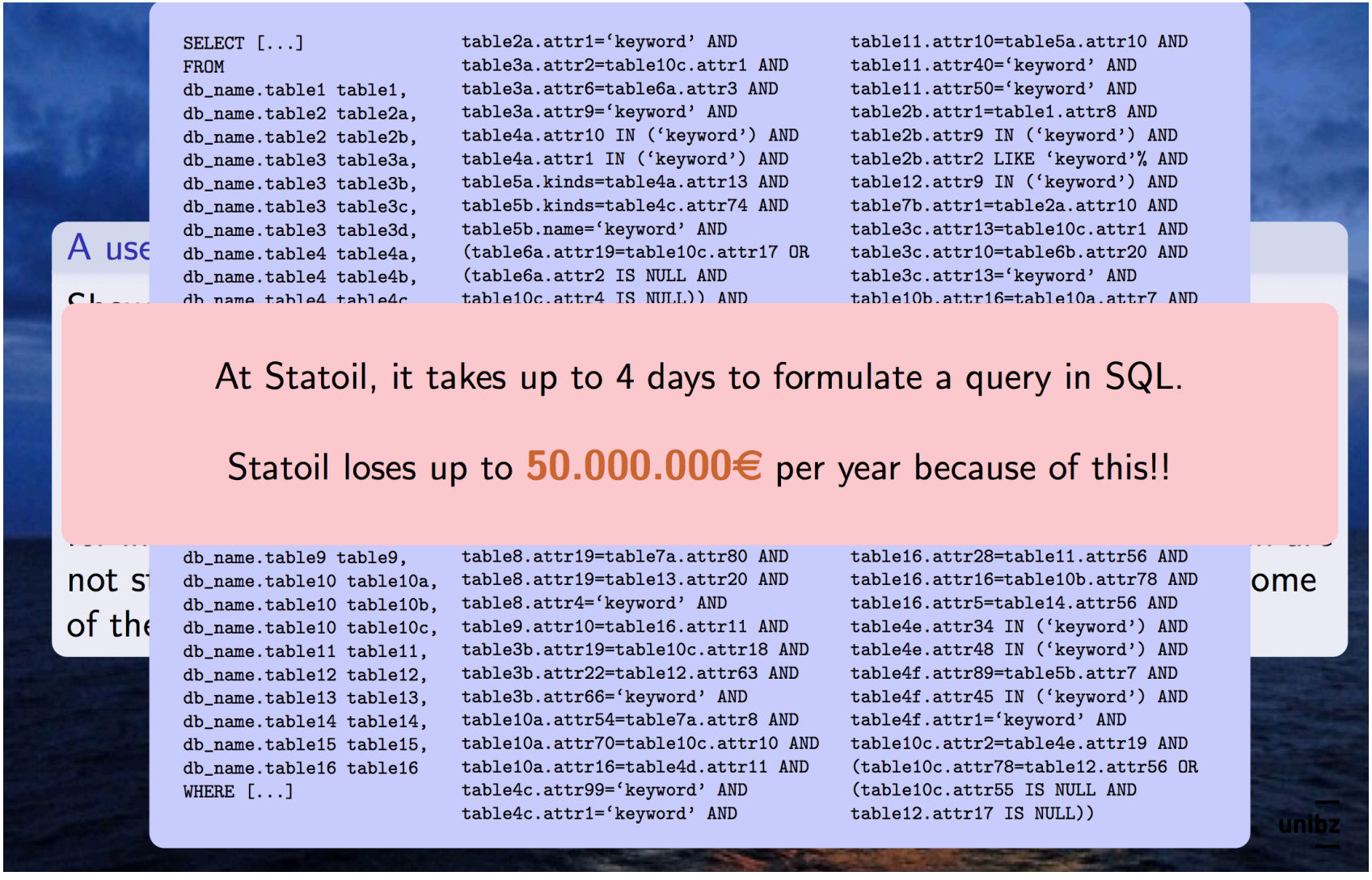
```
SELECT [...]
FROM
db_name.table1 table1,
db_name.table2 table2a,
db_name.table2 table2b,
db_name.table3 table3a,
db_name.table3 table3b,
db_name.table3 table3c,
db_name.table3 table3d,
db_name.table4 table4a,
db_name.table4 table4b,
db_name.table4 table4c,
db_name.table4 table4d,
db_name.table4 table4e,
db_name.table4 table4f,
db_name.table5 table5a,
db_name.table5 table5b,
db_name.table6 table6a,
db_name.table6 table6b,
db_name.table7 table7a,
db_name.table7 table7b,
db_name.table8 table8,
db_name.table9 table9,
db_name.table10 table10a,
db_name.table10 table10b,
db_name.table10 table10c,
db_name.table11 table11,
db_name.table12 table12,
db_name.table13 table13,
db_name.table14 table14,
db_name.table15 table15,
db_name.table16 table16
WHERE [...]
```

```
table2a.attr1='keyword' AND
table3a.attr2=table10c.attr1 AND
table3a.attr6=table6a.attr3 AND
table3a.attr9='keyword' AND
table4a.attr10 IN ('keyword') AND
table4a.attr1 IN ('keyword') AND
table5a.kinds=table4a.attr13 AND
table5b.kinds=table4c.attr74 AND
table5b.name='keyword' AND
(table6a.attr19=table10c.attr17 OR
(table6a.attr2 IS NULL AND
table10c.attr4 IS NULL)) AND
table6a.attr14=table5b.attr14 AND
table6a.attr2='keyword' AND
(table6b.attr14=table10c.attr8 OR
(table6b.attr4 IS NULL AND
table10c.attr7 IS NULL)) AND
table6b.attr19=table5a.attr55 AND
table6b.attr2='keyword' AND
table7a.attr19=table2b.attr19 AND
table7a.attr17=table15.attr19 AND
table4b.attr11='keyword' AND
table8.attr19=table7a.attr80 AND
table8.attr19=table13.attr20 AND
table8.attr4='keyword' AND
table9.attr10=table16.attr11 AND
table3b.attr19=table10c.attr18 AND
table3b.attr22=table12.attr63 AND
table3b.attr66='keyword' AND
table10a.attr54=table7a.attr8 AND
table10a.attr70=table10c.attr10 AND
table10a.attr16=table4d.attr11 AND
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
table2b.attr2 LIKE '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))
```

[G. Xiao, 2015]

Challenges Accessing Big Data



```
SELECT [...]
FROM
db_name.table1 table1,
db_name.table2 table2a,
db_name.table2 table2b,
db_name.table3 table3a,
db_name.table3 table3b,
db_name.table3 table3c,
db_name.table3 table3d,
db_name.table4 table4a,
db_name.table4 table4b,
db_name.table4 table4c,
db_name.table9 table9,
db_name.table10 table10a,
db_name.table10 table10b,
db_name.table10 table10c,
db_name.table11 table11,
db_name.table12 table12,
db_name.table13 table13,
db_name.table14 table14,
db_name.table15 table15,
db_name.table16 table16
WHERE [...]
```

```
table2a.attr1='keyword' AND
table3a.attr2=table10c.attr1 AND
table3a.attr6=table6a.attr3 AND
table3a.attr9='keyword' AND
table4a.attr10 IN ('keyword') AND
table4a.attr1 IN ('keyword') AND
table5a.kinds=table4a.attr13 AND
table5b.kinds=table4c.attr74 AND
table5b.name='keyword' AND
(table6a.attr19=table10c.attr17 OR
(table6a.attr2 IS NULL AND
table10c.attr4 IS NULL)) AND
table8.attr19=table7a.attr80 AND
table8.attr19=table13.attr20 AND
table8.attr4='keyword' AND
table9.attr10=table16.attr11 AND
table3b.attr19=table10c.attr18 AND
table3b.attr22=table12.attr63 AND
table3b.attr66='keyword' AND
table10a.attr54=table7a.attr8 AND
table10a.attr70=table10c.attr10 AND
table10a.attr16=table4d.attr11 AND
table4c.attr99='keyword' AND
table4c.attr1='keyword' AND
```

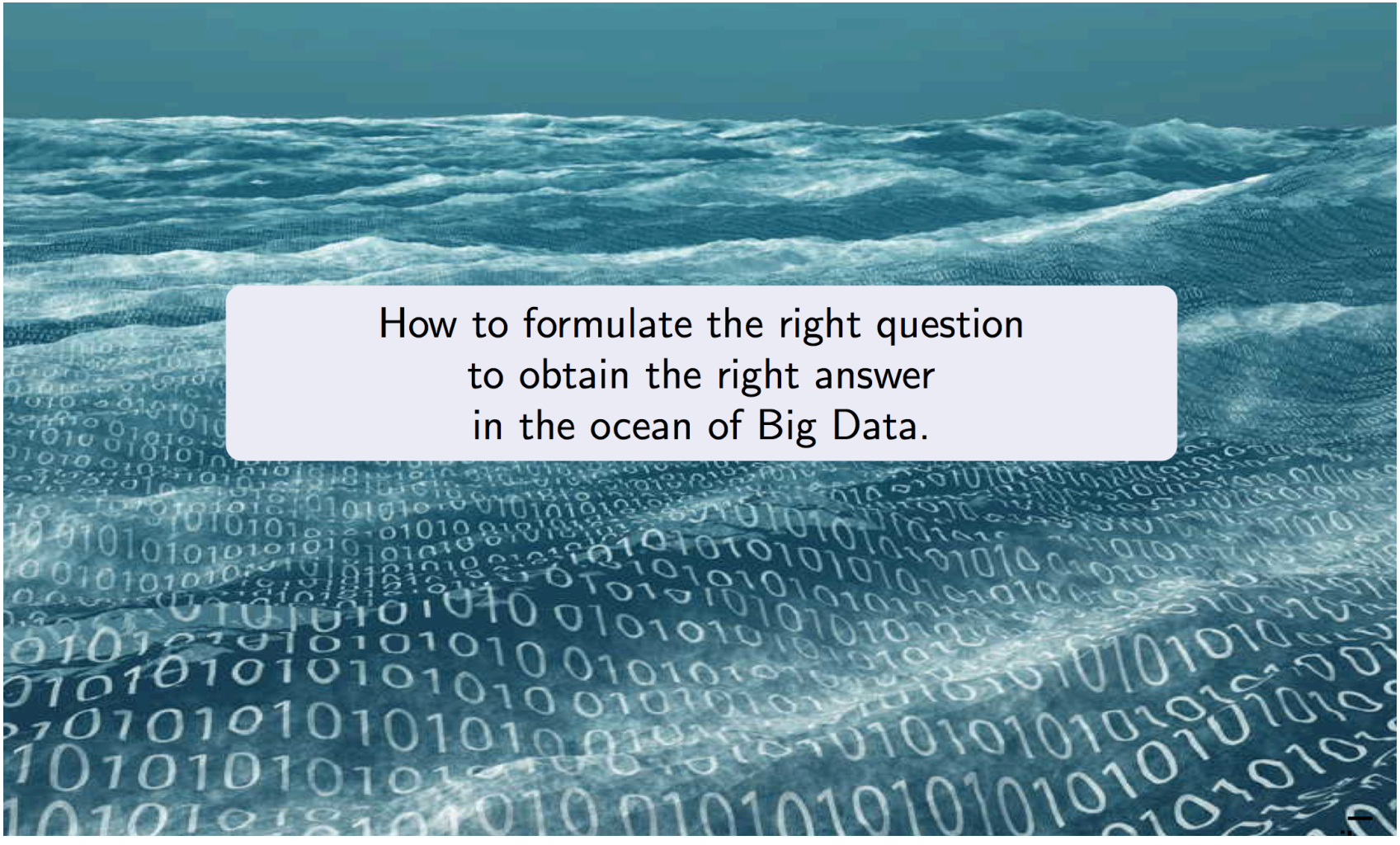
```
table11.attr10=table5a.attr10 AND
table11.attr40='keyword' AND
table11.attr50='keyword' AND
table2b.attr1=table1.attr8 AND
table2b.attr9 IN ('keyword') AND
table2b.attr2 LIKE 'keyword'% AND
table12.attr9 IN ('keyword') AND
table7b.attr1=table2a.attr10 AND
table3c.attr13=table10c.attr1 AND
table3c.attr10=table6b.attr20 AND
table3c.attr13='keyword' AND
table10b.attr16=table10a.attr7 AND
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))
```

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!!

[G. Xiao, 2015]

The Problem : information access



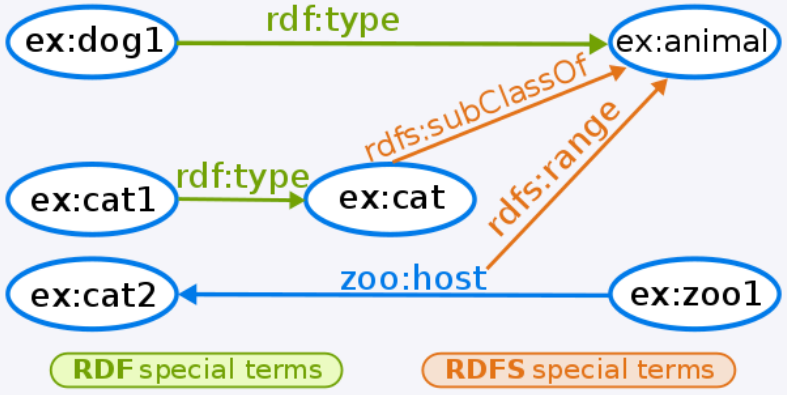
[G. Xiao, 2015]

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
• Dog1 is an animal • Cat1 is a cat • Cats are animals • Zoos host animals • Zoo1 hosts the Cat2	

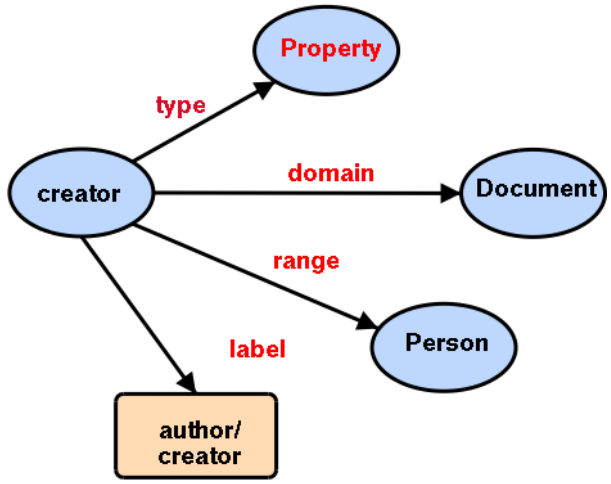
RDF/turtle

```
@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:zoo1    zoo:host      ex:cat2 .
```

RDFS Vocabulary vs. implicite information

Unlike sparql query which only matches graph patterns, RDFS vocabularies can help to deduce implicite information.

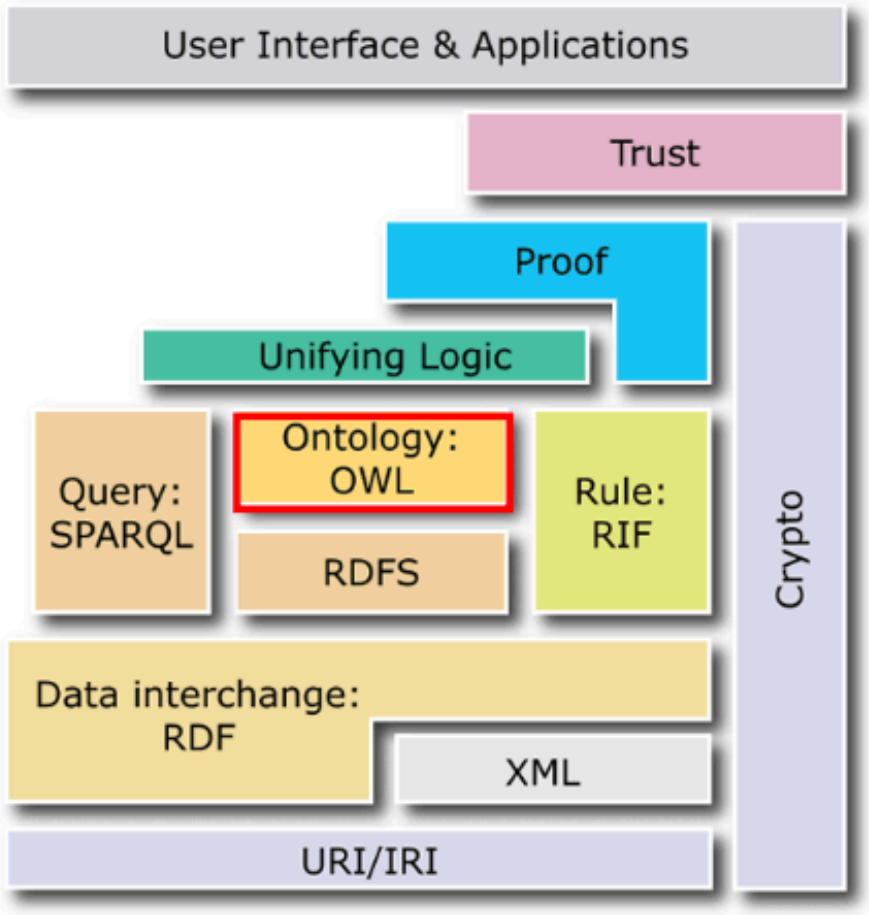
$\rightsquigarrow$ Metadata reasoning !



@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” .

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

Ontology : a modern name for knowledge bases

Applications :

- semantic web enable a common understanding of notions for semantic labeling of Web content
- medicine formal definitions that can be used by doctors, patients, insurance companies, etc, to communicate with each other

OWL (Ontology Web Language) ... and DLs

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

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

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

$\{John\} \sqcap \{Bill\} \sqsubseteq \bot$

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

$\{John\} \equiv \{Jim\}$

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

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

$\neg hasWife(Bill,Mary)$

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

Inconsistent information ???

:John owl :differentFrom :Bill .
:John owl :sameAs :Bill .

$\rightsquigarrow$ Inconsistency !

:John :hasWife :Mary .
$\neg hasWife(John, Mary)$.

$\rightsquigarrow$ Inconsistency !

OWL (Ontology Web Language) ... and DLs

```
:Woman rdfs:subClassOf :Person .
```

```
:Person owl:equivalentClass :Human .
```

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

$\mathbf{Woman} \sqcap \mathbf{Man} \sqsubseteq \bot$

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

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

Incoherent information ???

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

$\rightsquigarrow$:x is an incoherent concept !

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

OWL (Ontology Web Language) vs. FOL vs. DLs

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

Mother ≡ Woman ⊓ Parent

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

Parent ≡ Mother ⊔ Father

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

ChildlessPerson ≡ Person ⊓ ¬Parent

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

OWL (Ontology Web Language) vs. FOL vs. DLs

Person ⊓ ¬Parent (Jack)

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

OWL (Ontology Web Language) vs. FOL vs. DLs

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

Parent ≡ ∃hasChild.Person

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

Orphan ≡ ∀ hasChild⁻.Dead

OWL (Ontology Web Language) vs. FOL vs. DLs

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

JohnsChildren ≡ ∃hasParent.{John}

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

NarcisticPerson ≡ ∃loves.Self

OWL (Ontology Web Language) vs. FOL vs. DLs

≤4 hasChild.Parent (John)

```
: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
] .
```

≥2 hasChild.Parent (John)

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

=3 hasChild.Parent (John)

OWL (Ontology Web Language) vs. FOL vs. DLs

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

=5 hasChild.⊤ (John)

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

MyBirthdayGuests ≡ {Bill, John, Mary}

OWL (Ontology Web Language) vs. FOL vs. 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) vs. FOL vs. DLs

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

$$\textbf{hasParent} \circ \textbf{hasParent} \sqsubseteq \textbf{hasGrandParent}$$

```
: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) vs. FOL vs. DLs

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

Datatype *facets*

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

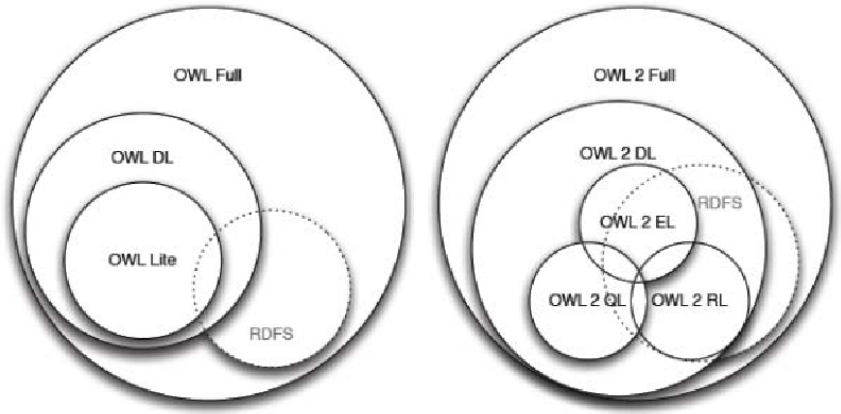
OWL (Ontology Web Language) vs. FOL vs. DLs

Feature	Related OWL vocabulary	FOL	DL
top/bottom class	`owl:Thing/owl:Nothing`	(axiomatise)	$\top/\bot$
Class intersection	`owl:intersectionOf`	$\wedge$	$\sqcap$
Class union	`owl:unionOf`	$\vee$	$\sqcup$
Class complement	`owl:complementOf`	$\neg$	$\neg$
Enumerated class	`owl:oneOf`	(ax. with $\approx$)	{a}
Property restrictions	`owl:onProperty`		
Existential	`owl:someValueFrom`	$\exists y \ldots$	$\exists R.C$
Universal	`owl:allValuesFrom`	$\forall y \ldots$	$\forall R.C$
Min. cardinality	`owl:minQualifiedCardinality` `owl:onClass`	$\exists y1 \ldots yn. \ldots$	$\geq n\ S.C$
Max. cardinality	`owl:maxQualifiedCardinality` `owl:onClass`	$\forall y1 \ldots yn+1. \ldots \rightarrow \ldots$	$\leq n\ S.C$
Local reflexivity	`owl:hasSelf`	R(x,x)	$\exists R.Self$

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