RDF and SPARQL

Mac Qiao

SEAT
Massey University

Data Model: RDF

Why we need RDF? Semantic Web.

> *"The Semantic Web is an extension of the current web in which information is given well-defined meaning, better enabling computers and people to work in cooperation."*
>
> *(Tim Berners-Lee et al. 2001.)*

Specific goals:

- A descriptive language with standard semantics.
  - Make semantics machine-processable and understandable.
- Incorporate logical infrastructure to reason about resources.
- W3C proposals: Resource Description Framework (RDF) and its query language SPARQL.

- Data Model: RDF
- Query Language: SPARQL

Data Model: RDF

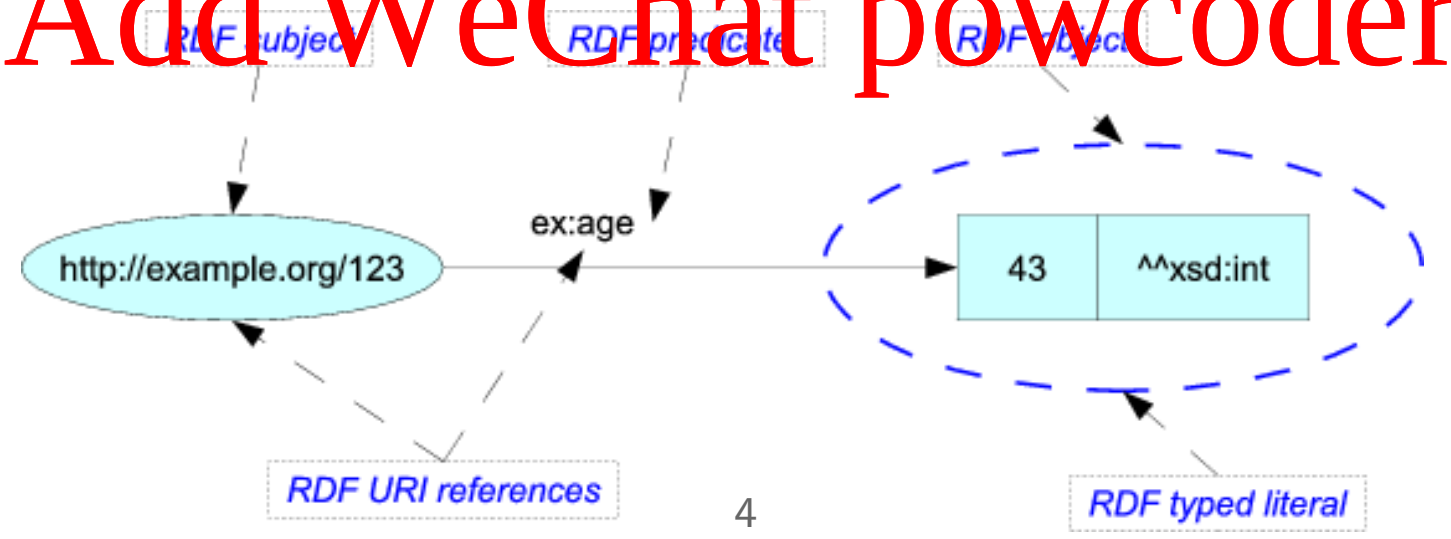
Resource Description Framework (RDF)

RDF is a graph-based model for representing the web as a graph of

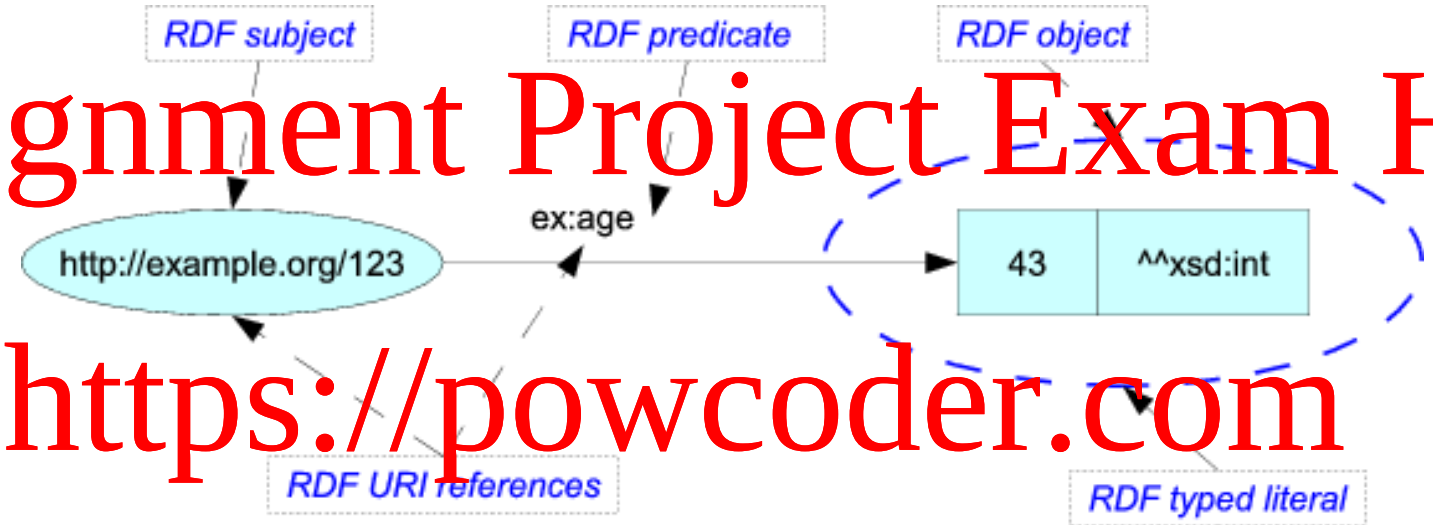
- Resources and their properties, using
- Internationalized Resource Identifiers (IRIs) [a] and literal values.
  - IRI: `http://example.org/123`
  - Literal: 438sd:int

[a] A generalized Unified Resource Identifier that permits more characters.

The graph is a collection of subject-predicate-object triples (RDF triples).



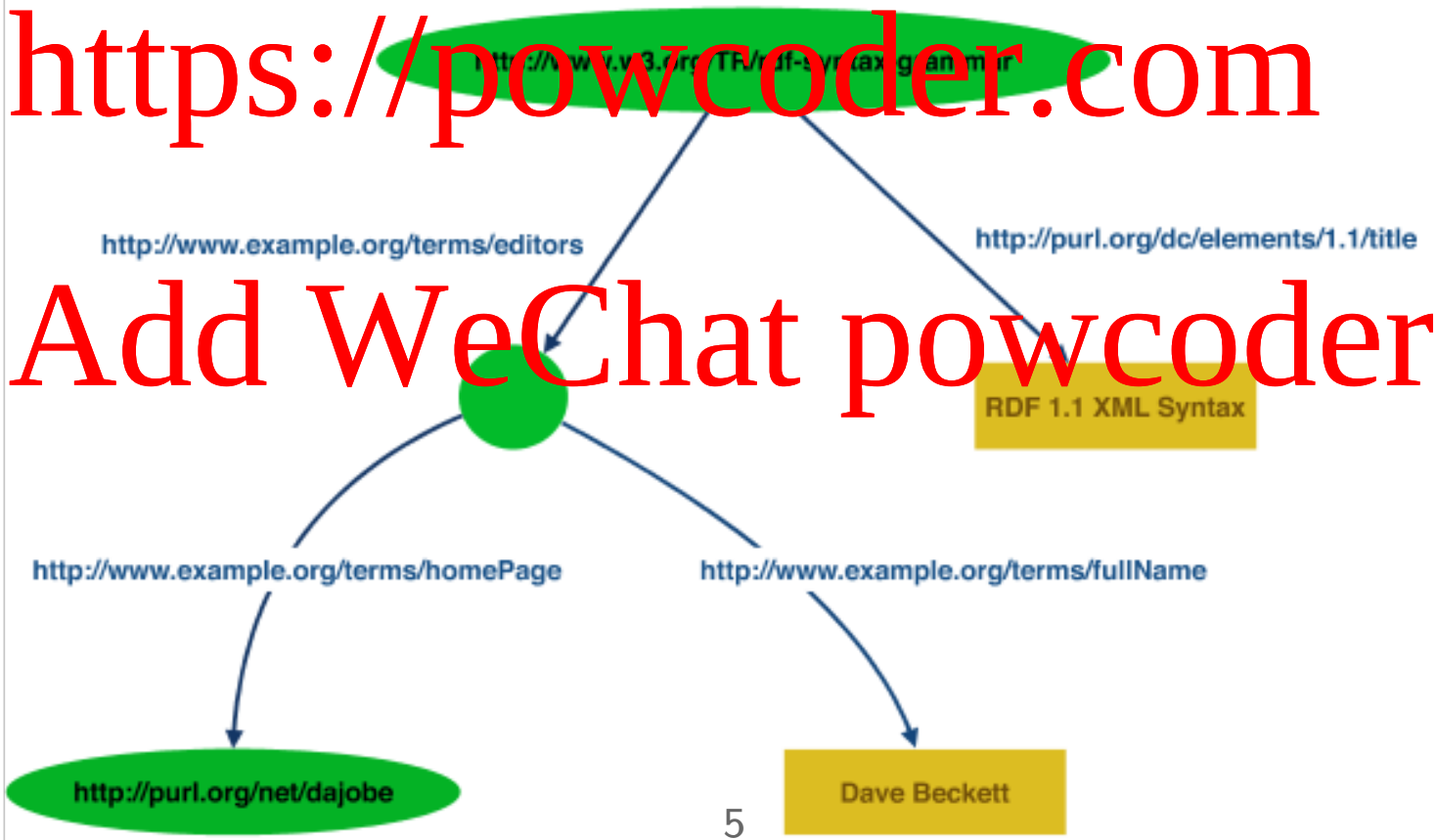
Data Model: RDF



- RDF Statement : an RDF triple.
- An RDF triple:
  - Subject :IRI or blank node.
  - Predicate : IRI.
  - Object : IRI, blank node or literal.

Data Model: RDF

- RDF Statement : an RDF triple.
- An RDF triple:
  - Subject :IRI or blank node.
  - Predicate : IRI.
  - Object : IRI, blank node or literal.



Data Model: RDF

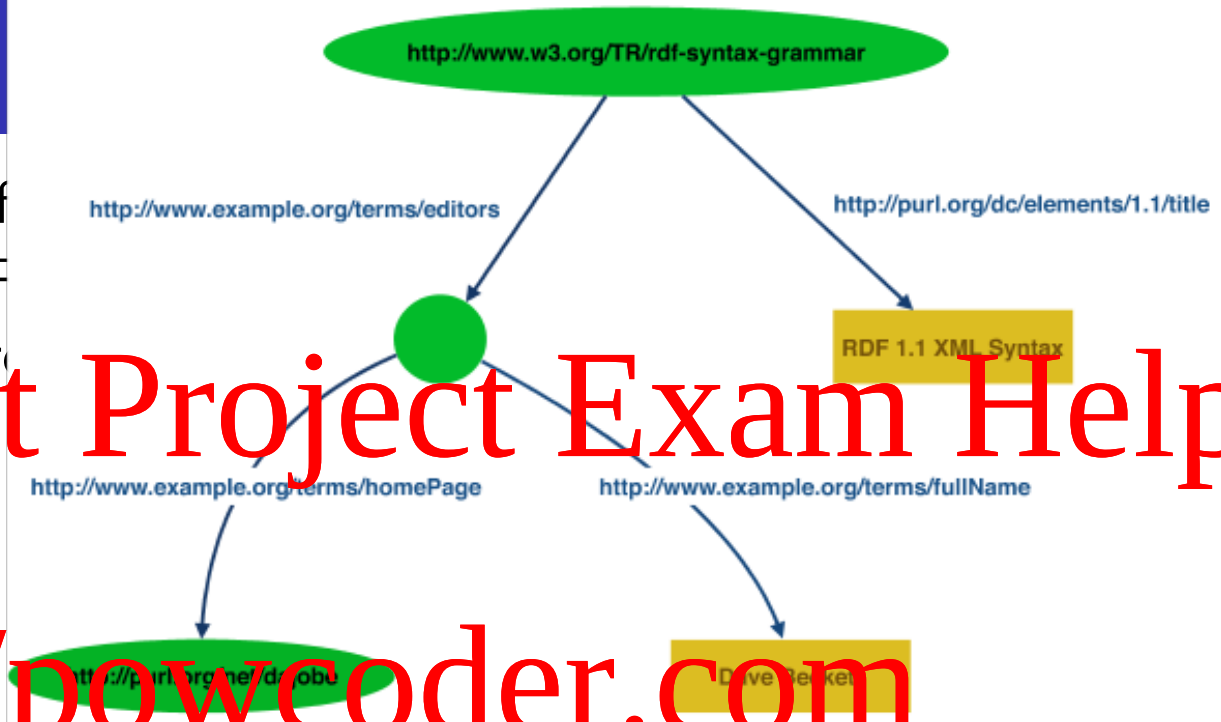
One can write RDF in different syntaxes:

- RDF/XML: respent RDF data in XML.
- Turtle: a compact presentation of RDF data in RDF triples.

Data Model: RDF

One can write RDF in diff

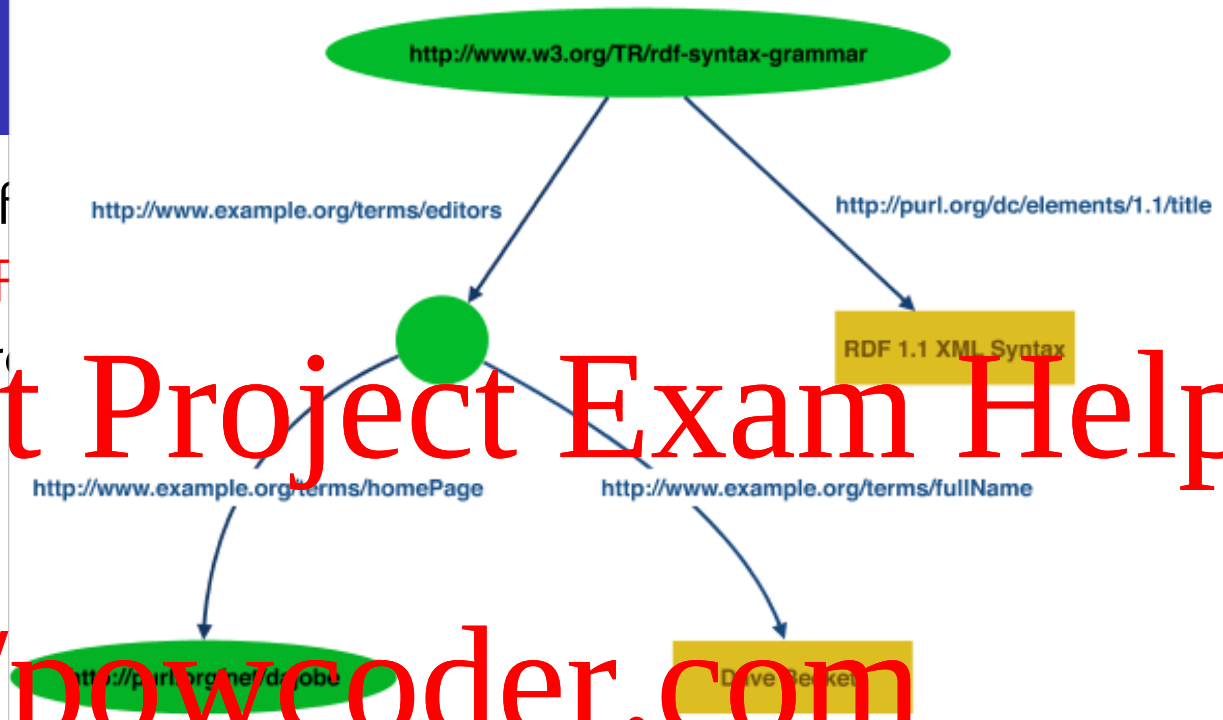
- RDF/XML: respent F
- Turtle: a compact pr



Data Model: RDF

One can write RDF in diff

- RDF/XML: respent F
- Turtle: a compact pr

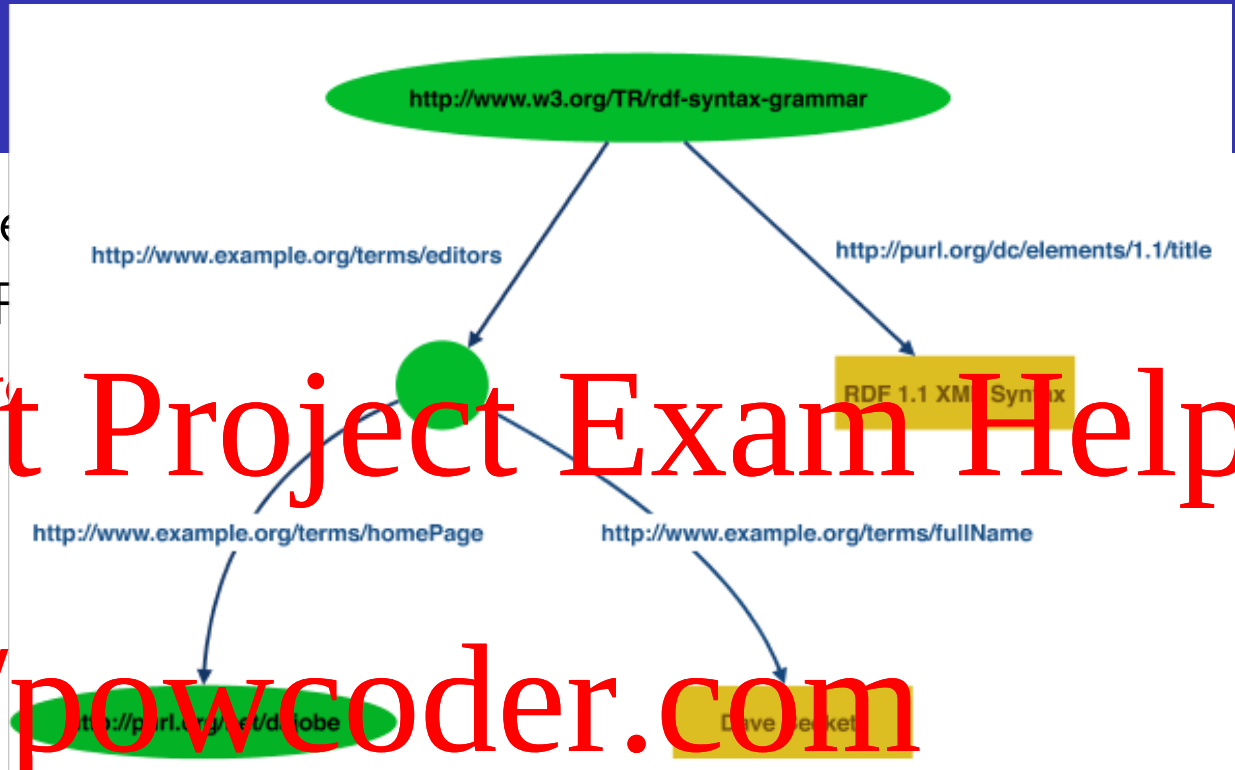



```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:dc="http://purl.org/dc/elements/1.1/"
         xmlns:ex="http://example.org/stuff/1.0/">
  <rdf:Description rdf:about="http://www.w3.org/TR/rdf-syntax-grammar"
           dc:title="RDF1.1 XML Syntax">
    <ex:editor>
      <rdf:Description ex:fullName="Dave Beckett">
        <ex:homePage rdf:resource="http://purl.org/net/dajobe/" />
      </rdf:Description>
    </ex:editor>
  </rdf:Description>
</rdf:RDF>
```

Data Model: RDF

RDF document can be pre

- RDF/XML: respent F
- Turtle: a compact pr




```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix dc: <http://purl.org/dc/elements/1.1/> .
@prefix ex: <http://example.org/stuff/1.0/> .

<http://www.w3.org/TR/rdf-syntax-grammar>
  dc:title "RDF/XML Syntax" ;
  ex:editors [
    ex:fullname "Dave Beckett";
    ex:homePage <http://purl.org/net/dajobe/>
  ] .
```

Data Model: RDF

RDF document can be presented in different syntaxes.

- RDF/XML: respent RDF data in XML.
- Turtle: a compact presentation of RDF data in RDF triples.

Next, we shall mainly use the Turtle syntax to describe an RDF document.

RDF Schema

RDF Schema standardizes RDF vocabulary for describing classes and properties

- using object-oriented types + domain/range
- subclasses (sc), subproperties (sp), domain/range, $\cdots$

Data Model: RDF

RDF Schema

RDF Schema standardizes RDF vocabulary for describing classes and properties

- using object-oriented types + domain/range
- subclasses (sc), subproperties (sp), domain/range, $\cdots$

Range and domain:

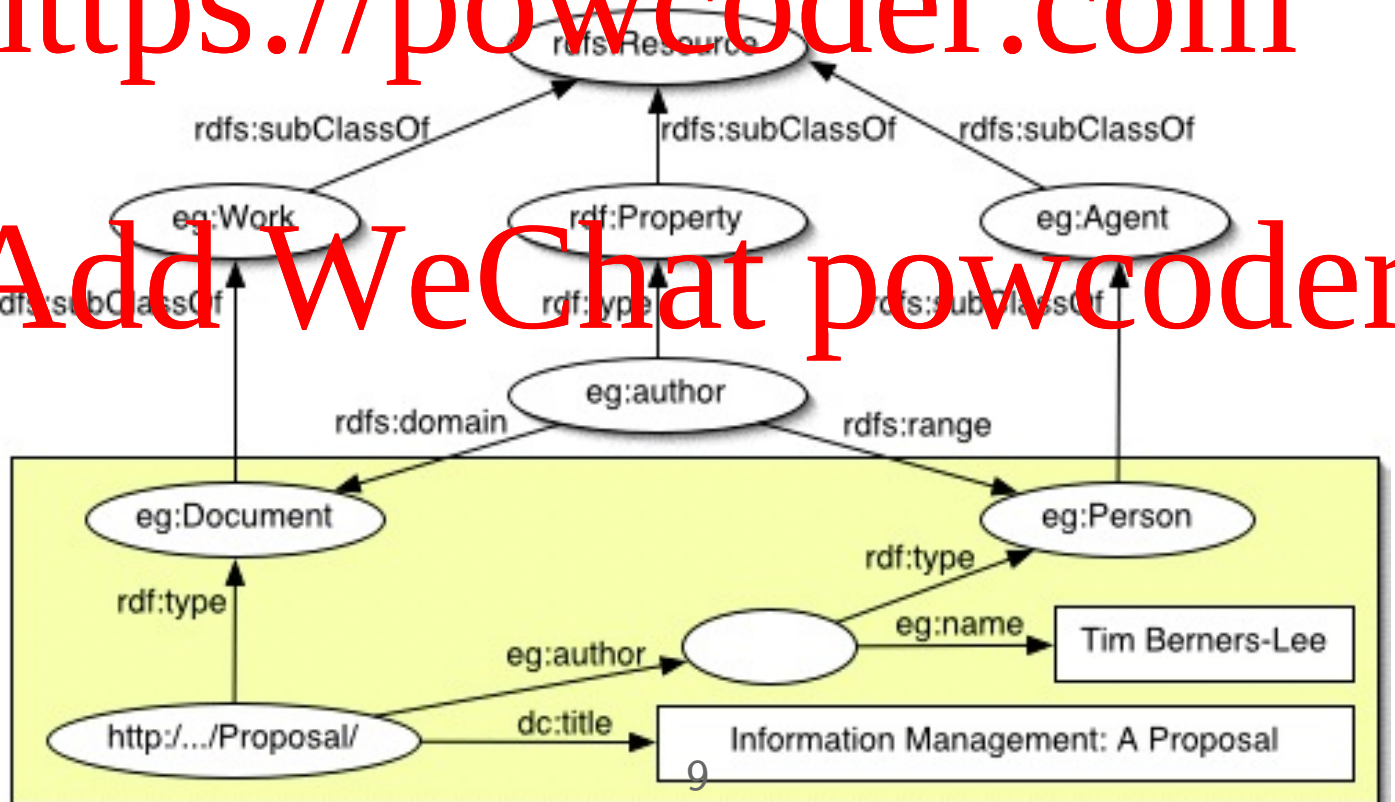
- A rdfs:domain B
  - A is an instance of the class rdf:Property.
  - B is a instance of the class rdfs:Class.
  - The subjects of triples whose predicate is A are the instances of B.
- A rdfs:range B
  - A is an instance of the class rdf:Property.
  - B is a instance of the class rdfs:Class.
  - The objects of triples whose predicate is A are the instances of B.

Data Model: RDF

RDF Schema

RDF Schema standardizes RDF vocabulary for describing classes and properties

- using object-oriented types + domain/range
- subclasses (sc), subproperties (sp), domain/range, ···



Data Model: RDF

RDF Schema

RDF Schema standardizes RDF vocabulary for describing classes and properties

- using object-oriented types + domain/range
- subclasses (sc), subproperties (sp), domain/range, $\cdots$

The semantics can be used as an inference system

- Sub-property:
  - $(A, sp, B) + (B, sp, C) => (A, sp, C)$
  - $(A, sp, B) + (X, A, Y) => (X, B, Y)$
- Subclass:
  - $(A, sc, B) + (B, sc, C) => (A, sc, C)$
  - $(A, sc, B) + (X, type, A) => (X, type, B)$
- Typing:
  - $(A, dom, B) + (X, A, Y) => (X, type, B)$
  - $(A, range, B) + (X, A, Y) => (Y, type, B)$

Data Model: RDF

prefix	Namespace IRI	RDF vocabulary
rdfs	`http://www.w3.org/2000/01/rdf-schema#`	The RDF Schema vocabulary
rdf	`http://www.w3.org/1999/02/22-rdf-syntax-ns#`	The RDF built-in vocabulary

rdfs

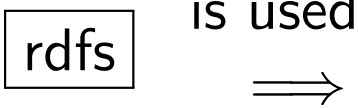
Data Model: RDF

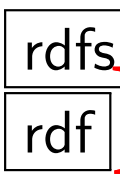
prefix	Namespace IRI	RDF vocabulary
rdfs	`http://www.w3.org/2000/01/rdf-schema#`	The RDF Schema vocabulary
rdf	`http://www.w3.org/1999/02/22-rdf-syntax-ns#`	The RDF built-in vocabulary

rdfs is used to define $\Longrightarrow$ rdf

Data Model: RDF

prefix	Namespace IRI	RDF vocabulary
rdfs	`http://www.w3.org/2000/01/rdf-schema#`	The RDF Schema vocabulary
rdf	`http://www.w3.org/1999/02/22-rdf-syntax-ns#`	The RDF built-in vocabulary





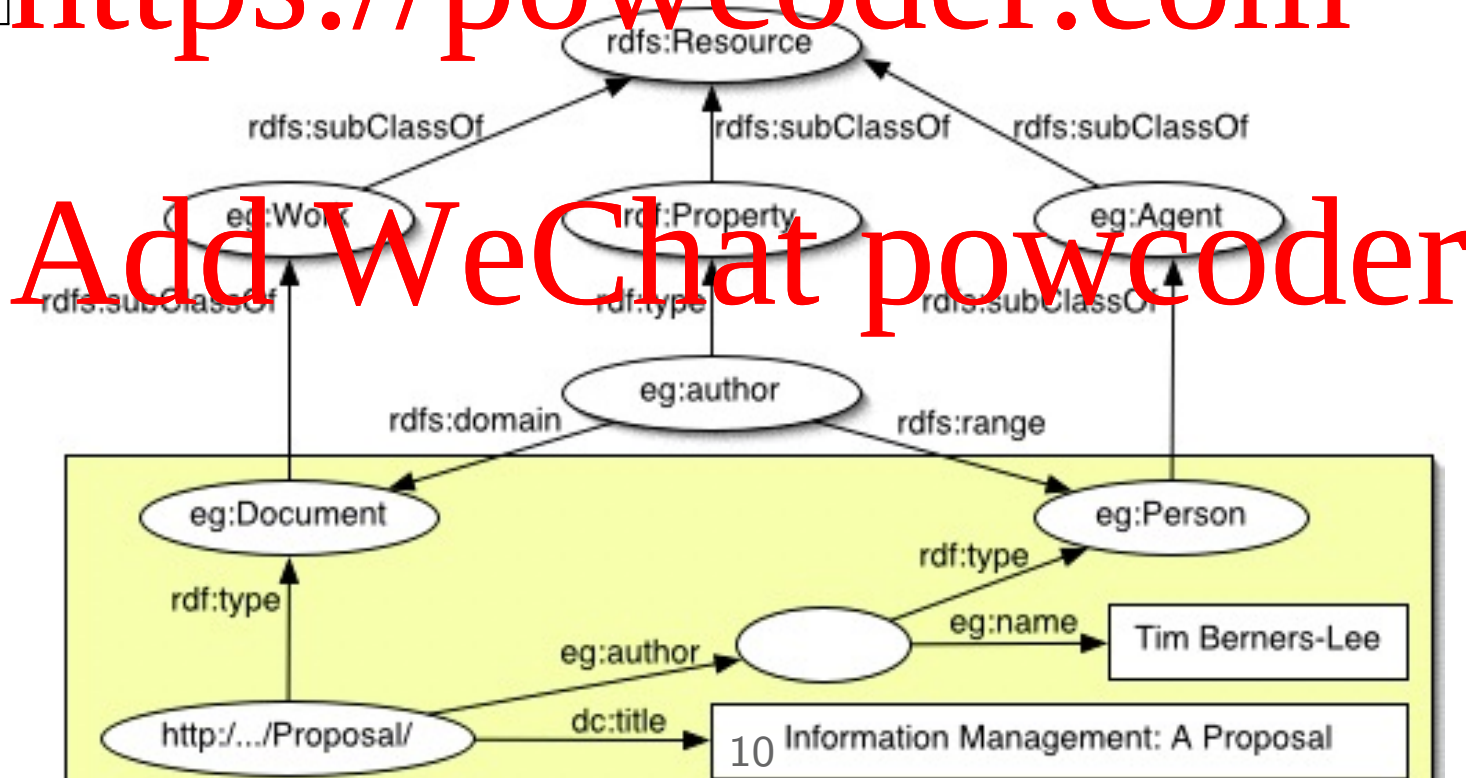
Data Model: RDF

prefix	Namespace IRI	RDF vocabulary
rdfs	`http://www.w3.org/2000/01/rdf-schema#`	The RDF Schema vocabulary
rdf	`http://www.w3.org/1999/02/22-rdf-syntax-ns#`	The RDF built-in vocabulary

rdfs is used to define rdf

$\Longrightarrow$

rdfs, rdf are used to define other vocabularies such as dc, owl, $\cdots$



Data Model: RDF

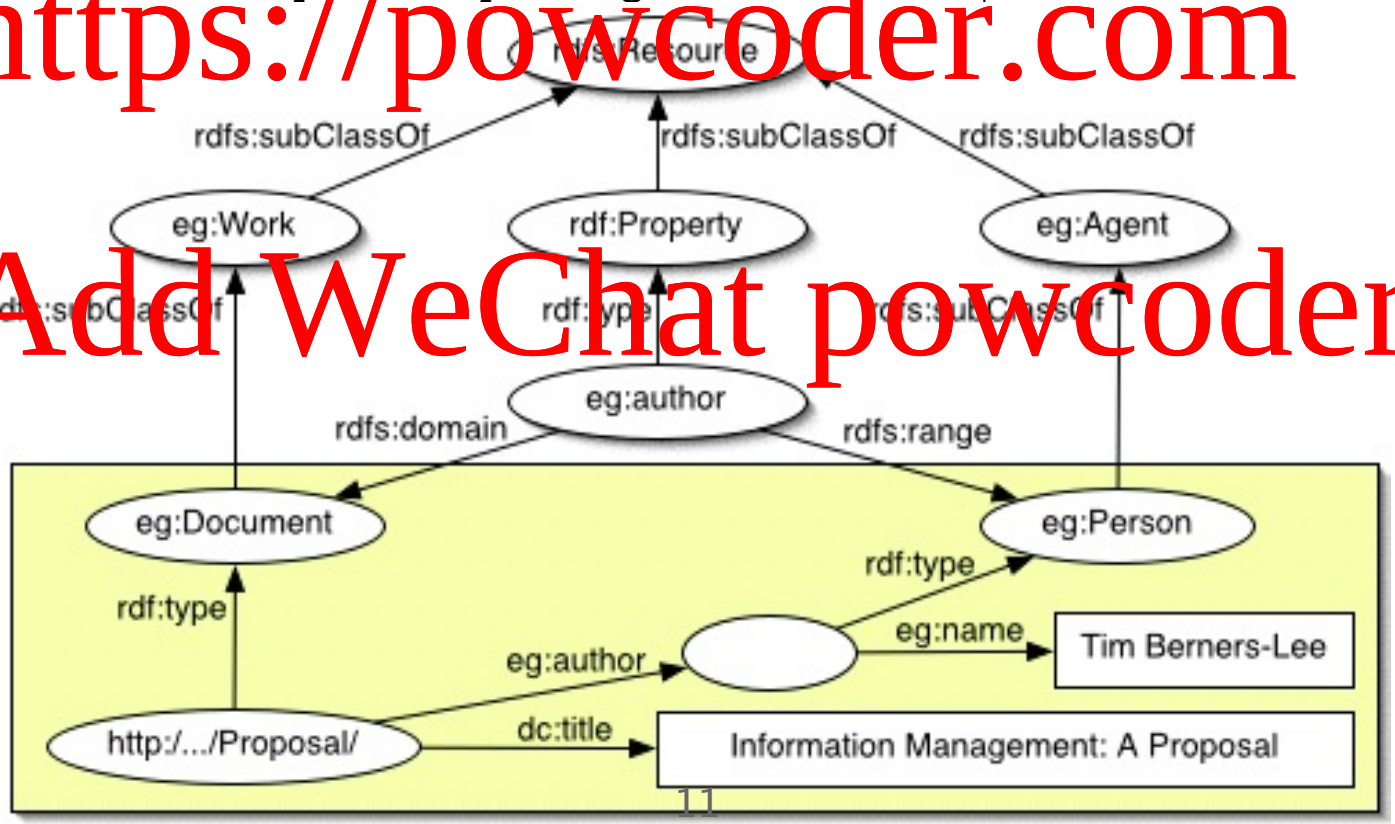
- Namespace URI: the common substring of the URIs in an RDF vocabulary.

prefix	Namespace IRI	RDF vocabulary
rdf	http://www.w3.org/1999/02/22-rdf-syntax-ns#	The RDF built-in vocabulary
rdfs	http://www.w3.org/2000/01/rdf-schema#	The RDF Schema vocabulary
xsd	http://www.w3.org/2001/XMLSchema#	The RDF-compatible XSD types
dc	http://purl.org/dc/elements/1.1#	"Dublin Core Metadata Element S
eg	http://example.org/	the IRI defined locally.

Data Model: RDF

- Namespace URI: the common substring of the URIs in an RDF vocabulary.

prefix	Namespace IRI	RDF vocabulary
rdf	http://www.w3.org/1999/02/22-rdf-syntax-ns#	The RDF built-in vocabulary
rdfs	http://www.w3.org/2000/01/rdf-schema#	The RDF Schema vocabulary
xsd	http://www.w3.org/2001/XMLSchema#	The RDF-compatible XSD types
dc	http://purl.org/dc/elements/1.1#	"Dublin Core Metadata Element S
eg	http://example.org/	the IRI defined locally.



Data Model: RDF

Ontology = ontos(being) + logos(word).

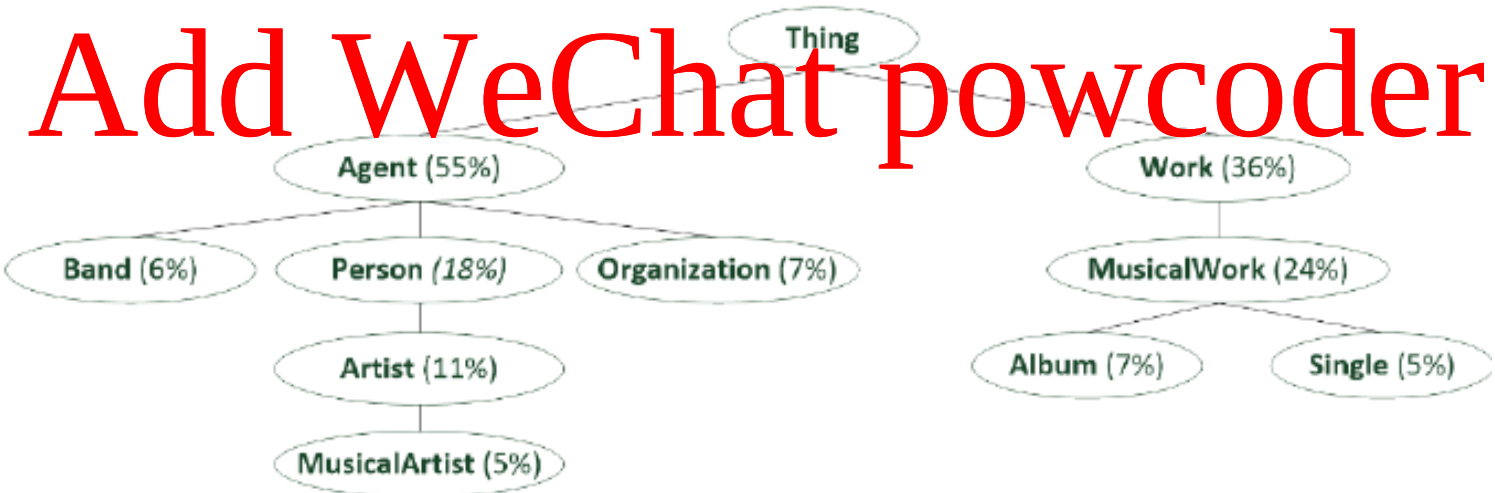
- An ontology is a systematic explanation of existence (400BC).
- Grube (1998)
  - Formal: machine readable
  - Explicit: specifications of concepts, properties, functions, axioms are defined.
  - Shared: consensual knowledge
  - Conceptual: abstract model of some phenomena in the world.

Data Model: RDF

Ontology = ontos(being) + logos(word).

- An ontology is a systematic explanation of existence (400BC).
- Gruber (1998)
  - Formal: machine readable
  - Explicit: specifications of concepts, properties, functions, axioms are defined.
  - Shared: consensual knowledge
  - Conceptual: abstract model of some phenomena in the world.

Example: Define the concept of "MusicalArtist". `http://dbpedia.org/fct/`



- Data Model: RDF
- Query Language: SPARQL

Querying RDF: SPARQL

An SQL query retrieves data from a database under the relational model.

```
SELECT name                    data needed
FROM student                   data source
WHERE age > 20                 data constraint
```


SPARQL

- a recursive acronym that stands for *SPARQL Protocol and RDF Query Language*
- a graph-matching query language for RDF graphs.
- recommended by W3C
- Like SQL, SPARQL consists of three components:
  - select: the entities to be returned
  - from: the data source (RDF graph)
  - where: the pattern to be matched against the RDF graph
  - optional prologue: namespace

Querying RDF: SPARQL

Example: find 10 films with the keyword of "Paddington" in thier labels.

```
prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>
prefix dbpedia-owl: <http://dbpedia.org/ontology/>
prefix movie: <http://data.linkedmdb.org/resource/movie/>

select distinct ?film where {
  { ?film a movie:Film } union
  { ?film a dbpedia-owl:Film }
  ?film rdfs:label ?label .
  filter regex( str(?label), "Paddington", "i")
}
limit 10
```

Try this on the SPARQL endPoint of dbpedia
`https://dbpedia.org/sparql`.

Query Language: SPARQL

The WHERE clause specifies the graph pattern which

- is an RDF graph
- involves variables
- has the basic pattern of triple pattern
- can be complex

Semantic: for a pattern P and an RDF graph G, we denote by $[\![P]\!]_G$ the set of matchings of P, that are, subgraphs of G that match P.

Triple pattern

Example: A triple pattern can have one/more variables.

- ?film rdfs:label ?label
- ?film a movie:film

Note: a is the abbreviation of rdfs:type.


```
select ?film where {
   ?film a dbpedia-owl:Film
} limit 10
```

https://dbpedia.org/sparql.

Query Language: SPARQL

A larger pattern is assembled with triple patterns.

The queries below are equivalent.

```
select ?film
where {
   {?film a movie:Film      }
   {?film rdfs:label ?label }
}
```

```
select ?film
where {
   ?film a movie:Film      .
   ?film rdfs:label ?label .
}
```


Semantics: $[\![P_1 \text{ AND } P_2]\!]_G = [\![P_1]\!]_G \bowtie [\![P_2]\!]_G$

Query Language: SPARQL

Alternative Graphs: Union


```
select ?film
where {
   {?film a movie:Film } UNION
   {?film rdfs:label ?label}
}
```


Return the films that EITHER satisfy the pattern of

- ?film a movie:Film or
- ?film rdfs:label ?label

Semantic: $[P1 UNION P2] = [P1] \cup [P2]$.

Query Language: SPARQL

Optional pattern:


```
select ?film ?label
where {
   {?film a movie:Film } OPTIONAL
   {?film rdfs:label ?label }
}
```


Return the films that satisfy the pattern of

- ?film a movie:Film

and return its labels if there are any.

Semantics: $[P_1 \text{ OPT } P_2]_G = [P_1]_G \mathbin{⟕} [P_2]_G$

Query Language: SPARQL

Filter clause: a boolean expression consists of

- RDF-model related operators
  - is Literal(?aNode)
  - isURI(?aNode)
  - ...
- Numeric values comparison
- Literal: regex (from XQuery)


```
select ?film where {
  ?film rdfs:label ?label .
  filter regex( str(?label), "Paddington", "i")
}
```


- xsd:boolean REGEX (string literal text, simple literal pattern, flags)
- flags
  - i: case-insensitive

See `https://www.w3.org/TR/xpath-functions/#regex-syntax` for more information.

Query Language: SPARQL

Filter clause: a boolean expression consists of

- RDF-model related operators
  - isLiteral(?aNode)
  - isURI(?aNode)
  - ...
- Numeric values comparison
- Literal: regex (from XQuery)


```
select ?film where  {
  ?film rdfs:label ?label .
  filter regex( str(?label), "Paddington", "i")
}
```


Semantics: $[P \text{ FILTER } R]_G = \{\mu \in [P]_G \mid \mu \models R\}$

Query Language: SPARQL

The pattern of a SPARQL query can be complicated.

Let $P1$ to $P9$ be triple patterns and R be a regular expression in XQuery


```
{ P1
  P2 }
```

Query Language: SPARQL

The pattern of a SPARQL query can be complicated.

Let $P1$ to $P9$ be triple patterns and R be a regular expression in XQuery.

- Grouping


```
{ { P1
    P2 }
  UNION
  { P3
    P4 }
}
```

Query Language: SPARQL

The pattern of a SPARQL query can be complicated.

Let $P1$ to $P9$ be triple patterns and R be a regular expression in XQuery.

- Grouping
- Optional parts

```
{ { P1
    P2
    OPTIONAL { P5 } }
  { P3
    P4
    OPTIONAL { P7 } }
}
```

Query Language: SPARQL

The pattern of a SPARQL query can be complicated.

Let $P1$ to $P9$ be triple patterns and R be a regular expression in XQuery

- Grouping
- Optional parts
- Nesting

```
{ { P1
    P2
    OPTIONAL { P5 } }
  { P3
    P4
    OPTIONAL { P7
      OPTIONAL { P8 } } }
}
```

Query Language: SPARQL

The pattern of a SPARQL query can be complicated.

Let $P1$ to $P9$ be triple patterns and R be a regular expression in XQuery.

- Grouping
- Optional parts
- Nesting
- Union of patterns

```
{ { P1
    P2
    OPTIONAL { P5 }  }
  { P3
    P4
    OPTIONAL { P7
      OPTIONAL { P8 } }  }
}
UNION
{ P9 }
```

Query Language: SPARQL

The pattern of a SPARQL query can be complicated.

Let $P1$ to $P9$ be triple patterns and R be a regular expression in XQuery.

- Grouping
- Optional parts
- Nesting
- Union of patterns
- Filtering

```
{ { P1
    P2
    OPTIONAL { P5 } }
  { P3
    P4
    OPTIONAL { P7
      OPTIONAL { P8 } } }
}
UNION
{ P9
  FILTER ( R ) }
```

Query Language: SPARQL

Let $P1$ and $P9$ be triple patterns and R be a regular expression in XQuery.

- Graph patterns: full parenthesized algebra

original SPARQL syntax	algebraic syntax
`{ 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 ) }`	$(P_1 \text{ FILTER } R)$

Semantics of SPARQL

Given an RDF graph G.

Let t be a triple pattern and $P1$ and $P2$ be two patterns

Definition

$$
\begin{aligned}
[t]_G &= \\
[P_1 \text{ AND } P_2]_G &= \\
[P_1 \text{ UNION } P_2]_G &= \\
[P_1 \text{ OPT } P_2]_G &=
\end{aligned}
$$

Query Language: SPARQL

Semantics of SPARQL

Given an RDF graph G.

Let t be a triple pattern and $P1$ and $P2$ be two patterns

Definition

$$
\begin{array}{lcl}
[t]_G & = & \text{the subgraphs of } G \text{ that matches } t. \\
[P_1 \text{ AND } P_2]_G & = & [P_1]_G \bowtie [P_2]_G \\
[P_1 \text{ UNION } P_2]_G & = & [P_1]_G \cup [P_2]_G \\
[P_1 \text{ OPT } P_2]_G & = & [P_1]_G \mathbin{⟕} [P_2]_G
\end{array}
$$

Query Language: SPARQL

Solution Modifiers.

- ORDER BY
- LIMIT
- DISTINCT
- PROJECTION
- OFFSET: control where the solutions start from in the overall sequence of solutions
- REDUCED: permit any non-unique solutions to be eliminated


```
prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>
prefix dbpedia-owl: <http://dbpedia.org/ontology/>
prefix movie: <http://data.linkedmdb.org/resource/movie/>

select distinct ?film where {
  { ?film a movie:Film } union
  { ?film a dbpedia-owl:Film }
  ?film rdfs:label ?label .
  filter regex( str(?label), "Paddington", "i")
}
order by ?film
limit 10
```

Query Language: SPARQL

SPARQL can be used to:

- select
- construct an RDF graph `construct{a triple} where{}`
- test whether a query pattern has a solution `ask{}`
- describe a resource with an RDF graph `describe aResource`

Query Language: SPARQL

Example: find 10 films with the keyword of "Paddington" in their labels.

```
prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>
prefix dbpedia-owl: <http://dbpedia.org/ontology/>
prefix movie: <http://data.linkedmdb.org/resource/movie/>

select distinct ?film where {
  { ?film a movie:Film       } union
  { ?film a dbpedia-owl:Film }
  ?film rdfs:label ?label .
  filter regex( str(?label), "Paddington", "i")
}
limit 10
```

Query Language: SPARQL

Example of AND , UNION : find 10 films either with the keyword of "Paddington" in their labels or acted by Tom Cruise.

```
prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>
prefix dbpedia-owl: <http://dbpedia.org/ontology/>
prefix movie: <http://data.linkedmdb.org/resource/movie/>

select distinct ?film where {
  { ?film a movie:Film      } union
  { ?film a dbpedia-owl:Film }
  {?film rdfs:label ?label .
  filter regex( str(?label), "Paddington", "i")} union
  {?film dbo:starring ?actor
  filter regex( str(?actor), "Tom_Cruise", "i")}
}
limit 10
```

Query Language: SPARQL

Example of OPT : find 10 films with the keyword of "Paddington" in their labels and show if they are acted by Hugh Grant.

```
prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>
prefix dbpedia-owl: <http://dbpedia.org/ontology/>
prefix movie: <http://data.linkedmdb.org/resource/movie/>

select distinct ?film ?actor where {
  { ?film a movie:Film      } union
  { ?film a dbpedia-owl:Film }
  {?film rdfs:label ?label .
  filter regex( str(?label), "Paddington", "i")} optional
  {?film dbo:starring ?actor
  filter regex( str(?actor), "Hugh_Grant", "i")}
}
limit 10
```

- Data Model: RDF
- Query Language: SPARQL

Resources:

- Prefixes: `http://dbpedia.org/sparql?help=nsdecl`
- DBpedia: `http://dbpedia.org/sparql`
- W3C: `https://www.w3.org/TR/rdf-sparql-query/`

References

- http://www.iro.umontreal.ca/~lapalme/ift6281/sparql-1_1-cheat-sheet.pdf
- http://webkr.cs.vu.nl/slides/rdfs.pdf
- http://marenas.sitios.ing.uc.cl/talks/pods11.pdf