



UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH

School of Professional & Executive Development

Linked Data and Semantic Modeling

Oscar Romero

Barcelona; February 1st, 2017

Semantic Modeling: Why

- New Challenges
 - Data exchange and exploitation
 - Semantics (metadata) as first-class citizen
 - Linked data (i.e., with semantic *pointers*)
 - Flexible and generic means to represent data
 - Semi-structured data models
- **Golden Rule in Data Management:**
“The way you store data will impact on how this data can be managed and exploited”
 - *Load First, Model Later*

Linked Data (The Idea)

- From the World Wide Web (WWW) to the Semantic Web
 - Published data that...
 - Is machine-readable,
 - Its meaning is explicitly defined,
 - Its linked to other external data sets,
 - And can be linked to from other external data sets
- Tim Berners-Lee outlined a set of rules:
 - Use URIs as names for things
 - Universal identifiers to represent real-world objects
 - Use HTTP URIs so that people can look up those names
 - Universally available (where to locate it)
 - When someone looks up a URI, provide useful information, using RDF and SPARQL
 - Description of the object features or characteristics
 - Include links to other URIs, so they can discover more things
 - Relationships as first-class citizens (information integration)

Linked Data (How To)

- As it was described, Linked Data sits at the conceptual level
 - Lack of standards!
- Linked Data So Far (At the logical level)
 - W3C Standards for *sharing meaning*
 - RDF (Resource Description Framework)
 - Subject predicate object
 - OWL / Ontologies
 - Common conceptualizations
 - URIs (*global identifiers*)
 - URN (Universal Resource Name) ~ id
 - URL (Universal Resource Location) ~ *where* to locate it
 - Dereference (HTTP protocol)

Publishing Linked Data

- Three basic steps:
 - Assign URIs and dereferencing capabilities
 - Set RDF links to other data sources
 - Provide metadata about published data
- Choosing URIs and Vocabularies
 - URI aliases
 - Different perspectives of the same entity
 - Whenever possible, use well-known RDF vocabularies
 - Alternatively, use your own vocabulary
 - Self-describe it
 - Map it to other vocabularies
- Link Generation
 - Automatically or semi-automatically
 - Shared name (common vocabularies)
 - Based on similarities (Record linkage community)
 - Duplicate detection (database community)
 - Ontology matching (knowledge representation community)
- Metadata
 - Quality issues (provenance, profiling, etc.)

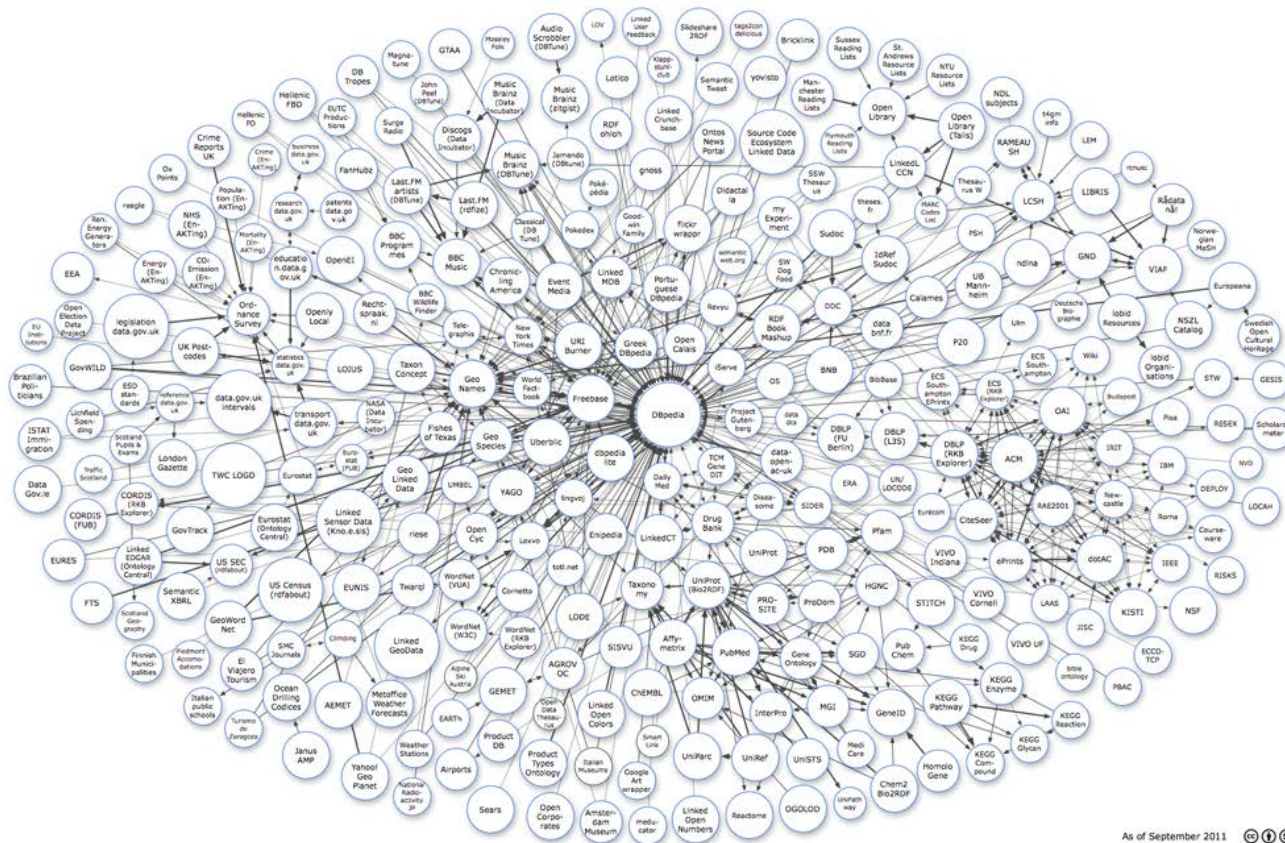
Challenges

- Data exposure
 - Viral uptake: RDF as *de facto* data model
 - Principle of *least power*
 - Appropriate architectures (platform level)
- Reuse
 - Reference conceptualizations
 - Ontology development and management
 - Community and practice-based
 - Schema mapping and data integration (aka data fusion)
 - Quality issues
 - Link maintenance
- Visualization and user interfaces
- Privacy
- An encompassed approach is missing
 - *Ladder of authority* (registries) - Trust
 - Licenses

Web of Data

- Architecture of the Web based on Linked Data, which has the following properties:
 - The Web of Data is generic
 - Anyone can publish
 - Data publishers can choose the vocabulary with which represent their data
 - Entities are connected by means of RDF links
- Consequences:
 - Data is separated from formatting
 - Data is self-describing
 - Dereferencing to deal with unknown vocabularies
 - HTTP as standard to locate entities
 - RDF as data model
 - By definition, it is **open**
 - Reuse, remix, revise, redistribute

The Linking Open Data Project



As of September 2011



“Linking Open Data cloud diagram, by Richard Cyganiak and Anja Jentzsch. <http://lod-cloud.net/>”

The Semantic Web Data Models

- Conceptual Data Models
 - Ontologies **LINKED DATA**
- Logical Data Models
 - XML **DATA SILOS**
 - JSON **DATA SILOS**
 - RDF **LINKED DATA**

Ontology Languages

RDF AND RDFS

RDF

- RDF: Resource description format
 - Resources, literals and properties
 - The basic RDF block is the ***triple***: a binary relationship between two resources or between a resource and a literal
 - *<Subject predicate object>*
 - Subject S has value *object* O for predicate P
 - Subject and predicate must be URIs
 - Object can be a URI or a literal (i.e., a constant value)
 - It accepts blank nodes (always preceded by :_)
 - The resulting metadata can be seen as a graph
 - rdf: A namespace for RDF
 - The URI is: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
- It is a simple language for describing *annotations* (facts) about Web resources identified by URIs
 - The most basic ontology language
 - Triples map to FOL as grounded atomic formulas (subject and object are constants)
 - Blank nodes map to existential variables

RDF: The Big Picture

- Semantics

<http://www.w3.org/TR/rdf11-primer/>

- **Triples** (i.e., RDF statements),
– and **semantic graphs** (aka RDF graphs)

- Syntax

- **XML-based serialization** (aka RDF syntax)
 - The way to express RDF triples in a machine-processable format

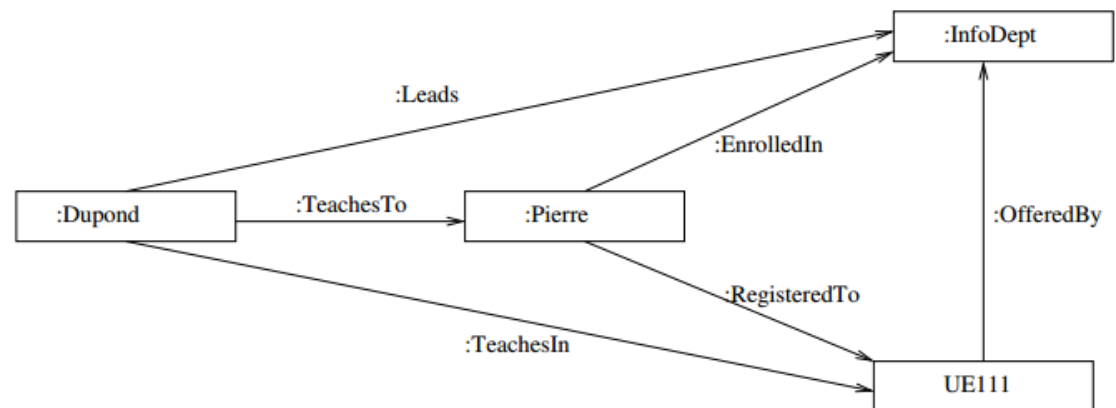
RDF Example (I)

- RDF triplets or statements:

```
{ :Dupond :Leads :CSDept }  
{ :Dupond :TeachesIn :UE111 }  
{ :Dupond :TeachesTo :Pierre }  
{ :Pierre :EnrolledIn :CSDept }  
{ :Pierre :RegisteredTo :UE111 }  
{ :UE111 :OfferedBy :CSDept }
```

Subject	Predicate	Object
:Dupond	:Leads	:CSDept
:Dupond	:TeachesIn	:UE111
:Dupond	:TeachesTo	:Pierre
:Pierre	:EnrolledIn	:CSDept
:Pierre	:RegisteredTo	:UE111
:UE111	:OfferedBy	:CSDept

- RDF graph:



RDF Example (II)

- RDF is typically serialized as XML (syntax)
- Example:

```
<?xml version="1.0"?>

<rdf:RDF
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:cd="http://www.recshop.fake/cd#">

  <rdf:Description
    rdf:about="http://www.recshop.fake/cd/Empire_Burlesque">
    <cd:artist>Bob Dylan</cd:artist>
    <cd:country>USA</cd:country>
    <cd:company>Columbia</cd:company>
    <cd:price>10.90</cd:price>
    <cd:year>1985</cd:year>
  </rdf:Description>
```

- Other RDF Syntaxes:
 - Turtle (human-readable)
 - N-triples
 - Notation 3
 - ...

Turtle:

```
(...)
<#empire-burlesque>
  cd:artist  <#Bob-Dylan> ;
  cd:country <#USA>      ;
  cd:price   10.90       .
(...)
```

Most Popular RDF Vocabularies

- Some organizations, groups, etc. created some RDF vocabularies (i.e., RDF graphs), which are available on-line (RDF namespaces) Some examples:
 - Simple knowledge organization system (SKOS)
<http://www.w3.org/TR/skos-reference/>
 - Generic purpose (mainly schema)
 - Friend-of-a-friend (FOAF)
<http://xmlns.com/foaf/spec/>
 - Statistical Data and Metadata Exchange (SDMX)
<http://sdmx.org/>
 - The RDF Data Cube Vocabulary (QB)
<http://www.w3.org/TR/vocab-data-cube/>
 - A Vocabulary for BI on the Web (QB4OLAP)
<http://publishing-multidimensional-data.googlecode.com/git-history/6db60dff91cf4571432f6bf0b31b339579d63795/index.html>

Activity: Modeling in RDF

- *Objective: Grasp the idea behind RDF modeling*
- (10') Model a RDF Graph capturing data about lecturers, courses and students:
 - *A student enrolls several courses from his faculty per semester. He is forced to enroll, at least, one course per semester. Each course has one responsible lecturer but potentially, it might have several lecturers.*
- (Home) As an alternative activity, write the triples you created in XML-RDF syntax
 - Also try Turtle

Basics on RDF Modeling

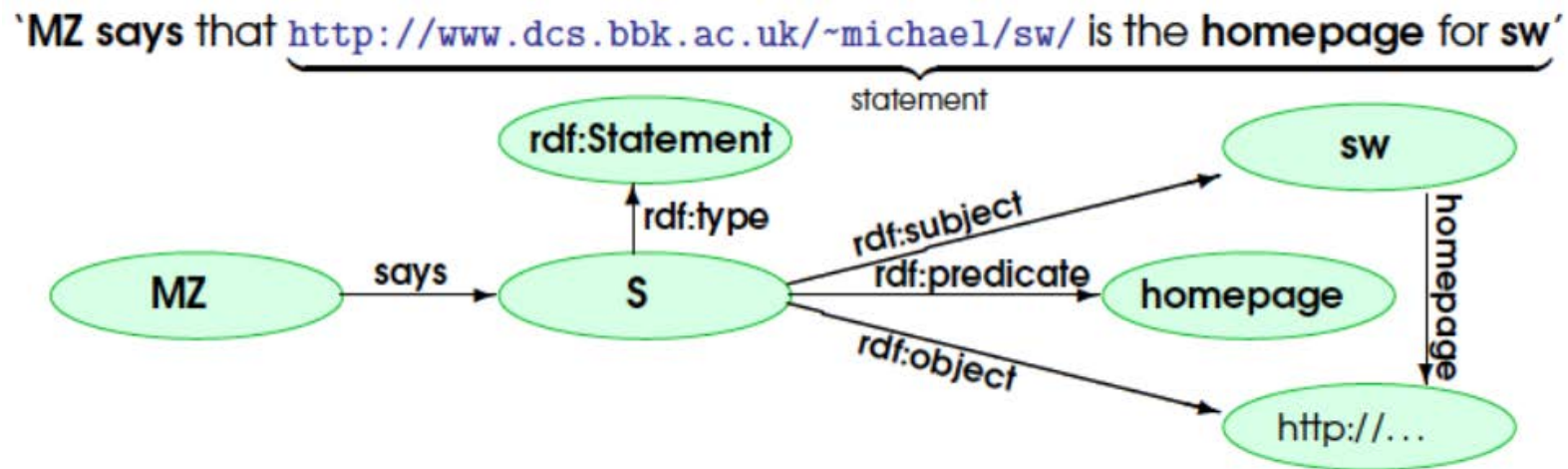
- RDF modeling is based on binary relationships
 - Similar problems as in the relational model
 - Blank nodes as solution (~reification)
- We cannot express neither schema nor constraints in RDF
 - Example: *at least one, at most three, the domain of a property must be of type X, etc.*

Blank Nodes

- Blank nodes do not have a URI and cannot be referenced
 - They can only be subjects or objects
- Its semantics are yet not clear, though
 - De facto use (i.e., most spread use, also in SPARQL): An **identifier** without a URI
 - Basically, to implement n-ary relationships by *reification*
 - W3C position: Incomplete data (potentially two blank nodes might be the same resource)
 - An unknown value,
 - A value that does not apply / anonymized value
- The de facto use is a pragmatic use
 - Facilitates reasoning (CWA)
- The Linked Open Data community discourages its use. Everything should have a URI!

Blank Nodes: Reification (I)

- Example of use (“quoting”):



- In this example, what is schema and what instances?

Blank Nodes: Reification (II)

- Example of use (container):

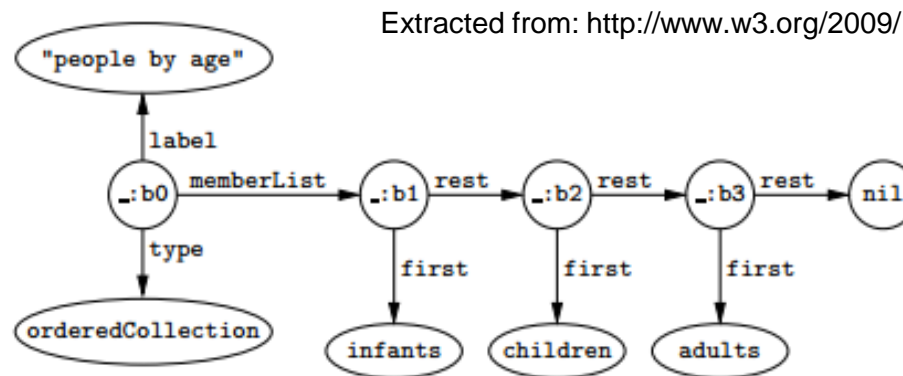


Fig. 2. An ordered collection in SKOS.

- These statements can be written as:
 - [*property object*] (e.g., [*label “people by age”*])
 - The subject is considered to be a blank node

RDF Schema (RDFS)

- Constraints on RDF statements must be stated in RDFS or OWL
- RDFS allows to specify constraints on individual and relationships.
More specifically:
 - Declare objects and subjects as instances of certain classes
 - E.g., *Oscar isa lecturer*
 - Inclusion statements between classes and between properties
 - E.g., lecturer *isa* human **TAXONOMIES!**
 - Relate the domain and the range of a property to some classes
 - E.g., the predicate fatherOf relates a subject and an object that must be instances of the class human
- RDFS extends RDF not only to consider *instances* but schema
 - The schema can be described with the same description
 - rdfs: A namespace for RDFS
 - The URI is: <http://www.w3.org/2000/01/rdf-schema#>

RDFS Example

- Instances:
 - :Oscar rdf:type :Lecturer
- Taxonomies:
 - (classes) :Lecturer rdfs:subclassOf :Human
 - (rels) :ResponsibleFor rdfs:subpropertyOf :Lectures
- Domain and Range:
 - :Lectures rdfs:domain :Human
 - :Lectures rdfs:range :Course

Some controversy here:

Some consider rdf:type to be part of RDF, some of RDFS

Reasoning (Inference)

- Non-explicit knowledge inferred from explicitly asserted knowledge
- A knowledge-base consists of schema and instances
 - Schema:
 - :Lecturer rdfs:subClassOf :Human
 - :ResponsibleFor rdfs:subPropertyOf :Lectures
 - :Lectures rdfs:domain :Human
 - :Lectures rdfs:range :Course
 - Instances (explicit facts asserted):
 - :Oscar rdf:type :Lecturer
 - :OpenData rdf:type :Course
 - :Oscar :ResponsibleFor :OpenData
 - Inferred knowledge:
 - :Oscar rdf:type :Human
 - :Oscar :Lectures :OpenData

RDFS Core Classes

- *rdfs:Resource*, the class of all resources (**everything** is a resource)
- *rdfs:Class*, the class of all classes
- *rdfs:Literal*, the class of all literals
- *rdf:Property*, the class of all properties
- *rdf:Statement*, the class of all statements

Example:

```
<rdfs:Class rdf:ID="lecturer">  
  ...  
</rdfs:Class>  
  
<rdf:Property rdf:ID="isTaughtBy">  
  ...  
</rdf:Property>
```


RDFS Core Properties (I)

- *rdf:type*, which relates a resource to its class
`:oscar rdf:type :lecturer`
 - The subject resource (i.e., *:oscar*) is **declared to be an instance** of the object **class** (i.e., *:lecturer*)
- *rdfs:subClassOf*, which relates a class to one of its superclasses
`:lecturer rdfs:subClassOf :human`
- *rdfs:subPropertyOf*, which relates a property to one of its superproperties
`:responsibleFor rdfs:subpropertyOf :lectures`

RDFS Core Properties (II)

- `rdfs:domain`, which specifies the domain of a property

`:lectures rdfs:domain :human`

- The **class** of those resources used as subjects for the predicate `:lectures` must be `:human`

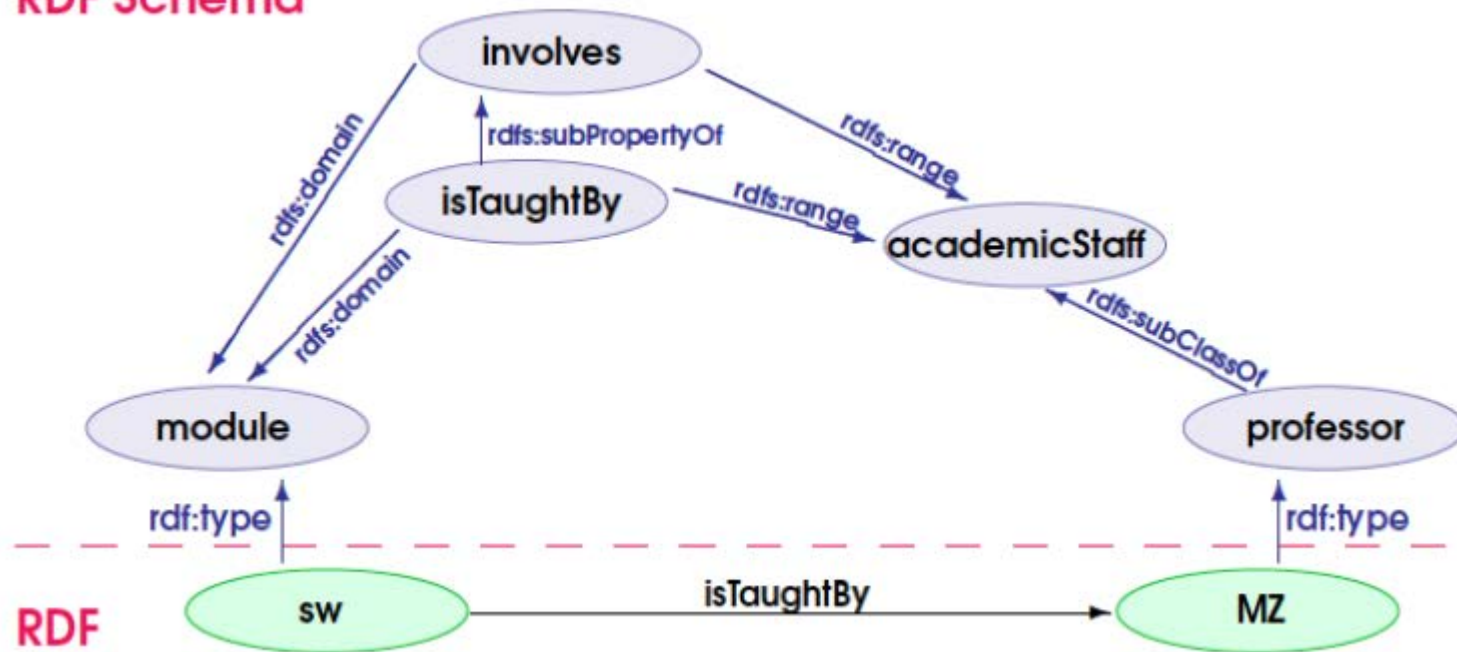
- `rdfs:range`, which specifies the range of a property

`:lectures rdfs:range :course`

- The **class** of those resources used as objects for the predicate `:lectures` must be `:course`

RDFS Semantics

RDF Schema



Next examples are provided by A. Vaisman (ITBA, Argentina)

RDF Semantics: Class and Subclass

- Consider the following inferred knowledge:
 - The object of *rdf:type* is a class
 - Every class is a subclass of *rdfs:Resource*
 - If C_1 is *rdfs:subClassOf* of C_2 then, both C_1 and C_2 are classes. Also:
 - *rdfs:subClassOf* is reflexive on classes
 - *rdfs:subClassOf* is transitive

RDFS Semantics: Property and Subproperty

- Consider the following inferred knowledge:
 - If P_1 is *rdfs:subPropertyOf* of P_2 then, both P_1 and P_2 are classes. Also:
 - *rdfs:subPropertyOf* is reflexive on properties
 - *rdfs:subPropertyOf* is transitive

RDFS Semantics: Domain and Range

- These are **axiomatic** triples
 - Core knowledge to be assumed to be true for any for any profile (reasoning)
 - If $P \text{ rdfs:domain } C$ then C is a class
 - If $P \text{ rdfs:range } C$ then C is a class

SPARQL

- SPARQL: SPARQL Protocol And RDF Query Language
 - Standard query language for RDF Graphs
 - Is a W3C Recommendation
 - It supports RDFS (or OWL) under specific entailments
- Based on pattern matching
 - Simple RDF graphs are used as query patterns

```
Select x,z where x Lectures y, y TaughtIn z, z rdf:type Faculty
```

SPARQL in a Nutshell

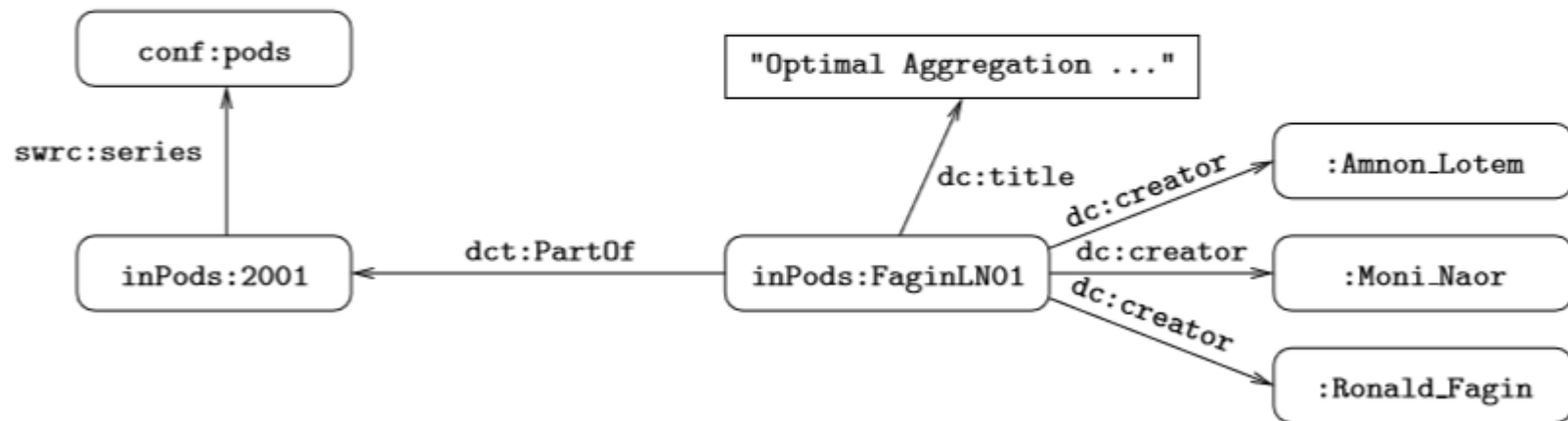
- Each query consists of three parts:
 - Pattern matching (body): optional, union, filtering...
 - Solution modifiers (head): projection, distinct, order, limit...
 - Output part: 4 query forms that retrieve either result sets or RDF graphs
 - SELECT: Returns all, or a subset of, the variables bound in a query pattern match
 - CONSTRUCT: Returns an RDF graph constructed by substituting variables in a set of triple templates
 - ASK: Returns a boolean indicating whether a query pattern matches or not
 - DESCRIBE: Returns an RDF graph that describes the resources found

SPARQL Example

```

: <http://dblp.13s.de/d2r/resource/authors/>
conf: <http://dblp.13s.de/d2r/resource/conferences/>
inPods: <http://dblp.13s.de/d2r/resource/publications/conf/pods/>
swrc: <http://swrc.ontoware.org/ontology#>
dc: <http://purl.org/dc/elements/1.1/>
dct: <http://purl.org/dc/terms/>

```



Give me all authors that have published at PODS

SPARQL Example

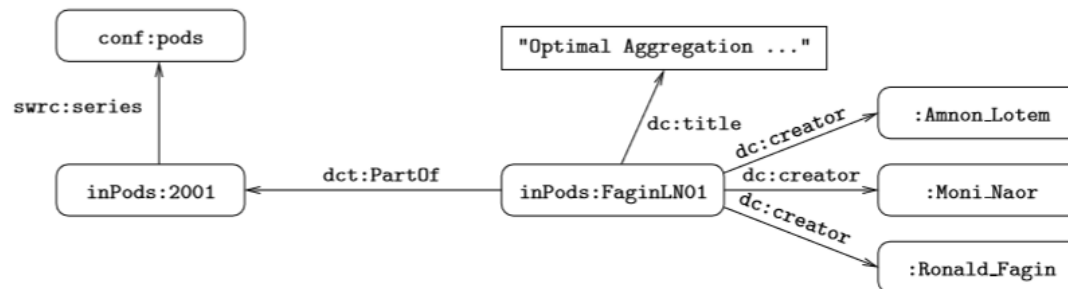
```
SELECT ?Author HEAD
WHERE {
```

```
  ?Paper dc:creator ?Author .
  ?Paper dct:PartOf ?Conf .
  ?Conf swrc:series conf:Pods .
```

BODY

```
}
```

```
  : <http://dblp.13s.de/d2r/resource/authors/>
  conf: <http://dblp.13s.de/d2r/resource/conferences/>
  inPods: <http://dblp.13s.de/d2r/resource/publications/conf/pods/>
  swrc: <http://swrc.ontoware.org/ontology#>
  dc: <http://purl.org/dc/elements/1.1/>
  dct: <http://purl.org/dc/terms/>
```



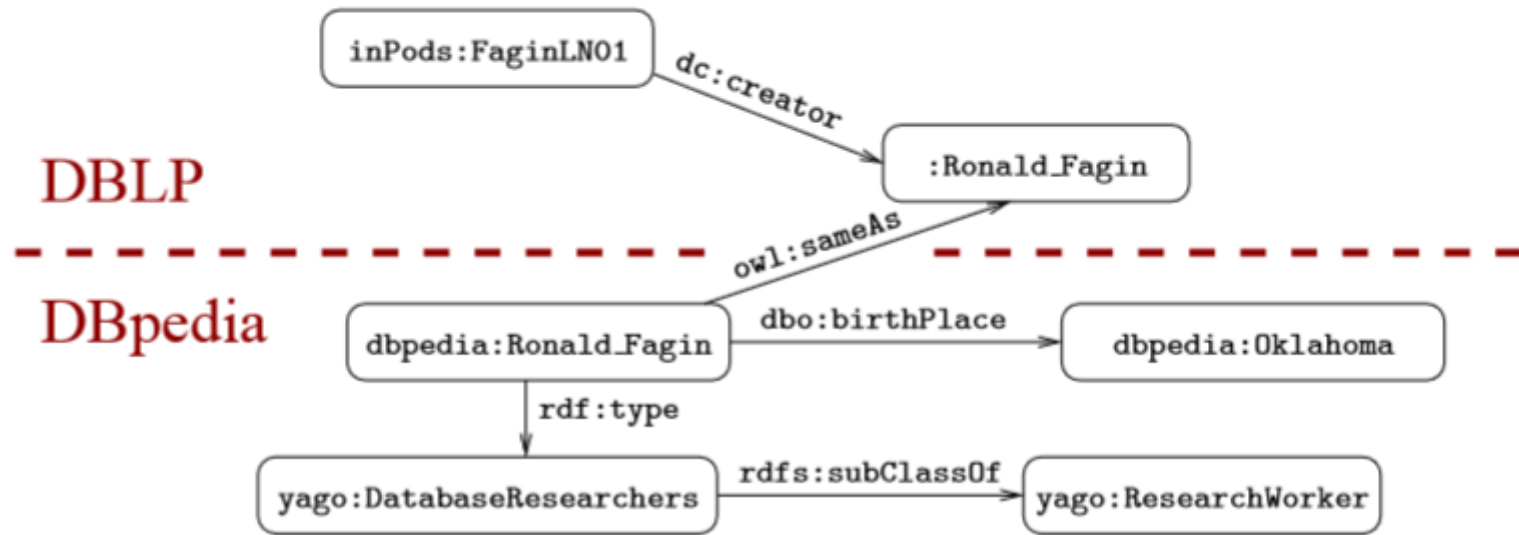
Query: Give me all authors that have published at PODS

SPARQL Features

- It enables us to extract information from interconnected RDF Graphs
- Follows an open-world assumption
 - Highly unstructured data by definition. Thus, the RDF graph will possibly be incomplete
- Should be able to interpret the triples according to predefined semantics (entailments)

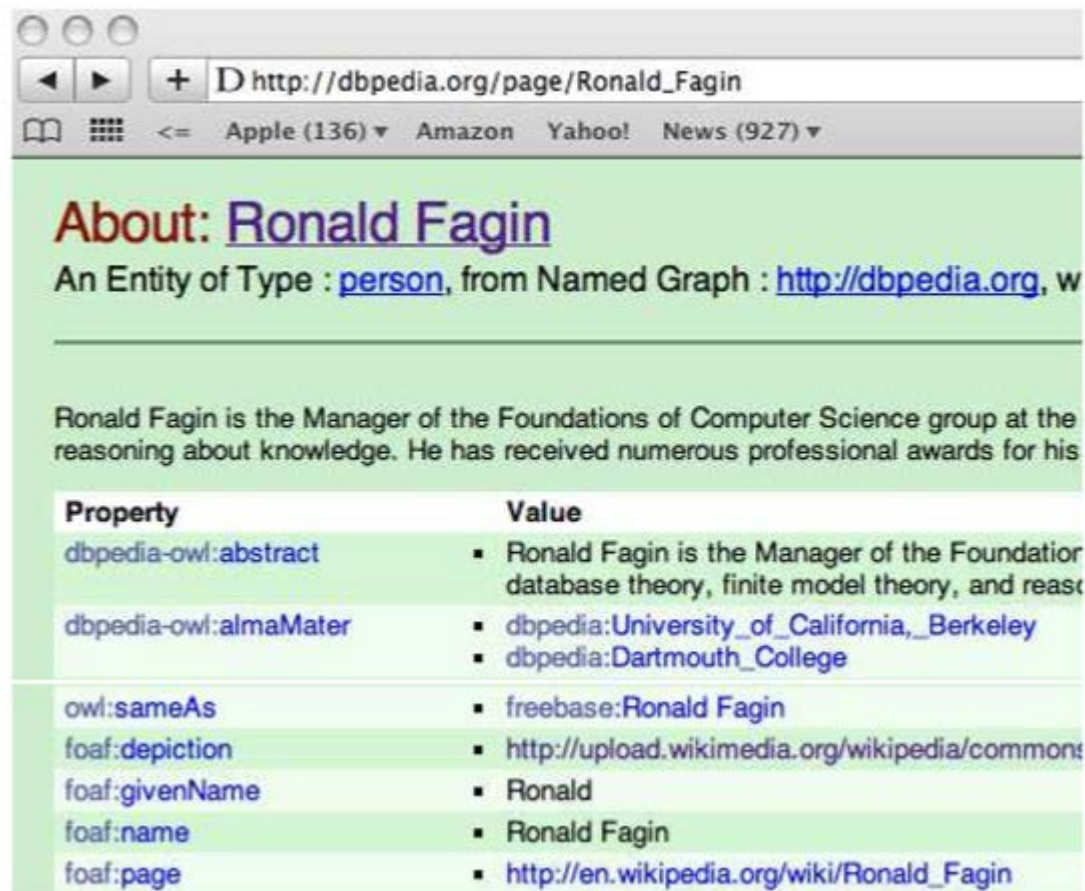
Interconnecting RDF Graphs

```
      : <http://dblp.13s.de/d2r/resource/authors/>  
dbpedia: <http://dbpedia.org/resource/>  
      rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>  
      rdfs: <http://www.w3.org/2000/01/rdf-schema#>  
      owl: <http://www.w3.org/2002/07/owl#>  
yago: <http://dbpedia.org/class/yago>  
dbo: <http://dbpedia.org/ontology/>
```



Interconnecting RDF Graphs

- Dereferenceable URIs are the glue



http://dbpedia.org/page/Ronald_Fagin

Apple (136) Amazon Yahoo! News (927)

About: [Ronald Fagin](#)

An Entity of Type : [person](#), from Named Graph : <http://dbpedia.org>, w

Ronald Fagin is the Manager of the Foundations of Computer Science group at the reasoning about knowledge. He has received numerous professional awards for his

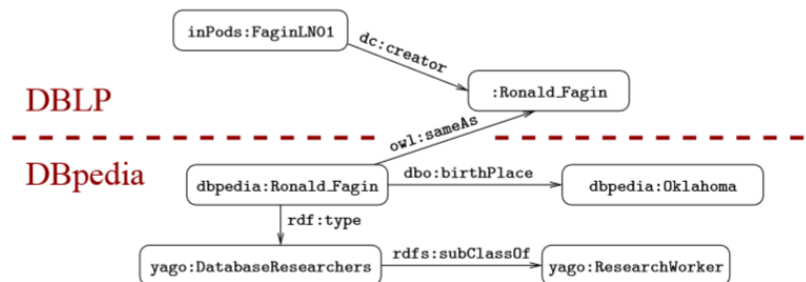
Property	Value
dbpedia-owl:abstract	<ul style="list-style-type: none">Ronald Fagin is the Manager of the Foundation database theory, finite model theory, and reaso
dbpedia-owl:almaMater	<ul style="list-style-type: none">dbpedia:University_of_California,_Berkeleydbpedia:Dartmouth_College
owl:sameAs	<ul style="list-style-type: none">freebase:Ronald Fagin
foaf:depiction	<ul style="list-style-type: none">http://upload.wikimedia.org/wikipedia/commons
foaf:givenName	<ul style="list-style-type: none">Ronald
foaf:name	<ul style="list-style-type: none">Ronald Fagin
foaf:page	<ul style="list-style-type: none">http://en.wikipedia.org/wiki/Ronald_Fagin

Interconnecting RDF Graphs

- Retrieve the authors that have published in PODS and were born in Oklahoma:

```
SELECT ?Author
WHERE {
    ?Paper dc:creator ?Author .
    ?Paper dct:PartOf ?Conf .
    ?Conf swrc:series conf:Pods .
    ?Person owl:sameAs ?Author .
    ?Person dbo:birthPlace dbpedia:Oklahoma .
}
```

```
: <http://dblp.13s.de/d2r/resource/authors/>
dbpedia: <http://dbpedia.org/resource/>
rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
rdfs: <http://www.w3.org/2000/01/rdf-schema#>
owl: <http://www.w3.org/2002/07/owl#>
yago: <http://dbpedia.org/class/yago>
dbo: <http://dbpedia.org/ontology/>
```



Dealing with Uncomplete Data

- Retrieve the authors that have published in PODS, and their Web pages if this information is available

```
SELECT ?Author
WHERE {
    ?Paper dc:creator ?Author .
    ?Paper dct:PartOf ?Conf .
    ?Conf swrc:series conf:Pods .
    ?Person owl:sameAs ?Author .
    ?Person dbo:birthPlace dbpedia:Oklahoma .
    OPTIONAL { ?Author foaf:homePage ?Webpage . }
}
```

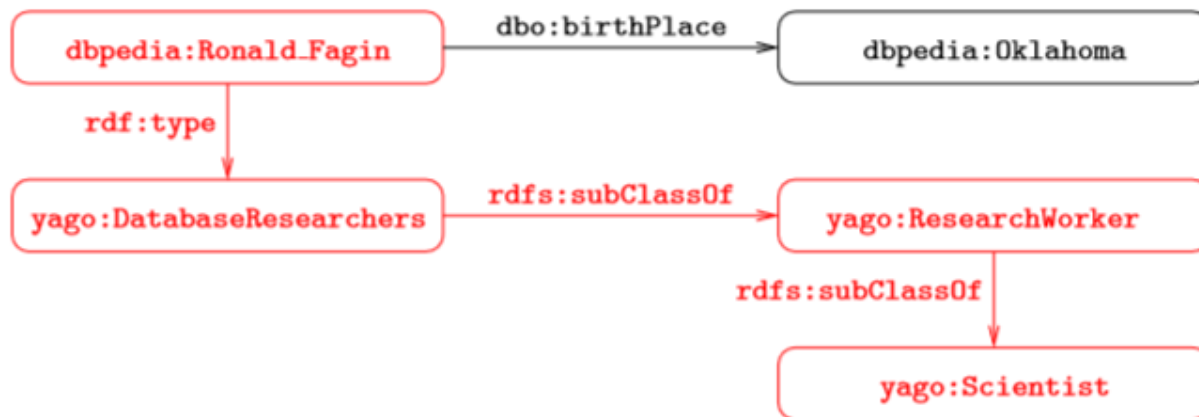
Interpreting Semantics

- Retrieve the scientists that were born in Oklahoma and that have published in PODS

```
SELECT ?Author
WHERE {
    ?Author rdf:type yago:Scientist .
    ?Paper dc:creator ?Author .
    ?Paper dct:PartOf ?Conf .
    ?Conf swrc:series conf:pods .
    ?Person owl:sameAs ?Author .
    ?Person dbo:birthPlace dbpedia:Oklahoma .
}
```


Interpreting Semantics

- Certain knowledge can be inferred from other relationships in the RDF Graph



SPARQL 1.1 - Entailment Regimes

- Simple entailment: evaluation of basic graph pattern by means of pattern matching
- More elaborate entailment relations have been developed
 - To retrieve solutions that *implicitly* follow
- Most popular ones:
 - RDF Schema entailment,
 - OWL 2 RDF-Based Semantics entailment,
 - Etc.

SPARQL Syntax

► Graph pattern:

`?X name ?Y`

$(?X, \text{name}, ?Y)$

`{ P1 . P2 }`

$(P_1 \text{ AND } P_2)$

`{ P1 OPTIONAL { P2 } }`

$(P_1 \text{ OPT } P_2)$

`{ P1 } UNION { P2 }`

$(P_1 \text{ UNION } P_2)$

`{ P1 FILTER (R) }`

$(P_1 \text{ FILTER } R)$

► SPARQL query:

`SELECT ?X ?Y ... { P }`

$(\text{SELECT } \{?X, ?Y, \dots\} P)$

SPARQL Syntax

- Filter Expressions (R)
 - Equality (=) among variables and RDF terms
 - Unary predicate bound
 - Boolean combinations (\wedge , \vee , \neg)
- Example:
 - `(?X = conf: pods)`
 - `\neg (?X = conf: pods)`
 - `(?X = conf: pods) \vee (?Y = conf: sigmod)`
 - `(?X = conf: pods) \wedge \neg (?Y = conf: sigmod)`

SPARQL Syntax

- Explicit precedence/association

```
{  
    t1  
    t2  
    OPTIONAL { t3 }  
    OPTIONAL { t4 }  
    t5  
}
```

- Is translated into:

```
((((t1 AND t2 ) OPT t3 ) OPT t4 ) AND t5 )
```

Activity: Querying with SPARQL

- *Objective: Grasp the idea behind SPARQL*
- (30') Answer the given exercise

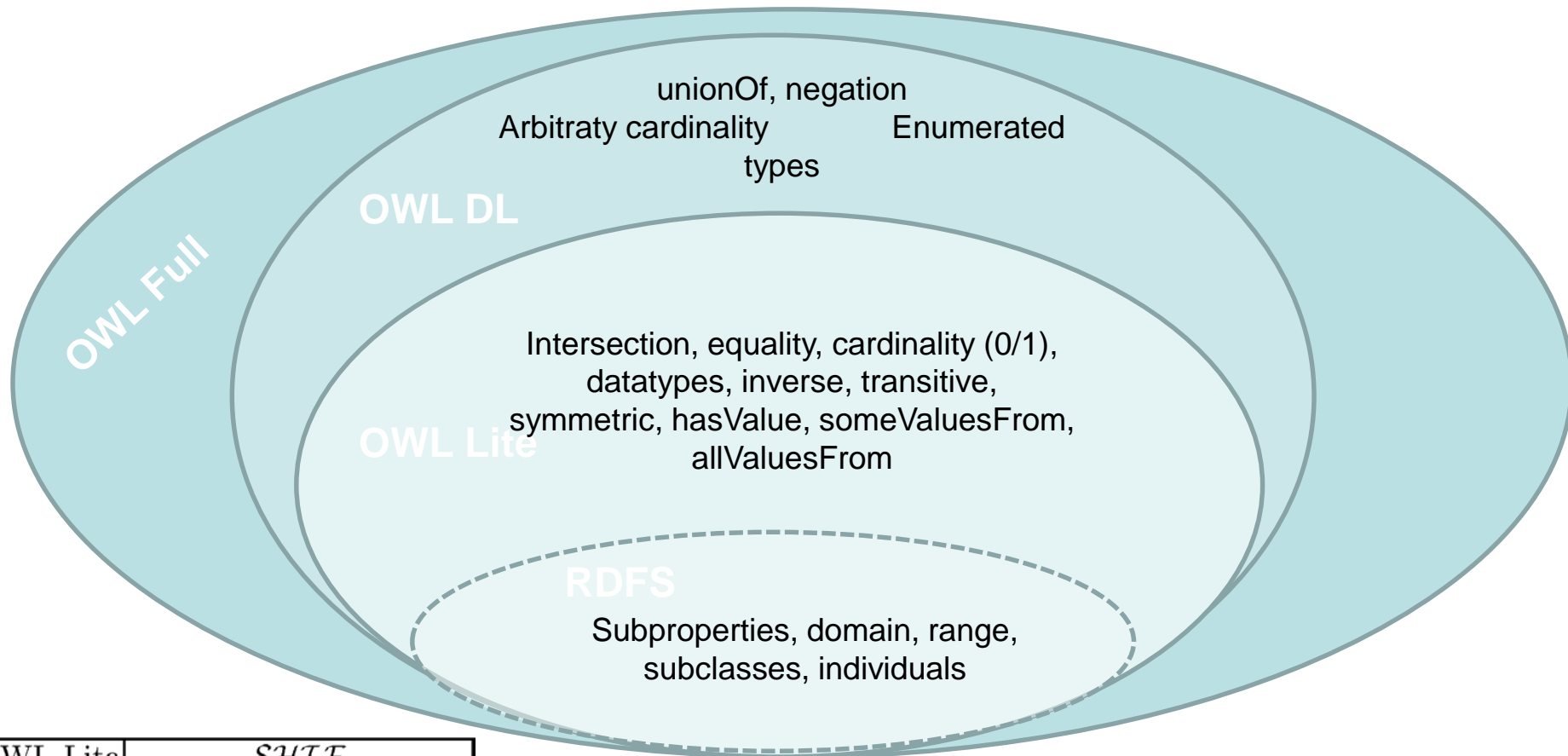
Ontology Languages

OWL

OWL

- W3C recommendation (2003)
- Based on OIL and DAML
- Uses RDF and XML as the underlying representation
- There were three languages in OWL 1.0:
 - Lite
 - DL
 - Full
- OWL 2.0 eliminates OWL Lite and adds three profiles: RL, QL, EL

OWL



OWL-Lite	<i>SHIF</i>
OWL-DL	<i>SHOIN</i>
OWL-Full	Unconstrained <i>SHOIN</i>

Source: Sven Groppe. Data Management and Query Processing in Semantic Web Databases

OWL Versions

- Critics to OWL 1.0
 - Complex reasoning over the ABox (i.e., on data complexity)
 - Integrity constraints are not allowed
 - Limited support to data types
- OWL 2.0
 - Syntactic facilities
 - *SROIQ* constructs
 - OWL 1.0 corresponds to *SHOIN*
 - User defined data types
 - Class attributes and primary keys

Syntax Example (Constructs)

OWL constructor	DL constructor	Example
intersectionOf	$C_1 \sqcap \dots \sqcap C_n$	Human \sqcap Male
unionOf	$C_1 \sqcup \dots \sqcup C_n$	Doctor \sqcup Lawyer
complementOf	$\neg C$	\neg Male
oneOf	$\{a_1\} \sqcup \dots \sqcup \{a_n\}$	{john} \sqcup {mary}
allValuesFrom	$\forall P.C$	\forall hasChild.Doctor
someValuesFrom	$\exists P.C$	\exists hasChild.Lawyer
maxCardinality	$(\leq n P)$	$(\leq 1$ hasChild)
minCardinality	$(\geq n P)$	$(\geq 2$ hasChild)

Syntax Example (Axioms)

OWL axiom	DL syntax	Example
subClassOf	$C_1 \sqsubseteq C_2$	Human \sqsubseteq Animal \sqcap Biped
equivalentClass	$C_1 \equiv C_2$	Man \equiv Human \sqcap Male
disjointWith	$C_1 \sqsubseteq \neg C_2$	Man $\sqsubseteq \neg$ Female
sameIndividualAs	$\{a_1\} \equiv \{a_2\}$	{presBush} \equiv {G.W.Bush}
differentFrom	$\{a_1\} \sqsubseteq \neg\{a_2\}$	{john} $\sqsubseteq \neg\{peter\}$
subPropertyOf	$P_1 \sqsubseteq P_2$	hasDaughter \sqsubseteq hasChild
equivalentProperty	$P_1 \equiv P_2$	hasCost \equiv hasPrice
inverseOf	$P_1 \equiv P_2^-$	hasChild \equiv hasParent ⁻
transitiveProperty	$P^+ \sqsubseteq P$	ancestor ⁺ \sqsubseteq ancestor
functionalProperty	$\top \sqsubseteq (\leq 1 P)$	$\top \sqsubseteq (\leq 1 \text{ hasFather})$
inverseFunctionalProperty	$\top \sqsubseteq (\leq 1 P^-)$	$\top \sqsubseteq (\leq 1 \text{ hasSSN}^-)$

OWL 2 Profiles

OWL 2 EL:

Based on *EL*++

Large number of
properties / classes

Reasoning:
Polynomial with
regard to the
ontology TBOX

OWL 2 QL:

Based on DL-Lite

Captures (most of)
ER and UML
expressive power

Reasoning:
Reducible to
LOGSPACE (i.e.,
DBs)

OWL 2 RL:

Based on Description
Logic programs

Scalable reasoning
without sacrificing
much expressivity

Reasoning:
Polynomial with
regard to the size of
the ontology

Managing Triples

- Storage:
 - Triplestores (e.g., Virtuoso)
 - Graph databases with a semantic layer on top of them (no market products but several academic prototypes)
- Management:
 - Ontology frameworks (e.g., Jena, Sesame, Corese, etc.)

Conclusions

- Semantic modeling tackles several challenges related to Big Data (more specifically, to the *Variety* challenge)
 - Deal with highly unstructured data
 - Similar to graph databases
 - External, not-under-control data semantically enriched to be used by anyone
 - Main difference with graph databases
 - Might facilitate data integration / data crossing
 - If a proper use of the semantics (entailments) is done