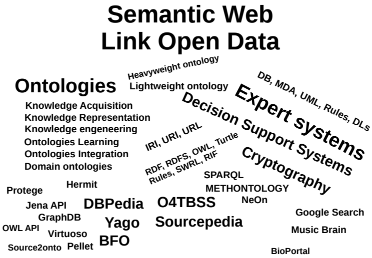
Knowledge engineering

Basic knowledge modeling and RDF graph storage



Dr. Azanzi Jiomekong

University of Yaounde I, Department of Computer Science

10 avril 2023

Questions

- How to represent simple knowledge on the Web ?
- How can we access the data we have already defined ?
- How to store RDF data ?

The responses to these questions are given in this course

RDF Schema Language

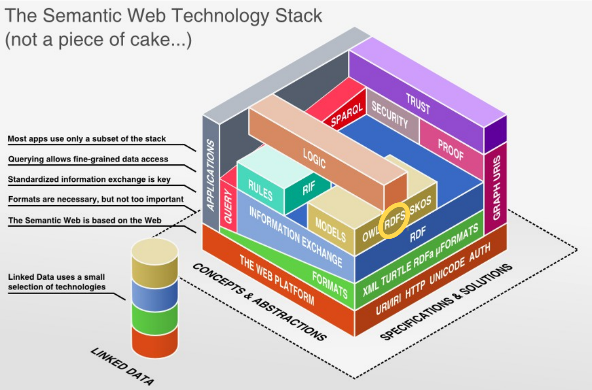
- Question : How to model classes and relations on which we can later make facts about individual and RDF ?
- $\longrightarrow$ Used RDF Schema Language
- Used to define the terminological knowledge of RDF knowledge base / an ontology

RDF Schema Language

Why RDFSchema ?

- An application program can define and use RDF data
- ...if the application program knows, which terms and classes to use, as e.g.
  - name, title, year, ...
  - name, blog, phone number, ...
  - author, cities, ...
- But...
  - Are all terms known ?
  - Are all terms correct ?
  - Are there (logical) relations among the terms ?
- We need a language for data definition $\longrightarrow$ **RDF Schema** officially called **"RDF Vocabulary Description Language"**

RDF Schema Language



RDFSchema

- Defines a data model for the definition of simple ontology
- Used to express ontologies RDF statement (facts)
- More than XML :
  - (small) ontological agreement about modeling primitives
  - Possibility to define own vocabularies

RDFSchema

- First W3C draft in April 1998,
  W3C Recommendation Feb. 2004.

- Defines a **data model** for the creation of RDF statements

- Allows :
  - Definition of **classes**
    Class instanciation in RDF via $< \mathit{rdf} : \mathit{type} >$
  - Defintion of **properties** and **restriction**
  - Definition of **hierarchies**
    Subclasses and Superclasses
    Subproperties and Superproperties

RDF Schema Language

Vocabulary

- Everything is a resource
  - *rdfs* : *resource* is a kind of a class
  - Everything is subclass of resource
  - *rdfs* : *class* is subclass of resource
  - *rdfs* : *class* is defined to be an *rdfs* : *class*
  - *rdfs* : *literal* which is a class but is also a subclass of resource
  - *rdf* : *property* is a subclass of resource and might be a class
  - Lot of properties are predefined : *rdfs* : *range*, *rdfs* : *domain*, etc.

RDF Schema Language

Vocabulary

- **rdfs :Class**
  Concepts of a class, defines an abstract object and is applied (with rdf :type) to create instances

- **rdf :Property**
  Base class for properties

- **rdfs :Literal**
  Class for literals

RDF Schema Language

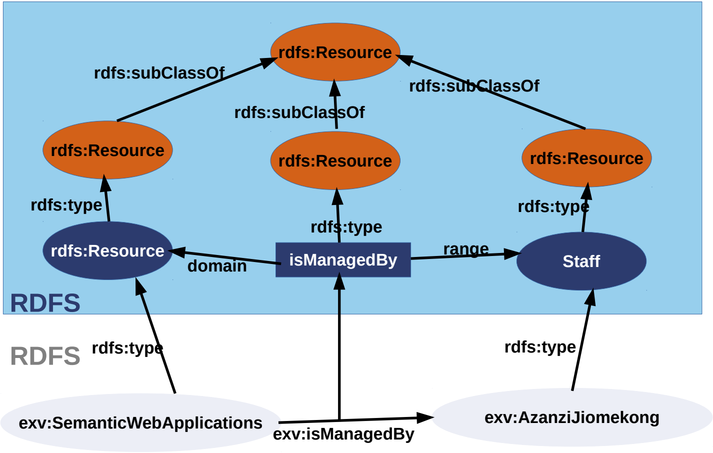
Vocabulary

- **rdfs :Resource**
  every entity of an RDF model is instance of this class

- Additionally
  rdfs :Datatype, **rdf :XMLLiteral**, **rdfs :Container**, **rdfs :ContainerMembershipProperty**

RDF Schema Language

Vocabulary



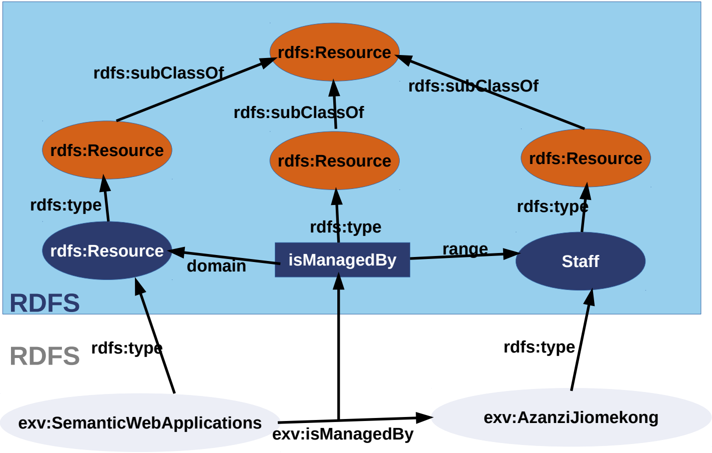
RDF Schema Language

Vocabulary : Properties

- **rdfs :subClassOf**
  Transitive property to define inheritance hierarchies for classes
- **rdfs :subProperty**
  Transitive property to define inheritance hierarchies for properties
- **rdfs :domain**
  defines the domain of a property concerning a class
- **rdfs :range**
  Defines range of a property concerning a class

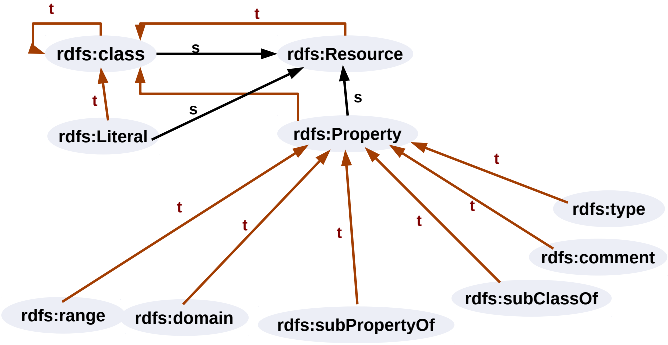
RDF Schema Language

Vocabulary



RDF Schema Language

Language Model



s: subclass relation

t: instance relation

RDF Schema Language

Vocabulary : further properties

- **rdfs :seeAlso**
  Defines a relation of a resource to another, which explains it

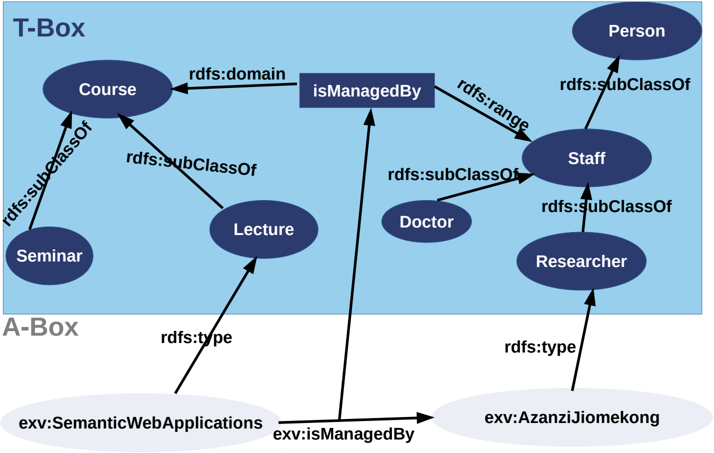
- **rdfs :isDefinedBy**
  subproperty of **rdfs :seeAlso**, defines the relation of a resource to its definition

- **rdfs :comment**
  Comment, usually as text

- **rdfs :label**
  Readable name of a resource (contrary to ID)

RDF Schema Language

Knowledge base example



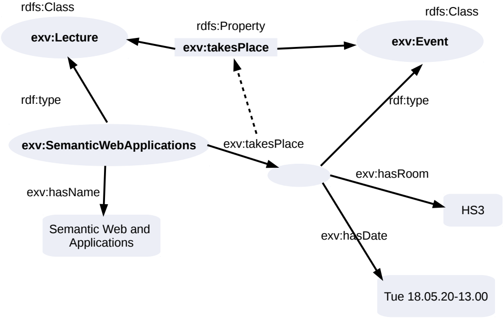
RDF Schema Language

Knowledge base example

```
@prefix rdf:<http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs:<http://www.w3.org/2000/01/rdf-schema#> .

:Course a rdfs:Class.
:Lesture a rdfs:Class;
       rdfs:subClassOf :Course.
:Seminar a rdfs:Class;
       rdfs:subClassOf :Course.

:Person a rdfs:Class.
:Staff a rdfs:Class;
       rdfs:subClassOf :Person.
:Researcher a rdfs:Class;
       rdfs:subClassOf :Staff.
:Doctor a rdfs:Class;
       rdfs:subClassOf :Staff.

:isManagedBy a rdf:Property;
       rdfs:domain :Course;
       rdfs:range :Staff.

:SemanticWebApplications a :Lecture.
:AzanziJiomekong a :Researcher.
:SemanticWebApplications :isManagedBy :AzanziJiomekong.
```

Semantics is the in RDF and RDFS ?

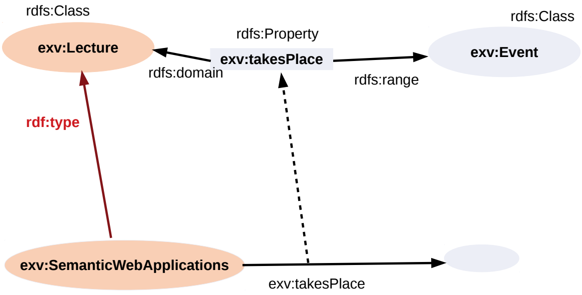
Let's consider the following knowledge base



The semantics of a term from an RDF(S) ontology is given in terms of its properties and its values (instances)

Semantics is the in RDF and RDFS

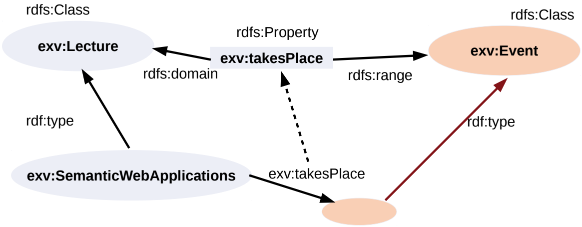
What conclusions can we deduce with RDF(S) ?



- Deduction of entity class membership from the domain of one of its properties
- We can deduce that Semantic Web Technologies must be a lecture

Semantics is the in RDF and RDFS

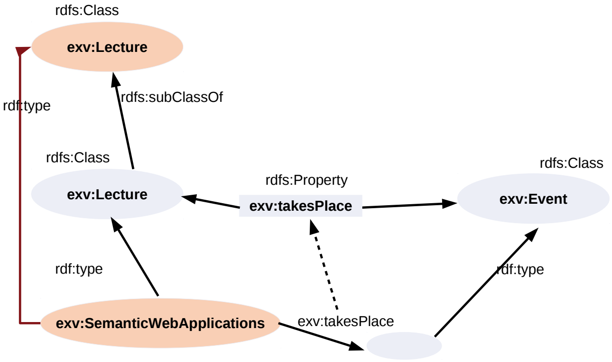
What conclusions can we deduce with RDF(S) ?



- Deduction of entity class membership from the range of its properties
- We can deduce that the blank node must represent an event

Semantics is the in RDF and RDFS

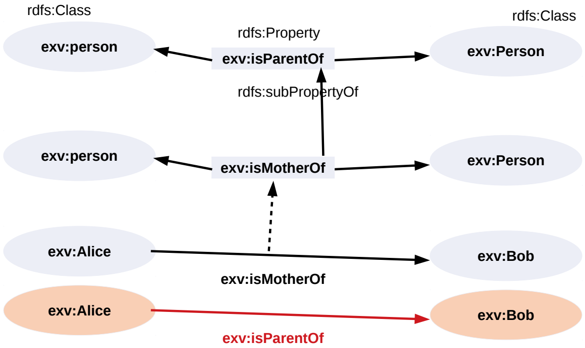
What conclusions can we deduce with RDF(S) ?



Deduction of entity superclass membership from a class hierarchy

Semantics is the in RDF and RDFS

What conclusions can we deduce with RDF(S) ?



Deduction of new facts from subproperty relationships

RDF Schema Language

- Small vocabulary
- We don't have something like a negation
- It is rather difficult to create inconsistencies and failure
- Everything define with RDFS is potentially true
- What is missing is the formal vocabulary
- For machine to understand, we need a formal representation with logic

Question

How to store RDF triples to carry out efficient SPARQL queries ?

Storing RDF data

RDBMS

- One column for the subject, one column for the property and one column as the object
- Convert the SPARQL query into SQL query
- Generate the SPARQL result from the SQL result
- Make the SPARQL endpoint on the top of the graph database

Storing RDF data

RDBMS

- RDF graphs can be stored in a set of triples (s, p, o)
- Triples can be stored in a RDBMS
- Three steps for SPARQL query processing :
  - Convert SPARQL query to SQL query
  - Use RDBMS to answer SQL query
  - Generate SPARQL query result from SQL query result

Storing RDF data

Five ways to encode RDF triples

- Giant Triple Store
- ID Based Triple Store
- Quad Tables
- Properties Tables
- Vertically Partitioned Tables (Binary Tables)
- Hexastores

Storing RDF data

Giant Triple Storage

- Basic idea
  - Store all RDF triples in a single table
  - Translate SPARQL query into SQL query
  - Performance depends on efficient indexing

- Advantages : easy to implement, works for huge numbers of properties is indexes are chosen with care

- Disadvantage : many self joins

Storing RDF data

Giant Triple Storage

Subject	Property	Object
ID1	type	FullProfessor
ID1	teacherOf	INFO201
ID1	bachelorFr	UDS
ID1	mastyerFr	UY1
ID1	phdFr	UY1
ID2	type	AssocProf
ID2	worksFor	UY1
ID2	teacherOf	INF4288
ID2	bachelorFr	UMA
ID2	phdFr	UY1
ID3	type	GradStudent
ID3	advisor	ID2
ID3	teachAss	INFO201
ID3	bachelorFr	UY1
ID3	mastyerFr	UY1
ID4	type	GradStudent
ID4	advisor	ID1
ID4	TakesCour	SWA
ID4	bachelorFr	UMA

Storing RDF data

ID Based Triple Storage

- Use numerical identifier for each RDF term in the dataset
- Saves space and enhances efficiency

RDF Term	ID
:ID1	1
rdf:type	2
:FullProfessor	3
:teacherOf	4
‘‘SWA’’	5
:bachelorFrom	6
”UDS”	7
:masterFrom	8
”UY1”	9

s	p	o
1	2	3
1	4	5
1	6	7
1	8	9
1	10	11
12	13	14
12	15	7
12	4	16

Storing RDF data

Quad Tables

- Storing **multiple** RDF graphs
- Used for provenance, versioning, contexts, etc.

RDF Term	ID
:ID1	1
rdf:type	2
:FullProfessor	3
:teacherOf	4
‘’SWA’’	5
:bachelorFrom	6
’’UDS’’	7
:masterFrom	8
’’UY1’’	9

s	s	p	o
100	1	2	3
100	1	4	5
101	1	6	7
100	1	8	9
102	1	10	11
100	12	13	14
102	12	15	7
100	12	4	16

Storing RDF data

Properties Tables

- Combining all (or some) properties of similar subjects in n-ary tables
- Use ID based encoding for efficiency

Professors

ID	type	teacherOf	bachelorFrom	masterFrom	phdFrom	worksFor
ID 1	FullProfessor	"Algo"	"UY1"	"UY1"	"UMA"	NULL
ID 2	AssocProfessor	"OOP"	"UMA"	NULL	"UDA"	UY1

Students

ID	type	advisor	bachelorFrom	masterFrom	teachingAss	takesCourses
ID 3	GradStudent	ID2	UDA	UY1	SWA	NULL
ID 4	GradStudent	ID1	UDA	NULL	"NULL"	"OOP"

Storing RDF data

Properties Tables

Advantages :

- Fewer joins
- if the data is structures, we have a relational DB

Disadvantages

- Potentially a lot of NULLs
- Clustering is not trivial
- Multi-value properties are complicated

Storing RDF data

Vertically Partitioned Tables (Binary Tables)

- For each unique property create a two column table
- Use ID based encoding for efficiency

type	
ID 1	FullProfessor
ID 2	AssocProfessor
ID 3	GradStudent
ID 4	GradStudent

teacherOf
"Algo"
"OOP"

bachelorFrom	
ID 1	"UY1"
ID 2	"UMA"
ID 3	"UY1"
ID 4	"UDA"

masterFrom	
ID 1	"UY1"
ID 3	"UY1"

phdFrom	
ID 1	"UMA"
ID 2	"UDA"

worksFor	
ID2	"UY1"

advisor	
ID3	ID2
ID4	ID1

bachelorFrom	
ID 1	"UY1"
ID 2	"UMA"
ID 3	"UDA"
ID 4	"UDA"

teachingAss	
ID3	"SWA"

takesCourses	
ID4	"OOP"

Storing RDF data

Vertically Partitioned Tables (Binary Tables)

Advantages :

- Support multi-value properties
- No NULLs
- Read only needed attributes (i.e. les I/O)
- No clustering
- Excellent performance (if number of properties is small, queries with bounded properties)

Disadvantages :

- Expensive inserts
- Bad performance (large number of properties, queries with unbounded properties)

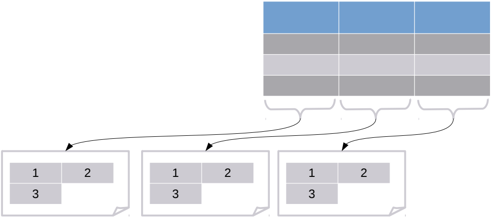
Storing RDF data

Vertically Partitioned Tables (Binary Tables)

- Different physical storage models for relational DBs
- Two types of storage : Row Based Storage and Column Based Storage

Row Based Storage :

- Tuples (i.e. DB records) are stored consecutively
- Entire row needs to be read even if few attributes are projected

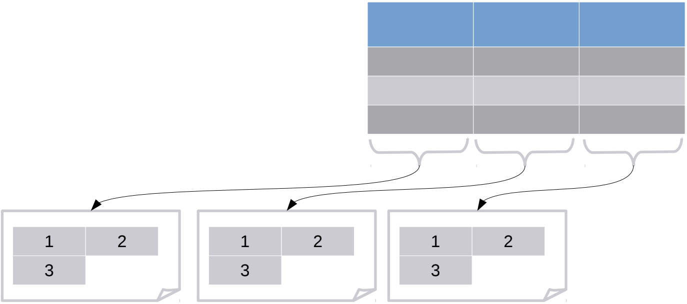


Storing RDF data

Vertically Partitioned Tables (Binary Tables)

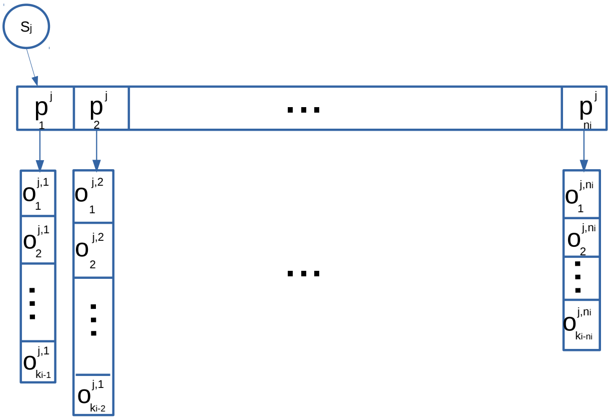
Column Based Storage :

- Read columns relevant to the query
  $\longrightarrow$ projection is free
- Inserts are expensive



Storing RDF data

Hexastores



Storing RDF data

Hexastores

- Creates an index for every possible combination to enable efficient processing
- spo, pos, osp, sop, pso, ops

The **spo** index :

- Suject key S_i points to sorted vector of n_i property keys, $\{p_1{}^j.p_2{}^j.\ \ldots\ p_{n_1}{}^j\}$
- Each property key $p_i{}^j$ is linked to a sorted list of objects keys
- The object key lists are shared with the pso index

Advantage : fast joined (in the beginning)

Disadvantages :

- 5 times more storage
- weak performance, when disk access is necessary

Storing RDF data

Triple Stores

- AnzoGraph - https://docs.cambridgesemantics.com/anzograph/v2.2/userdoc/home.htm
- Apache Jena TDB -
- Open Link Virtuoso -
- AllegroGraph -
- GraphDB -
- Stardog -

W3C maintains a list of triple stores : https://www.w3.org/wiki/LargeTripleStores

Limits of RDF and RDFS

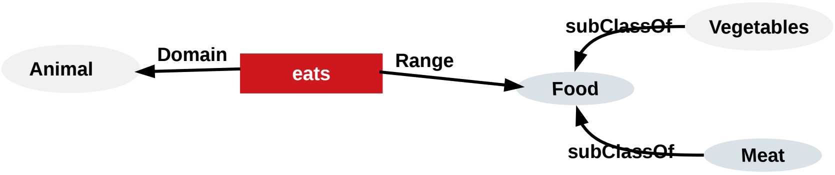
- RDF and RDFS are sufficient to represent the knowledge of the world ?

- What can be represented and what can't be represented ?

- How can we represent the following knowledge :
  - "Certain animals only eat meat. Animal like Cow only eats vegetables. Certain animals eat vegetables and meats" ? $\longrightarrow$ Locality of global properties
  - "Man and human are subclasses of Human. A man is not a woman" $\longrightarrow$ Disjunctive Classes
  - "Human has two parents" $\longrightarrow$ Restrictions on properties - Cardinality restriction

Limits of RDF and RDFS

Locality of global properties

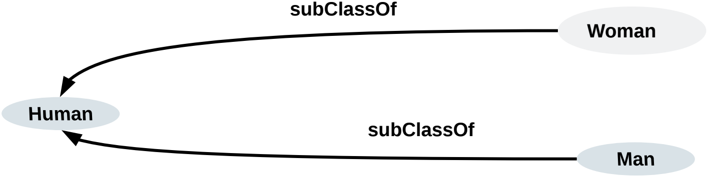


Problem

- Cows only eat vegetables
- Other animals also eat meat

Limits of RDF and RDFS

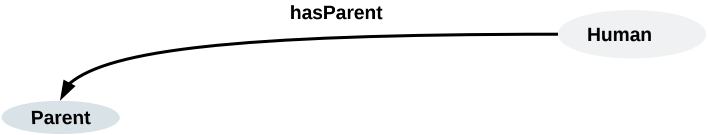
Disjunctive Classes



Problem : Subclass relation cannot express disjunctive class (subclass) merbership

Limits of RDF and RDFS

Cardinality Restrictions



Problem : Every human has two parents

Limits of RDF and RDFS

Special Property Constraints

- **Transitivity** (e.g. "is greater than")
- **Uniqueness** (e.g. "is mother of")
- **Inversiveness** (e.g. "is parent of" and "is child of")

Summary

- We started with the URI to identify things
- The HTTP protocol to exchange messages
- How to create facts how to formulate facts and express facts in RDF
- Models from which these facts can be made in RDFS
- Store RDF data

Question

How to solve the limits of RDF and RDFS ?