RDFS - Resource Description Framework Schema

Outline

- First some basic information about RDFS
- Some notes about the notation i will use in the examples
- Some important RDFS Classes
- Some important RDFS Properties (w/examples)
- That’s it!

RDF Schema

- RDF Schema or RDFS is a language that can be used to define the vocabulary (i.e., the terms) to be used in an RDF graph.
- RDFS is a simple language - we can only define taxonomies and do some basic inference about them.

- Many RDFS components are included in the more expressive Web Ontology Language (OWL).

Notation

- For RDF statements i will use notation like this: (Subject, Predicate, Object) (omitting the rdfs or rdf prefix)
- Class names will be written with an initial uppercase letter, while properties and instance names are written with an initial lowercase letter.
- Defining classes and properties is done implicitly. Instead of showing

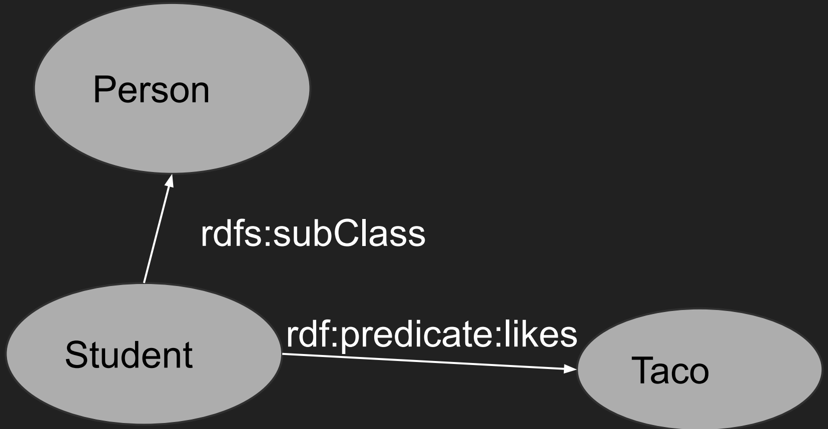
  (Person, rdf:type, Class) or

  (hasParent, rdf:type, Property) everywhere i will just use the Class or Properties without further ado.

What is RDFS?

Recall: RDF defines the structure of the data - (Student, likes, Taco)

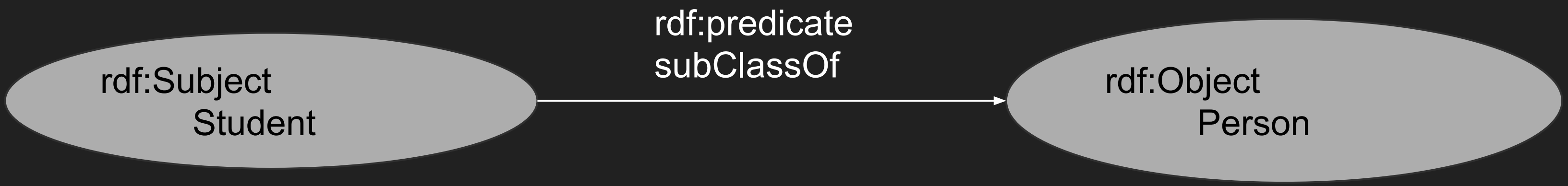
RDFS defines the semantic relationships of RDF. It means that it enables us to add metadata about RDF. For example we can use the RDFS property rdfs:subClassOf to describe the RDF subject in the example below. (Student is a subclass of Person)



What is RDFS? (cont.)

RDFS is simply a vocabulary for describing RDF - written in RDF!

In my previous example rdfs:subClassOf is the rdf:predicate in the RDF statement.



RDFS classes

Classes:

- rdfs:Resource - class of everything. All things described by RDF are resources.
- rdfs:Class - is recursive and is the class of Classes
- rdfs:Properties - the class of properties
- rdfs:Literal – literal values such as strings and integers. Property values such as textual strings are examples of RDF literals. Literals may be plain or typed.
- rdfs:Datatype – the class of datatypes. rdfs:Datatype is both an instance of and a subclass of rdfs:Class. Each instance of rdfs:Datatype is a subclass of rdfs:Literal.

RDFS Properties

- rdfs: subClassOf - as we already talked about. Enables us to create subclasses and hierarchies of classes.
- rdfs:subPropertyOf - works in the same way as subClassOf, but for properties.

Example: If we have the RDF statements:

(john, primaryDriver, CompanyCar)
(primaryDriver, subPropertyOf, driver)

-> We can then infer: (john, driver, CompanyCar).

In other words; Since John has the subProperty primaryDriver, John will also inherit the superProperty (driver).

RDFS properties (cont.)

- rdfs: Domain - declares the class of the subject in a triple whose predicate is that property
- rdfs: Range - declares the class or datatype of the object in a triple whose predicate is that property.

Example - if we have the RDF statements:

(hasMother, rdfs:range, Female)

(hasMother, rdfs:range, Person)

(frank, hasMother, maria)

-> We can infer that Maria is a Female and a Person, since the range of the hasMother property is Female and Person.

Likewise, if we set the domain of the hasMother property to Male. Only Males could have a mother.. Which is not logically correct, but perfectly legal to model. If we set the domain of the hasMother property to Male and Female it would make more sense.

- When setting domain we set constraints on the subject class.
- When setting range we set the domain of the object class.

Example where RDFS has shortcomings

We want to model that the parent of a Human is also a Human. And that the parent of a Tiger is also a Tiger. We set the domain and range properties on the hasParent property for both the Human and the Tiger Class. We then model that Tina's parent is John.

(hasParent, rdfs:domain, Human)

(hasParent, rdfs:range, Human)

(hasParent, rdfs:domain, Tiger)

(hasParent, rdfs:range, Tiger)

(tina, hasParent, john)

The problem is that Tina and John are then inferred to be both humans and tigers. We have not said anywhere that humans cannot be tigers and vice versa (nor can we say this in RDFS). (This is where OWL comes in).

RDFS Utility properties

RDFS also have some other properties that can be useful to add further details.

- rdfs:seeAlso
- rdfs:label
- rdfs: comment
- rdfs: isDefinedBy

They are quite self-explanatory.