Semantic Data

Chapter 7 : The semantic web query language SPARQL

Jean-Louis Binot

Sources

- There are no additional required references for this chapter.
- Sources and useful additional reading :

  The W3C documentation of SPARQL provides complete information on the language :

  - SPARQL 1.0 Query Language for RDF : 2008
  - SPARQL 1.1 Query Language : 2013
  - SPARQL 1.1 Update : 2013

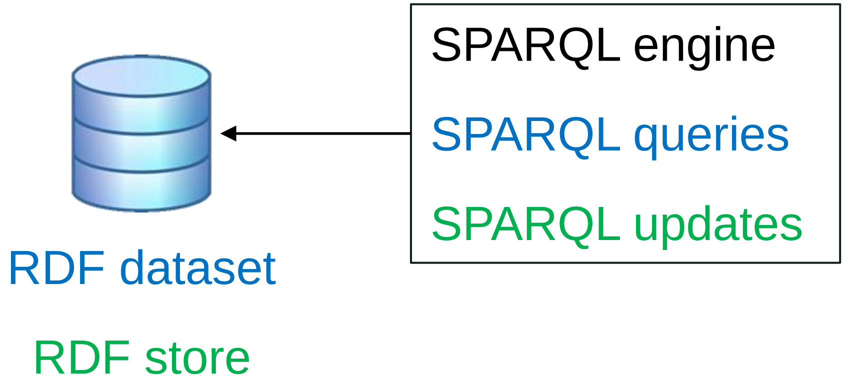
- University courses having partially inspired ideas and examples for this chapter :
  - *Query Language for RDF (SPARQL 1.1), A. Zimmermann, Université Jean Monnet, Saint-Etienne.*
  - *An Introduction to SPARQL and Queries over Linked Data, O. Hartig, ICWE 2012 tutorial.*
  - *Apprentissage symbolique et web sémantique, B. Amman, UPMC.*

Agenda

1	Querying RDF graphs
2	SPARQL syntax and semantics
3	Entailment regimes and OWL

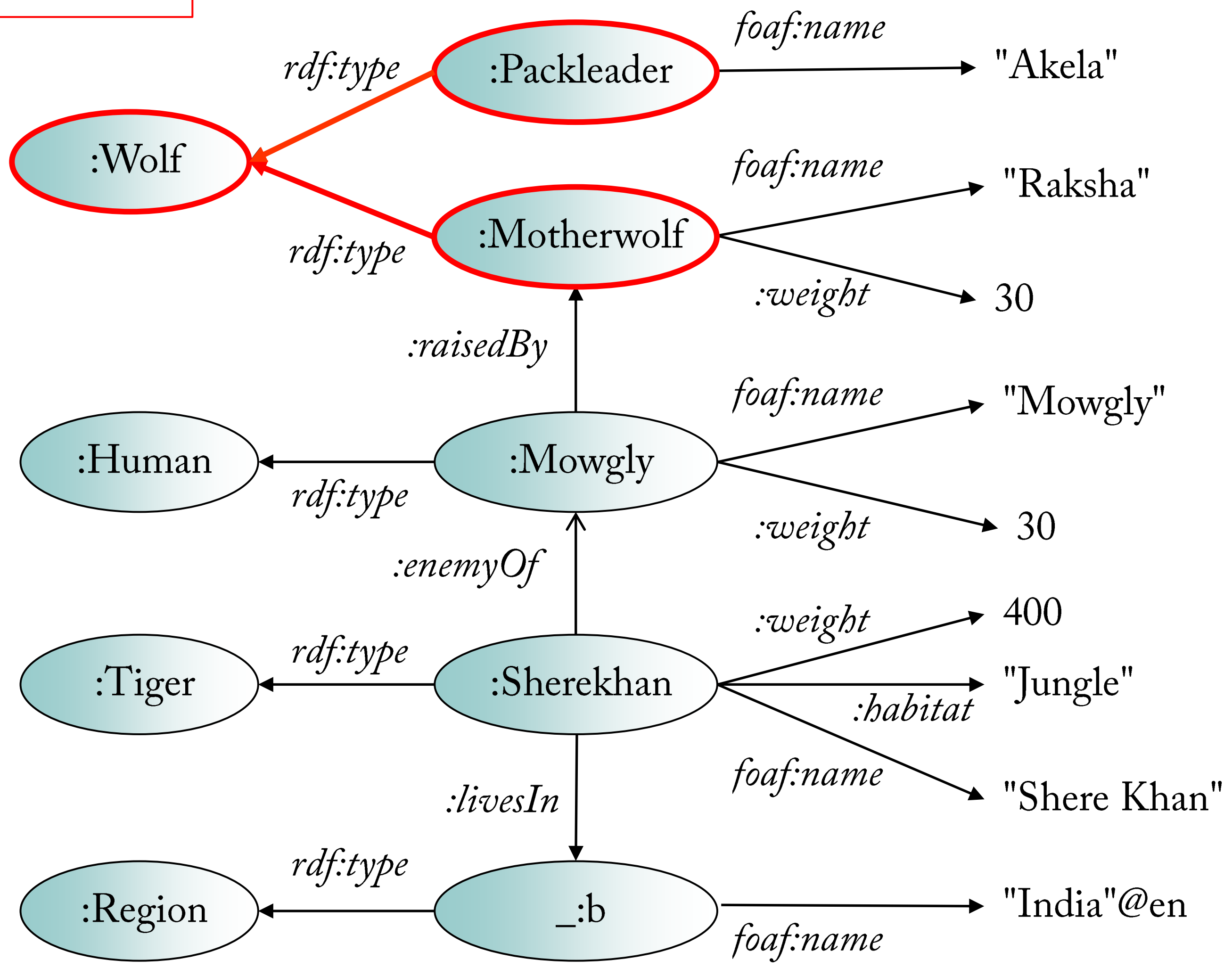
What is SPARQL ?

- SPARQL is a recursive acronym for SPARQL Protocol And RDF Query Language.
  - Two versions : 1.0 (2008) and 1.1 (2013).
  - Version 1.1 also offers update facilities.
- It is the main language for querying linked data.
  - The data can be expressed in RDF, RDFS or OWL.
  - It is expressed in RDF triples.
- RDF triples data consist of a (or several) graph(s).
  - One default graph plus possibly additional named graphs.
- SPARQL is the languages of choice for querying RDF graph databases (RDF stores).



Graph matching: basic idea

Dataset



Query pattern

Result

wolf
:Motherwolf
:Packleader

Graph matching: basic idea

- The query is a graph pattern matched against the graph dataset.

  Types of graph patterns :

  - The basic graph pattern.
  - Group, optional, union graph patterns, named graphs.

- Basic graph pattern :

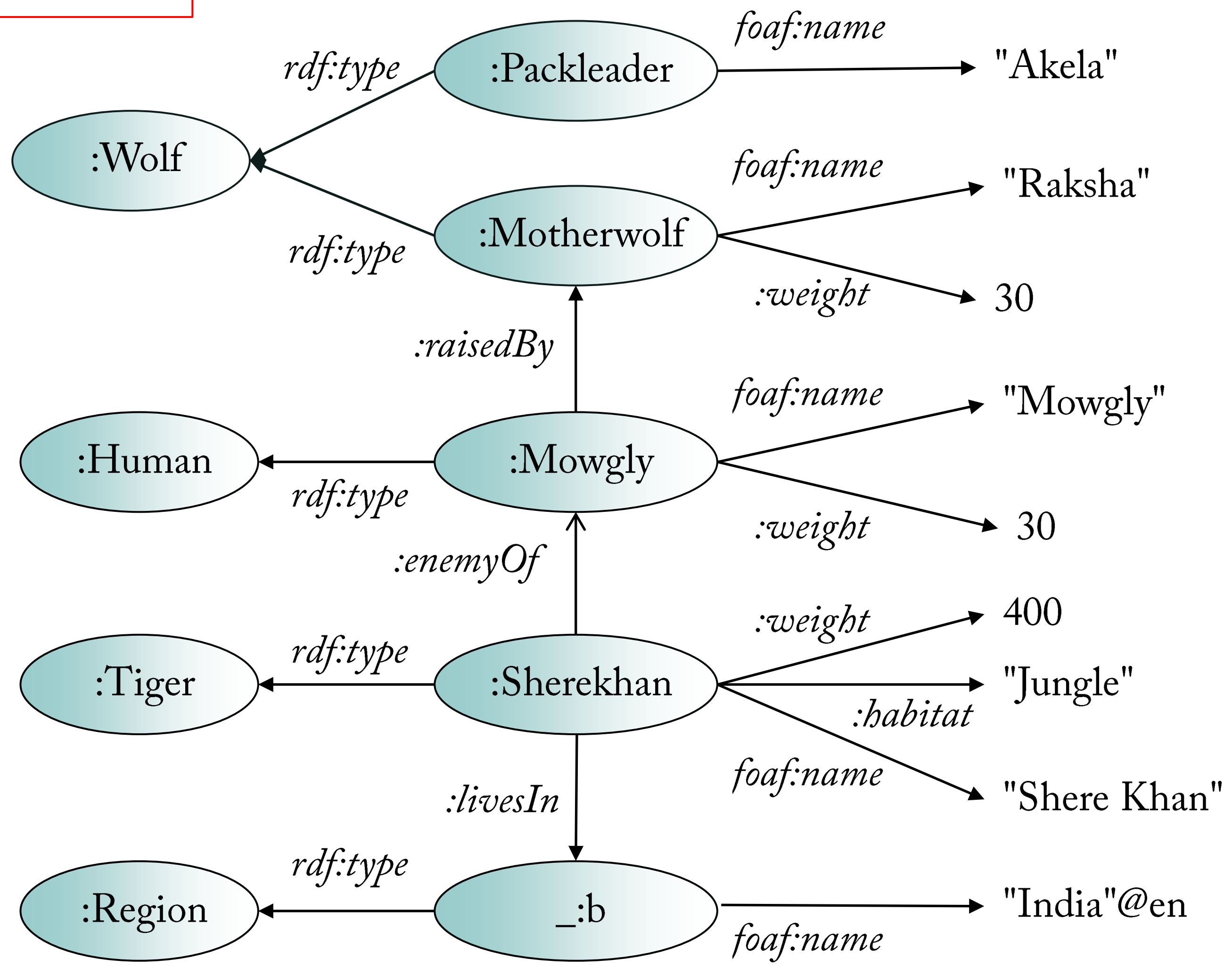
  - A set of triples, considered as conjoined.
  - They are expressed in Turtle syntax and grouped between {}. Turtle's abbreviations can be used.
  - A variable, prefixed with the symbol ?, can appear in subject, predicate or object position.

  {?entity rdf:type :Wolf .}

  {?entity a :Wolf .}

Graph matching: basic idea ./.

Dataset



Query pattern

{ ?entity foaf:name ?name . }

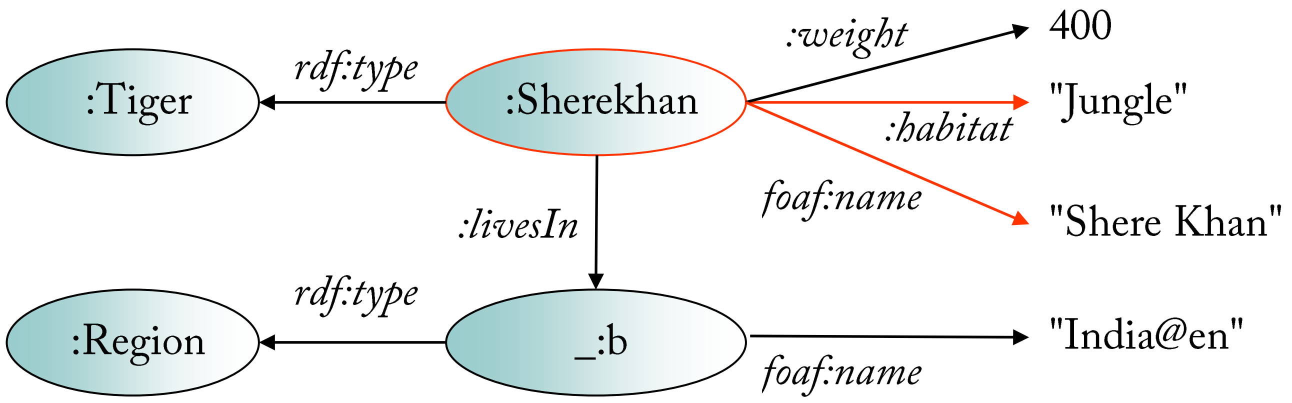
Variables can appear in any position of a triple.

Result

entity	name
:Mowgly	Mowgly
_:b0	India@en
:Motherwolf	Raksha
:Packleader	Akela
:Sherekhan	Shere Khan

Graph matching: combining triples

Dataset



Query pattern

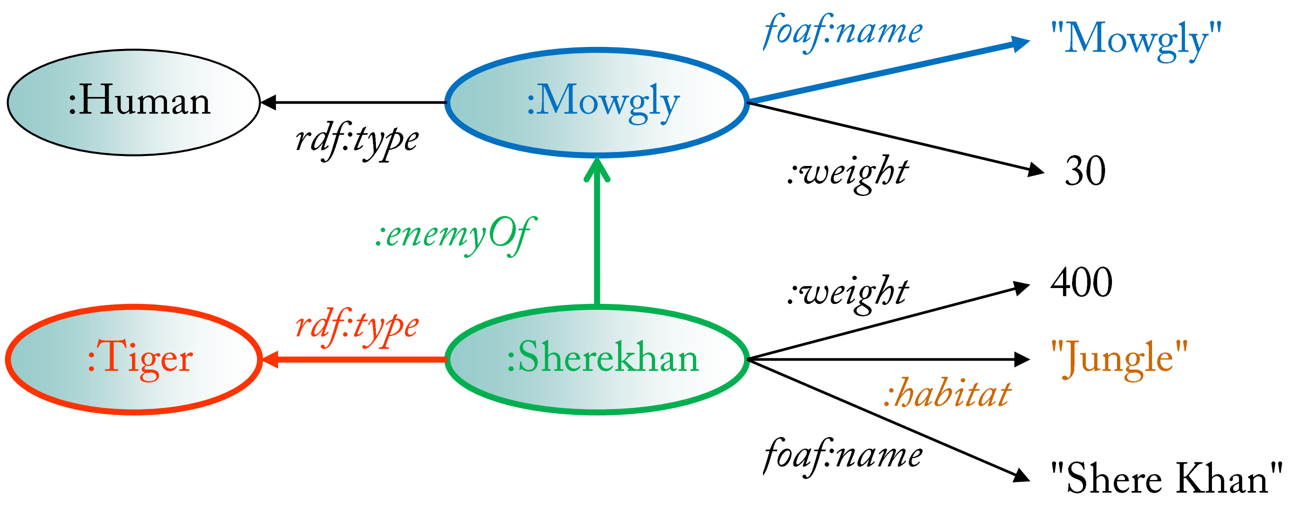
{?entity foaf:name "Shere Khan";
:habitat ?habitat .}

Result

entity	habitat
:Sherekhan	Jungle

Graph matching: traversing a graph

Dataset



Query pattern

{ ?entity foaf:name "Mowgly" .
?enemy :enemyOf ?entity ;
a ?type .
?enemy :habitat ?habitat}

Result

entity	enemy	type	habitat
:Mowgly	:Sherekhan	:Tiger	Jungle

Agenda

1	Querying RDF graphs
2	SPARQL syntax and semantics
3	Entailment regimes and OWL

A full example

DATASET FILE (Turtle)

```
PREFIX : <http://www.example.org/>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX owl: <http://www.w3.org/2002/07/owl#>

…

:Mowgly:Mowgly          rdf:type :Human ;
                        :weight 30 ;
                        :raisedBy :Motherwolf;
                        foaf:name "Mowgly" .

:Sherekhan:Sherekhan    rdf:type :Tiger ;
                        foaf:name "Shere Khan" ;
                        :livesIn _:b ;
                        :habitat "Jungle" ;
                        :enemyOf :Mowgly ;
                        :weight 400 .

:Motherwolf:Motherwolf rdf:type :Wolf ;
                        :weight 30 ;
                        foaf:name "Raksha" .

:eats rdf:type          owl:ObjectProperty ;
                        owl:inverseOf :is_eated_by ;
                        rdfs:domain :Animal .
…
```

RDF/XML

SPARQL QUERY ENDPOINT
http://librdf.org/query/

DATASET :
http://www.montefiore.ulg.ac.be/~binot/INFO8005/Junglebook.rdf

QUERY:

```
PREFIX : <http://www.example.org/>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT * WHERE
{ ?entity foaf:name "Mowgly" .
?enemy :enemyOf ?entity ; a ?type.
?enemy :habitat ?habitat}
```

Count	entity	enemy	type	habitat
1	http://www.example.org/Mowgly	http://www.example.org/Sherekhan	http://www.example.org/Tiger	Jungle

SPARQL basic query structure

- SQL like syntax (SELECT … FROM …. WHERE) :

\# prefix declarations.

PREFIX ex: <http://example.com/resources/>

\# query form : how to form the answer.

SELECT	instantiation values (bindings) of variables.
CONSTRUCT	RDF graph.
DESCRIBE	generation of a description of resources found in the source graph.
ASK	boolean value.

\# dataset definition (optional if using the default source graph).

FROM <source graph>

\# query graph pattern.

WHERE { <graph pattern> }

\# query modifiers : modifying the answer.

ORDER BY, DISTINCT, LIMIT, OFFSET …

Graph patterns

- A basic graph pattern is a set of (conjoined) triples.
- A graph pattern may be defined by combining smaller patterns :

If Gp and Gp' are graph patterns, the following are also graph patterns :

{Gp}	Group graph pattern (a solution must be a solution of every group).
Gp FILTER (<test>)	Constraints on solution expressed by Boolean expressions. The FILTER condition can appear anywhere in the pattern.
Gp OPTIONAL {Gp'}	Optional graph pattern.
{Gp} UNION {Gp'}	Union graph pattern.
GRAPH ?g {Gp_1 … Gp_n}	Allows ?g to range over a list of named graph patterns.

Examples

SELECT * WHERE

{?entity foaf:name ?name .
OPTIONAL
{?entity :weight ?weight}}

entity	name	weight
:Mowgly	Mowgly	30^^xsd:integer
_:b0	India@en	
:Motherwolf	Raksha	30^^xsd:integer
:Packleader	Akela	
:Sherekhan	Shere Khan	400^^xsd:integer

{?animal :weight ?weight .
FILTER (?weight > 100) .
?animal foaf:name ?name}

animal	weight	name
:Sherekhan	400^^xsd:integer	Shere Khan

{{?entity a :Wolf }
UNION
{?entity :weight ?weight}}

entity	weight
:Motherwolf	
:Packleader	
:Mowgly	30^^xsd:integer
:Motherwolf	30^^xsd:integer
:Sherekhan	400^^xsd:integer

(for an example of GRAPH graph patterns please refer to SPARQL documentation).

Matching literals and blank nodes

SELECT * WHERE

{?entity :livesIn ?country .
?country foaf:name "India"}

no answer

{?entity :livesIn ?country .
?country foaf:name "India"@en }

entity	country
:Sherekhan	_:b0

- Language tags in SPARQL are preceded by a @. The query must specify the full tag.
- Integers are recognized as shorthand notation for the integer datatype.
- A blank node may appear in the answer.
  - The blank node label is generated for - and consistent in - the solution set; it may not be the same as in the dataset.

Query modifiers

- **LIMIT** : puts an upper bound on the number of solutions returned.

```
SELECT * WHERE
{?entity foaf:name ?name. }
LIMIT 3
```

entity	name
:Mowgly	Mowgly
_:b0	India@en
:Motherwolf	Raksha

- **DISTINCT** : eliminates duplicate solutions.

```
SELECT DISTINCT ?weight WHERE
{?animal :weight ?weight. }
```

weight
30^^xsd:integer
400^^xsd:integer

- **ORDER BY** : requires an **ASC**ending or **DESC**ending order of a solution sequence.

```
SELECT * WHERE
{?x foaf:name ?name }
ORDER BY DESC(?name)
```

x	name
:Sherekhan	Shere Khan
:Motherwolf	Raksha
:Mowgly	Mowgly
_:b0	India@en
:Packleader	Akela

Query form DESCRIBE

- The DESCRIBE form returns a single result RDF graph containing RDF data about the resource identified by the variable in the query graph pattern.
- This data is determined by the SPARQL query processor; the query client does not need to know the structure of the RDF in the data source.

```
DESCRIBE ?x WHERE
{?x foaf:name "Shere Khan". }
```

```
<rdf:RDF
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns="http://www.example.org/"
    xmlns:owl="http://www.w3.org/2002/07/owl#"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
    xmlns:foaf="http://xmlns.com/foaf/0.1/"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema#" >
  <rdf:Description rdf:about="http://www.example.org/Sherekhan">
    <weight rdf:datatype="http://www.w3.org/2001/XMLSchema#integer">400</weight>
    <ennemyOf rdf:resource="http://www.example.org/Mowgly"/>
    <habitat>Jungle</habitat>
    <livesIn rdf:nodeID="A0"/>
    <foaf:name>Shere Khan</foaf:name>
    <rdf:type rdf:resource="http://www.example.org/Tiger"/>
  </rdf:Description>
  <rdf:Description rdf:nodeID="A0">
    <foaf:name xml:lang="en">India</foaf:name>
    <rdf:type rdf:resource="http://www.example.org/Region"/>
  </rdf:Description>
</rdf:RDF>
```

Aggregates

- SPARQL 1.1 introduced aggregates :

  COUNT, SUM, MIN, MAX, AVG, GROUP_CONCAT, and SAMPLE.

  - GROUPBY is used to divided the solutions into groups; the aggregate value is calculated for each group.
  - Aggregate expressions are required to be aliased to a variable, using AS.

- Example

Find out how many entities in each class of named individuals are in the dataset.

```
SELECT ?class (COUNT(?entity) AS ?count)
WHERE {
   ?entity foaf:name ?name .
   ?entity a ?class .
} GROUPBY ?class
```

class	count
:Wolf	2^^xsd:integer
:Region	1^^xsd:integer
:Human	1^^xsd:integer
:Tiger	1^^xsd:integer

Semantics of an answer to a graph pattern query

- As we did for RDF, let us define :
  - M_V as a functional mapping from the set of variables V of a graph G to a set of literals, blank nodes and URIs;
  - $M_V(G)$ as the graph resulting from applying the mapping M_V to graph G.
- A mapping M_V is a solution for a query graph pattern G_p against a source graph G_s if $M_V(G_p)$ is a subgraph* of G_s.
- The result of a SELECT request is :
  - the set of all solutions for the graph pattern of the request, or equivalently :
  - the set of all instances of M_V such as $M_V(G_p)$ is entailed by the source graph : $G_s \vDash M_V(G_p)$.

Note : the above definition corresponds to simple graph entailment.
Applying RDF or RDFS inference rules will lead to different entailment regimes (cf. later).

* : A *subgraph* of an RDF graph is a subset of the triples in the graph.

Agenda

1	Querying RDF graphs
2	SPARQL syntax and semantics
3	Entailment regimes and OWL

Beyond RDF ?

- SPARQL is very appropriate for querying triple graphs :
  - It offers a natural and intuitive way to navigate graphs.
  - It is perceived by non-IT users as having a better readability than SQL.
  - It will be the language of choice for RDF graph databases.

- Quid, however, of querying higher level ontology formalisms, such as RDFS and OWL ?

Reminder : RDF/RDFS entailment patterns

	If S contains:	then S RDF(S)_entails :
rdfD1	a p "v"^^d .	a p _:b . _b rdf:type d .
rdfD2	a p b .	p rdf:type rdf:Property .
rdfs1	any IRI a in D	a rdf:type rdfs:Datatype .
rdfs2	p rdfs:domain x . a p b .	a rdf:type x .
rdfs3	p rdfs:range x . a p b .	b rdf:type x .
rdfs4a	a p b .	a rdf:type rdf:Resource .
rdfs4b	a p b .	b rdf:type rdf:Resource .
rdfs5	p rdfs:subPropertyOf q . q rdfs:subPropertyOf r .	p rdfs:subPropertyOf r .
rdfs6	a rdf:type rdf:Property .	a rdfs:subPropertyOf a .
rdfs7	rdfs:subPropertyOf q . a p b.	a q b .
rdfs8	c rdf:type rdfs:Class .	c rdfs:subClassOf rdf:Resource .
rdfs9	c rdfs:subClassOf d . a rdf:type c.	a rdf:type d .
rdfs10	c rdf:type rdfs:Class .	c rdfs:subClassOf c .
rdfs11	c rdfs:subClassOf d . d rdfs:subClassOf e .	c rdfs:subClassOf e .
rdfs12	rdf:type rdfs:ContainerMembershipProperty .	p rdfs:subPropertyOf rdfs:member .
rdfs13	d rdf:type rdfs:Datatype .	d rdfs:subClassOf rdfs:Literal .

Influence of entailment regime

- Dataset :

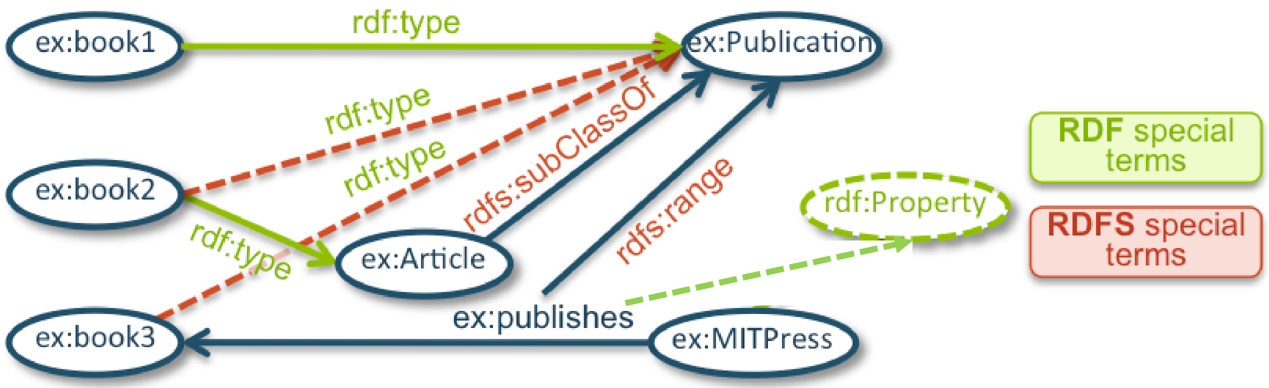
ex:book1 rdf:type ex:Publication .

ex:book2 rdf:type ex:Article .

ex:Article rdfs:subClassOf ex:Publication .

ex:publishes rdfs:range ex:Publication .

ex:MITPress ex:publishes ex:book3 .



SELECT ?prop WHERE { ?prop rdf:type rdf:Property }

Answer under

- Simple entailment : {}
- RDF entailment : {ex:publishes}

SELECT ?pub WHERE { ?pub rdf:type ex:Publication }

Answer under

- Simple and RDF entailment : {ex:book1}
- RDFS entailment : {ex:book1, ex:book2, ex:book3}

- The answer provided by a SPARQL engine depends on the entailment regime supported by that specific engine (e.g., Ontop and Ontotext support RDFS).

Querying OWL with SPARQL

- Dataset (fragment of ontology in OWL Turtle syntax)

```
:Lion        rdf:type owl:Class ;
             rdfs:subClassOf      :Carnivore ,
                                   [ rdf:type owl:Restriction ;
                                     owl:onProperty :livesIn ;
                                     owl:hasValue :Afrika ] .
```

- Query : *where do lions live ?*

```
SELECT * WHERE
{:Lion rdfs:subClassOf ?class .
?class owl:onProperty :livesIn .
?class owl:hasValue ?place . }
```

class	place
_:b0	:Afrika

- Querying OWL with SPARQL requires to know and navigate the OWL triple structure.
- Better solutions include using a programming interface (OWLAPI), or OBDA.

Summary

- Web ontology languages (RDF, RDFS, OWL) are all based on a graph of triples data model. Querying triple datasets amounts to querying a graph.
- The SPARQL query language is based on graph pattern matching. The core structure is the basic graph pattern, corresponding to a conjunction of triples expressed in Turtle. More complex graph patterns are provided.
- SPARQL has been provided with an SQL-like syntax to facilitate its adoption, supporting selection, grouping, filtering, aggregates …
- As for RDF, SPARQL semantics is defined in terms of graph entailment.
- Different entailment regimes exist, depending on the language used and the choices of the query engines. RDF and RDFS entailments must take into account the inference capabilities of these languages, beyond simple entailment.

References

- *[Hartig 2012] : Hartig O.*, An Introduction to SPARQL and queries over Linked Data, chapter 2 SPARQL, *tutorial of the 12th International Conference on Web Engineering ICWE 2012, Berlin, 2012.*

THANK YOU