

# **192.161 Management of Graph Data**

**(4.0 VU / 6.0 ECTS)**

**2025W**

**RDF**

**Katja Hose**  
**Maxime Jakubowski**

*[mogda@list.tuwien.ac.at](mailto:mogda@list.tuwien.ac.at)*

- RDF and Linked Data
- RDFS

# The Semantic Web



## How the journey began...

- Expressing meaning
- Knowledge representation
- Ontologies
- Agents
- Evolution of knowledge

### The Semantic Web

A new form of Web content that is meaningful to computers will unleash a revolution of new possibilities

by [TIM BERNERS-LEE](#), [JAMES HENDLER](#) and [ORA LASSILA](#)

The entertainment system was belting out the Beatles' "We Can Work It Out" when the phone rang. When Pete answered, his phone turned the sound down by sending a message to all the other *local* devices that had a *volume control*. His sister, Lucy, was on the line from the doctor's office: "Mom needs to see a specialist and then has to have a series of physical therapy sessions. Biweekly or something. I'm going to have my agent set up the appointments." Pete immediately agreed to share the chauffeuring. At the



doctor's office, Lucy instructed her Semantic Web agent through her handheld Web browser. The agent promptly retrieved information about Mom's *prescribed treatment* from the doctor's agent, looked up several lists of *providers*, and checked for the ones *in-plan* for Mom's insurance within a *20-mile radius* of her *home* and with a *rating of excellent or very good* on trusted rating services. It then began trying to find a match between available *appointment times* (supplied by the agents of individual providers through their Web sites) and Pete's and Lucy's busy schedules. (The emphasized keywords indicate terms whose semantics, or meaning, were defined for the agent through the Semantic Web.)

In a few minutes the agent presented them with a plan. Pete didn't like it—University Hospital was all the way across town from Mom's place, and he'd be driving back in the middle of rush hour. He set his own agent to redo the search with stricter preferences about *location* and *time*. Lucy's agent, having *complete trust* in Pete's agent in the context of the present task, automatically assisted by supplying access certificates and shortcuts to the data it had already sorted through.

### Killer app: Agents

- Retrieve information about prescribed treatment/therapy
- Look for providers/specialists with good ratings in proximity
- Schedule times according to multiple busy schedules
- Share information with other agents and “negotiate”

“A really important thing about data is:  
the more things you have to **connect**  
together, the more powerful it is.”



MIT News

ON CAMPUS AND AROUND THE WORLD



Tim Berners-Lee was honored with the Turing Award for his work inventing the World Wide Web, the first web browser, and "the fundamental protocols and algorithms [that allowed] the web to scale."

Photo: Henry Thomas

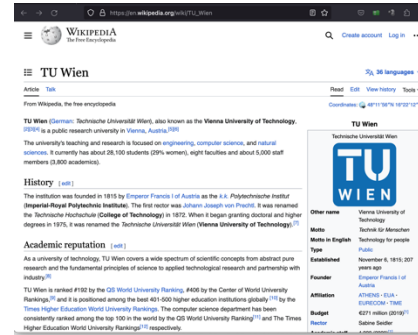
## Tim Berners-Lee wins \$1 million Turing Award

CSAIL researcher honored for inventing the web and developing the protocols that spurred its global use.



## The Web of Documents

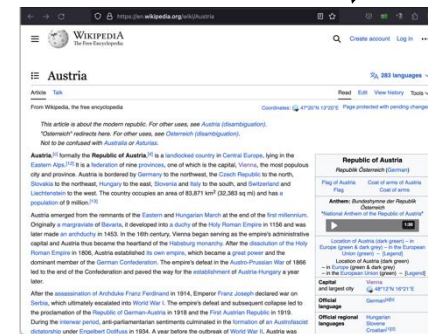
TU Wien



Vienna



Austria

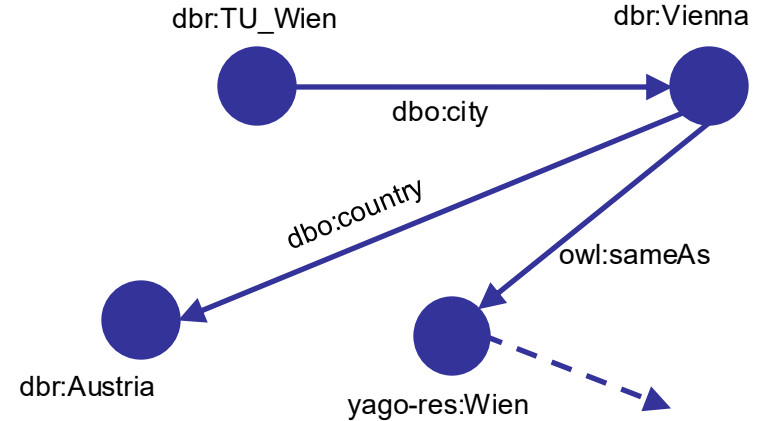


- Links between documents (websites, social media, etc.)
- Human-readable knowledge
- Normally accessed by humans via a Web browser
- Links are not qualified

## The Web of Data

### Linked Data

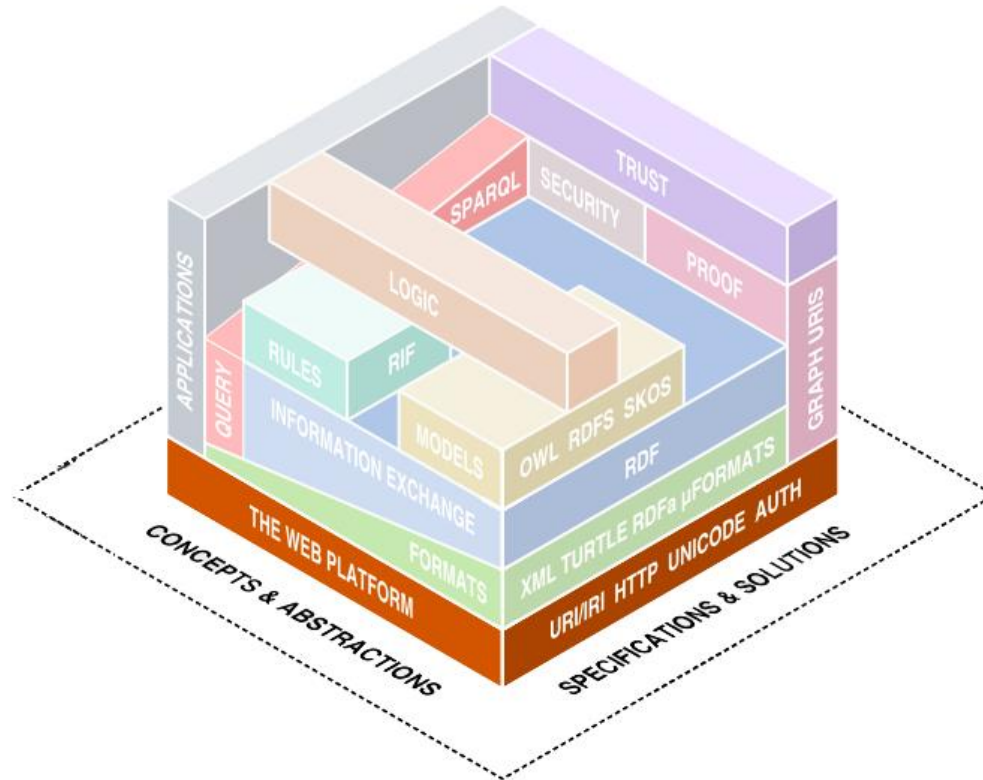
- Links between things/entities
- Machine-readable knowledge
- Normally accessed by machines
- Links are qualified
- A decentralized database



# Web Technology Foundations



The Semantic Web is based on the Web



## **TCP/IP (Transfer Control Protocol/Internet Protocol):**

protocols for data transfer via a network

## **HTTP (Hypertext Transfer Protocol):**

- application protocol to transfer information (not just) on the WWW
- request/response protocol between client and server
- content negotiation (header lists acceptable media types etc.)

## **HTML (Hypertext Markup Language):**

- language for the creation of web pages
- describes how it should be displayed by a web browser

## **XML (Extensible Markup Language):**

- general purpose markup language for the creation of special markup languages
- used to facilitate data sharing among different systems

## Basic principles

- “Everything has an identifier”
- “One name for one resource”

**Uniform** (different types of identifiers constructed according to a uniform schema)

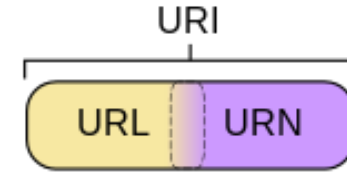
**Resource** (whatever may be identified via URI)

(e.g., web pages, books, locations, persons, relations among objects, abstract concepts etc.)

**Identifier** (to distinguish one resource from another)

## Examples:

```
ldap://[2001:db8::7]/c=GB?objectClass?one
mailto:John.Doe@example.com
news:comp.infosystems.www.servers.unix
tel:+1-816-555-1212
telnet://192.0.2.16:80/
urn:oasis:names:specification:docbook:dtd:xml:4.1.2
http://dbpedia.org/resource/Vienna
```



URI: Uniform Resource can be

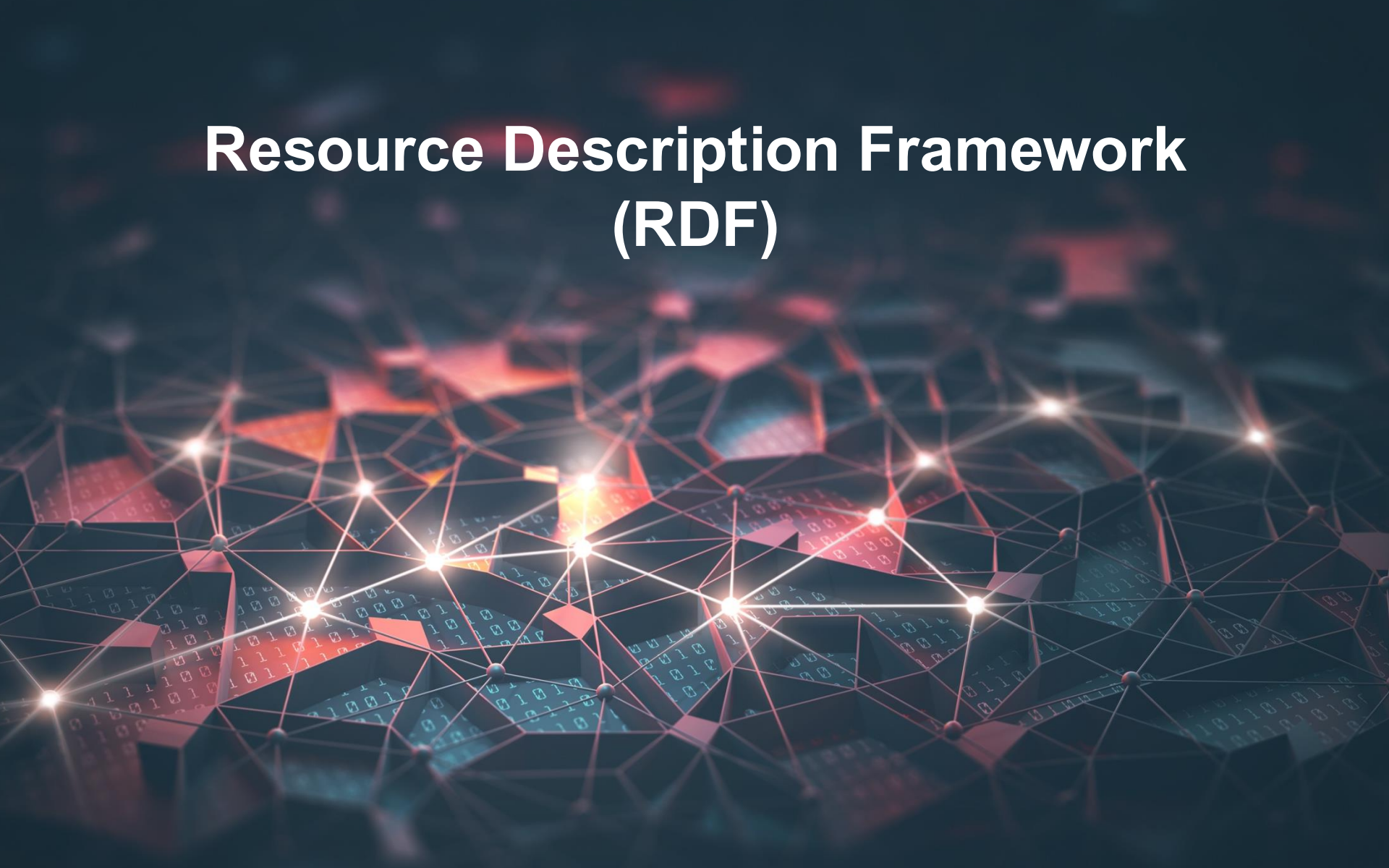
- **URN (RFC 2141):** Uniform Resource Name
  - persistent identifier within a particular namespace
  - Refers to a resource without implying location or how to access it
  - e.g., urn:isbn:0-486-27557-4 refers to a specific edition of Shakespeare's Romeo and Juliet
- **URL (RFC 1738):**
  - Uniform Resource Locator
  - specifies (primary) access mechanism
  - may change during the lifetime of a resource
- **both URN+URL**

IRI (Internationalized Resource Identifier, RFC 3987) are URIs

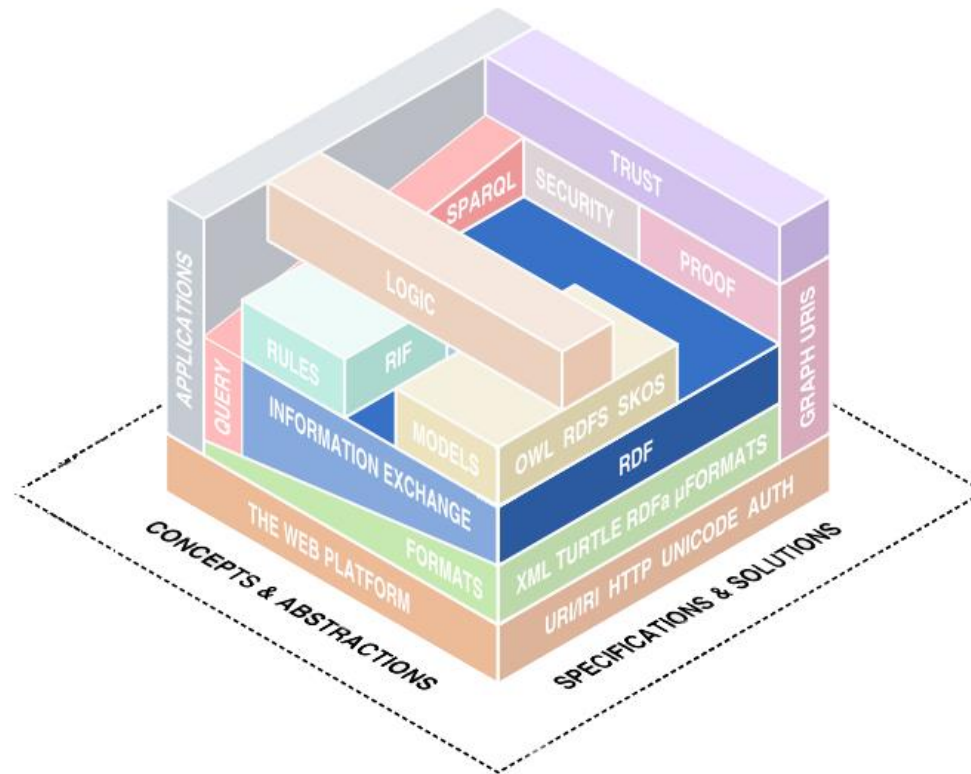
- Character set of IRIs: Unicode (ISO 10646)
- Character set of URIs: US-ASCII

In the remainder of this presentation occurrences of the term URI can be replaced with IRI.

# Resource Description Framework (RDF)



Standard information exchange is key



- **Resource Description Framework** (RDF, W3C standard)
- Basic concepts
  - A **resource** can basically be anything
    - e.g. persons, places, Web documents, abstract concepts
  - **Descriptions** of resources
    - Attributes
    - Relationships
  - The **framework** contains
    - A data model, and
    - Languages and syntaxes

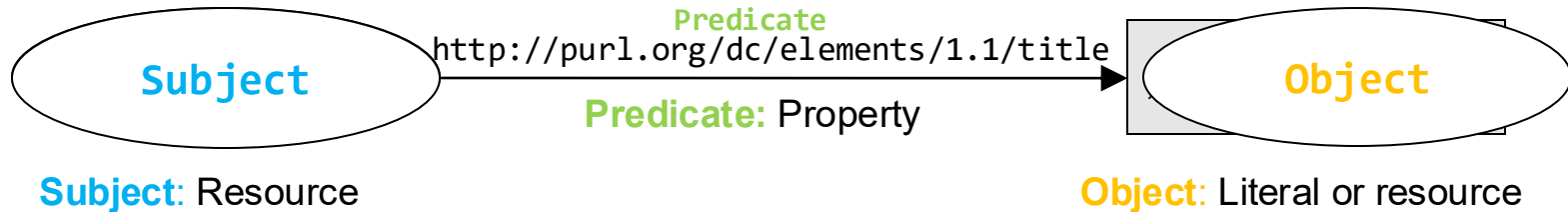
- Developed in the late 1990s
- Final W3C recommendation in 2004 (RDF 1.1)
- Graph-based model
- Formal semantics → machine-readable
- Knowledge expressed as a list of statements
- All statements follow the same simple schema (**RDF Triple**)
- Abstract model with various syntax notations and serialization formats:
  - Graph notation
  - RDF/XML (historically first serialization format)
  - N3/Turtle
  - N-Triples
  - JSON-LD



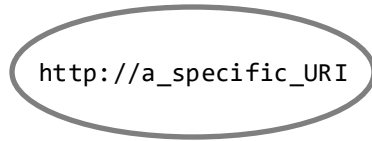


### Basic RDF building block

- **Subject**: a resource, which may be identified with a URI
- **Predicate**: a URI-identified specification of the relationship between subject and object
- **Object**: a resource or literal to which the subject is related

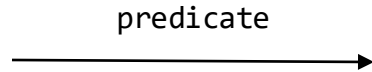


# Graph Notation Elements



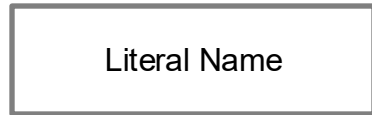
## Resource

Has a unique URI



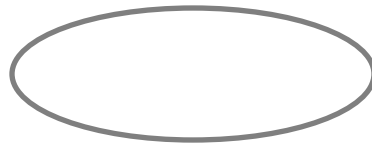
## Predicate

Connects two resources



## Literal

- Only text, no URI
- Plain (lexical value and optional language tag) or typed

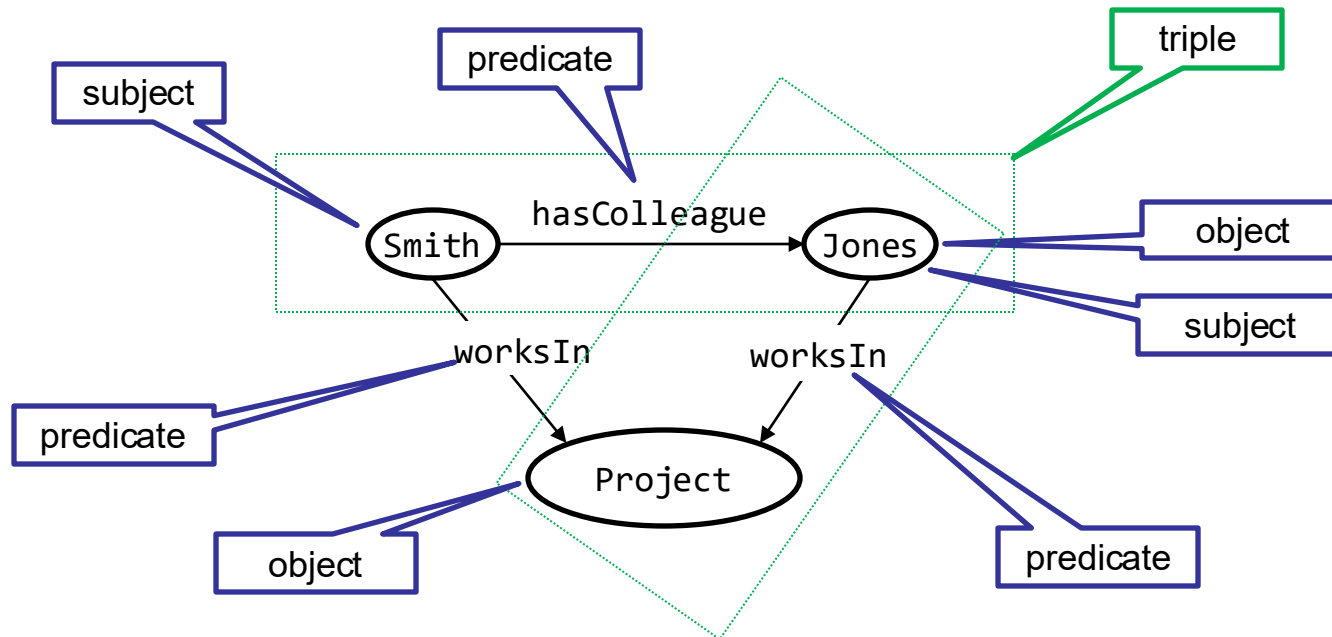


## Blank Node

- No URI
- For unnamed objects

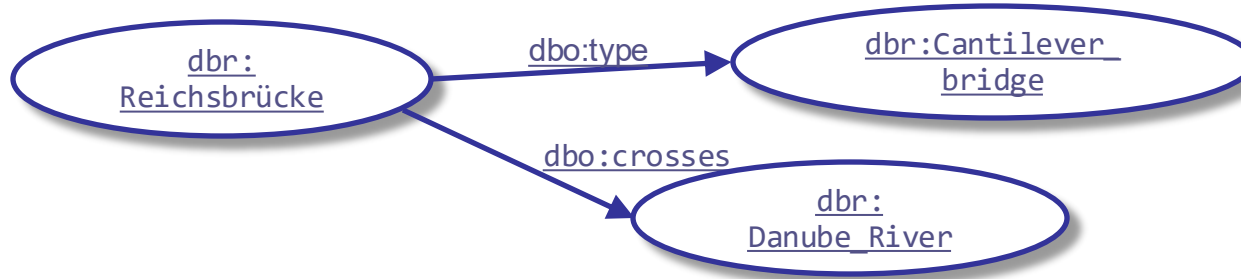
A set of RDF assertions in triples form a labeled directed graph with:

- **Resources:** the subjects and objects are nodes of the graph
- **Predicates:** each predicate becomes a label for an arc connecting the subject to the object.



!

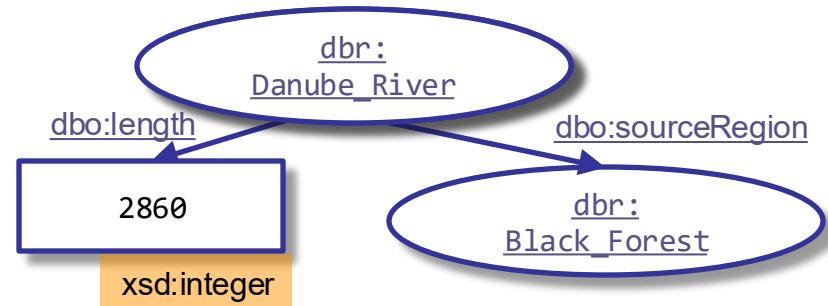
The object of one statement can be the subject of another.



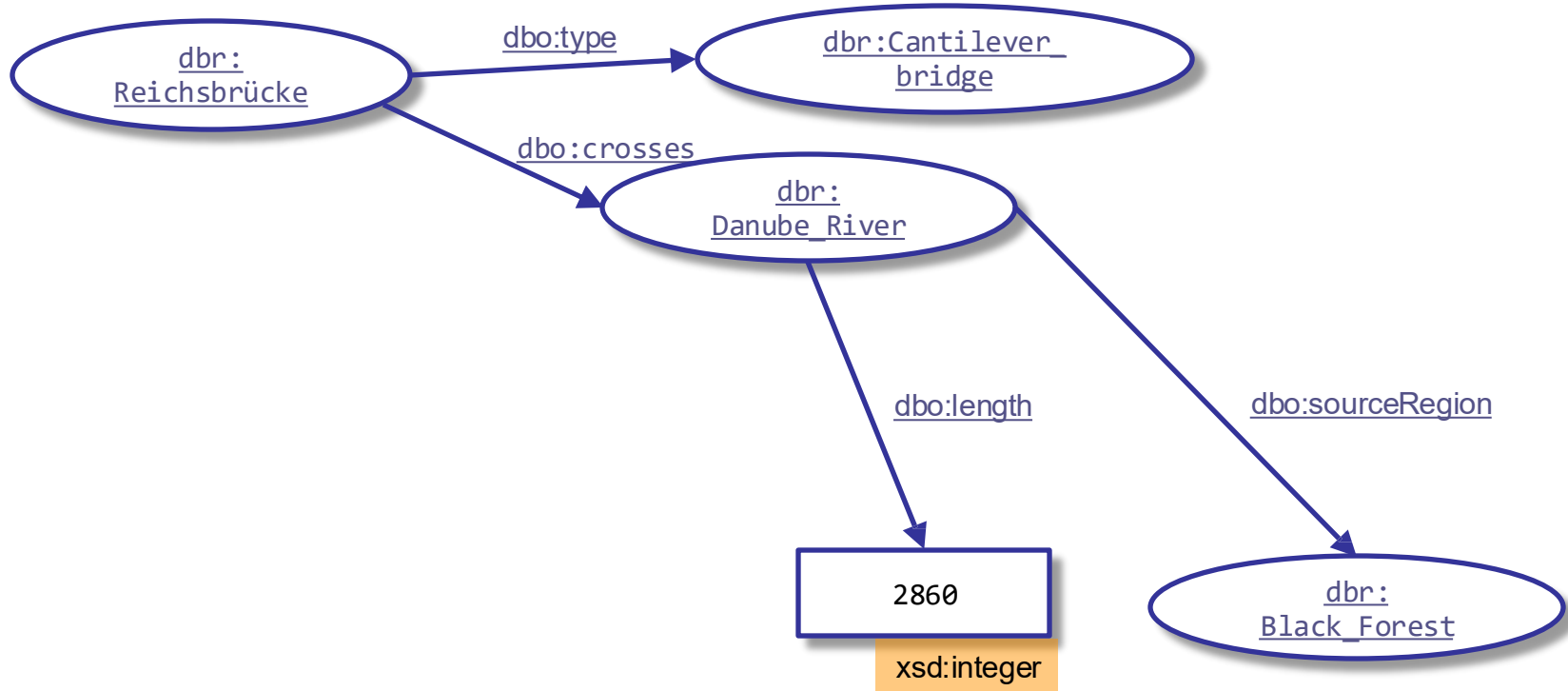
<http://dbpedia.org/resource/Reichsbrücke>

becomes

dbr:Reichsbrücke



# Merging/Integrating Multiple Graphs



Plain literals have a **lexical form** (their lexical value) and optionally a **language tag**.

“27” , “Hello world”@en

- Formed by pairing a string with a URIref that identifies a particular datatype
- Indicates what datatype should be used to interpret a given literal
- RDF has no built-in set of datatypes of its own (except `rdf:XMLLiteral`)
- RDF datatype concepts are based on a conceptual framework from **XML Schema datatypes** that defines
  - the value space,
  - the lexical space and
  - the lexical-to-value mapping for a datatype (see RDF specifications)

```
"27"^^http://www.w3.org/2001/XMLSchema#integer
```

# XML Schema datatypes

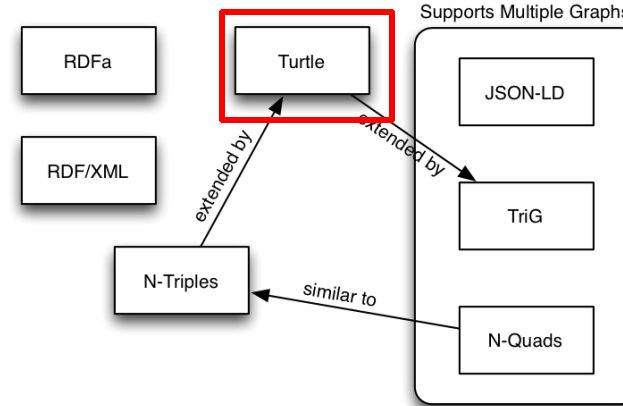
	Datatype	
Core types	<a href="#">xsd:string</a>	<a href="#">xsd:byte</a>
	<a href="#">xsd:boolean</a>	<a href="#">xsd:short</a>
	<a href="#">xsd:decimal</a>	<a href="#">xsd:int</a>
	<a href="#">xsd:integer</a>	<a href="#">xsd:long</a>
		<a href="#">xsd:unsignedByte</a>
IEEE floating-point numbers	<a href="#">xsd:double</a>	<a href="#">xsd:unsignedShort</a>
	<a href="#">xsd:float</a>	<a href="#">xsd:unsignedInt</a>
Time and date	<a href="#">xsd:date</a>	<a href="#">xsd:unsignedLong</a>
	<a href="#">xsd:time</a>	<a href="#">xsd:positiveInteger</a>
	<a href="#">xsd:dateTime</a>	<a href="#">xsd:nonNegativeInteger</a>
	<a href="#">xsd:dateTimeStamp</a>	<a href="#">xsd:negativeInteger</a>
Recurring and partial dates	<a href="#">xsd:gYear</a>	<a href="#">xsd:nonPositiveInteger</a>
	<a href="#">xsd:gMonth</a>	<a href="#">xsd:hexBinary</a>
	<a href="#">xsd:gDay</a>	<a href="#">xsd:base64Binary</a>
	<a href="#">xsd:gYearMonth</a>	<a href="#">xsd:anyURI</a>
	<a href="#">xsd:gMonthDay</a>	<a href="#">xsd:language</a>
	<a href="#">xsd:duration</a>	<a href="#">xsd:normalizedString</a>
	<a href="#">xsd:yearMonthDuration</a>	<a href="#">xsd:token</a>
	<a href="#">xsd:dayTimeDuration</a>	<a href="#">xsd:NMTOKEN</a>
Limited-range integer numbers		<a href="#">xsd:Name</a>
		<a href="#">xsd:NCName</a>
Encoded binary data		
Miscellaneous XSD types		

## RDF Serialization Formats



RDF 1.0

RDF 1.1



- Useful for inter-machine communication
- Every **Description** element describes a resource (referred to by an IRI)
- Every attribute or nested element inside a **Description** is a **property** of that Resource

```
<rdf:Description rdf:about="http://musicbrainz.org/artist/b10bbbfc-cf9e-42e0-be17-e2c3e1d2600d#" >
  <foaf:name>The Beatles</foaf:name>
  <owl:sameAs rdf:resource="http://dbpedia.org/resource/The_Beatles" />
</rdf:Description>
```

RDF/XML

**Subject**

**Predicate**

**Object**

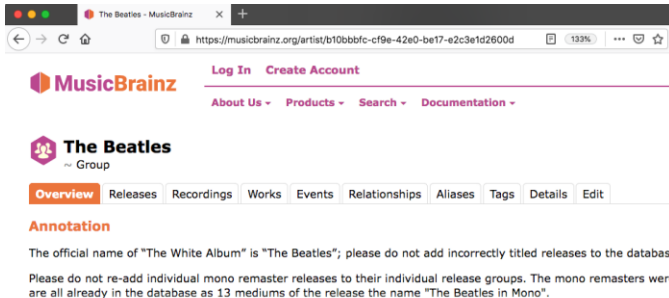
[http://musicbrainz.org/artist/  
b10bbbfc-cf9e-42e0-be17-e2c3e1d2600d#\\_>](http://musicbrainz.org/artist/b10bbbfc-cf9e-42e0-be17-e2c3e1d2600d#_)

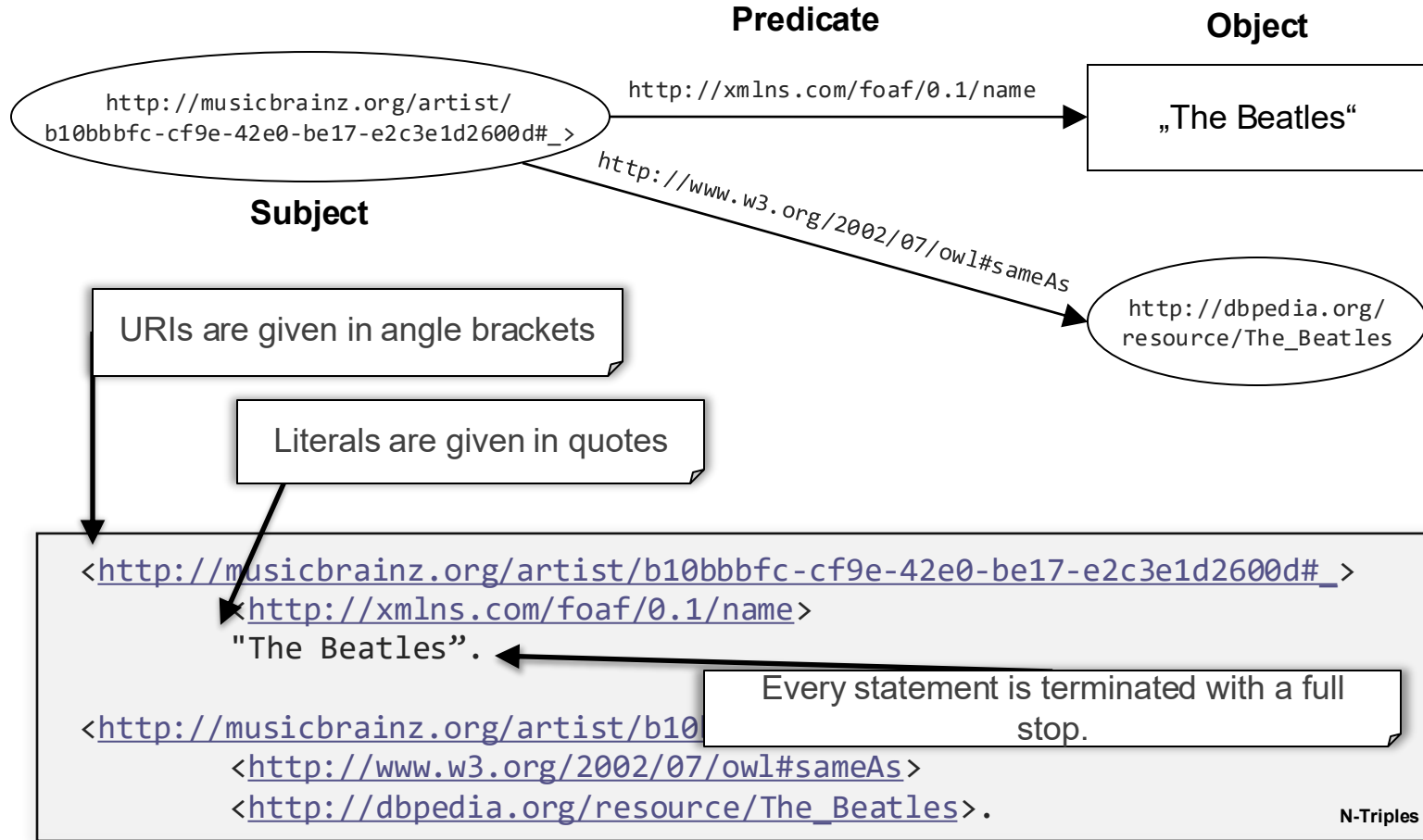
<http://xmlns.com/foaf/0.1/name>

„The Beatles“

<http://www.w3.org/2002/07/owl#sameAs>

[http://dbpedia.org/  
resource/The\\_Beatles](http://dbpedia.org/resource/The_Beatles)





- Easy to read and write
- <subject> <predicate> <object>
- Every subject, predicate, object is identified with a URI
- Two shortcuts for several statements about the same subject
  - “.” introduce another predicate of the same subject
  - “,” introduce another object with the same predicate and subject

```
<#pat> <#child> <#al>, <#chaz>, <#mo>;  
      <#age> 24;  
      <#eyecolor> "blue".
```

N3

**Interpretation:**

Pat has the children al, chaz, mo;

Pat's age is 24;

Pat's eyecolor is blue

- "Terse RDF Triple Language"
- Subset of N3
- Many URIs share the same basis → use prefixes for namespace declarations

```
@prefix rdf:<http://www.w3.org/1999/02/22-rdf-syntax-ns#>.
@prefix rdfs:<http://www.w3.org/2000/01/rdf-schema#>.
@prefix owl:<http://www.w3.org/2002/07/owl#>.
@prefix mo:<http://purl.org/ontology/mo/>.
@prefix dbpedia:<http://dbpedia.org/resource/>.
```

Turtle

Note: You can find commonly used prefixes here: <http://prefix.cc>

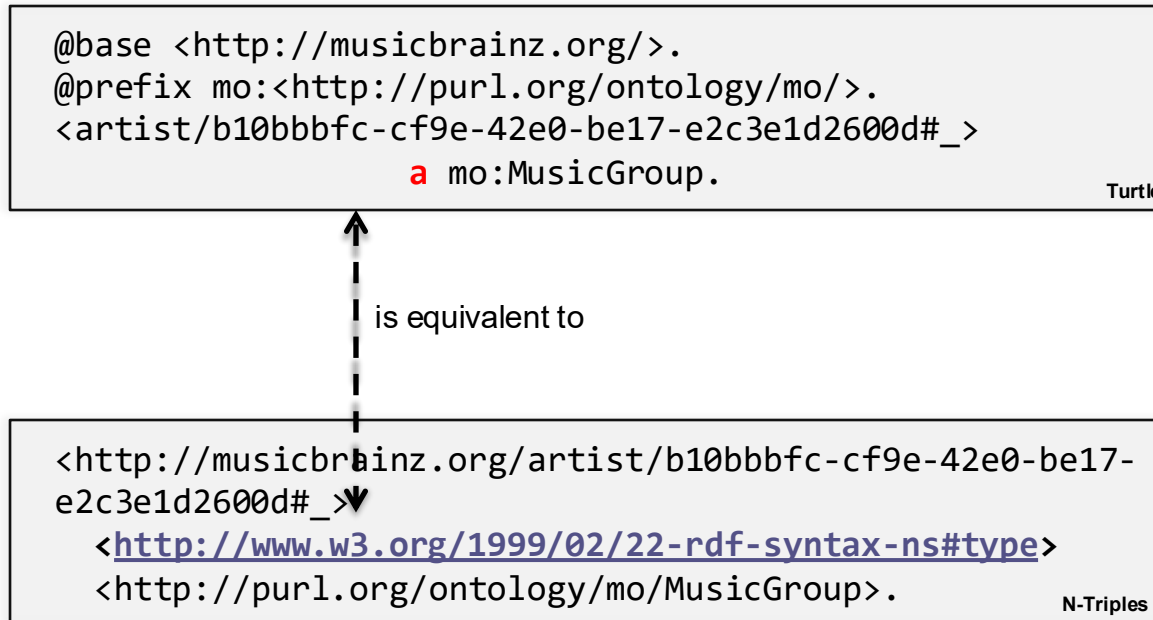
- Unique base:

```
@base <http://musicbrainz.org/>.
```

Turtle

e

Has a simple shorthand for class membership (**a**):



often abbreviated as `rdf:type`

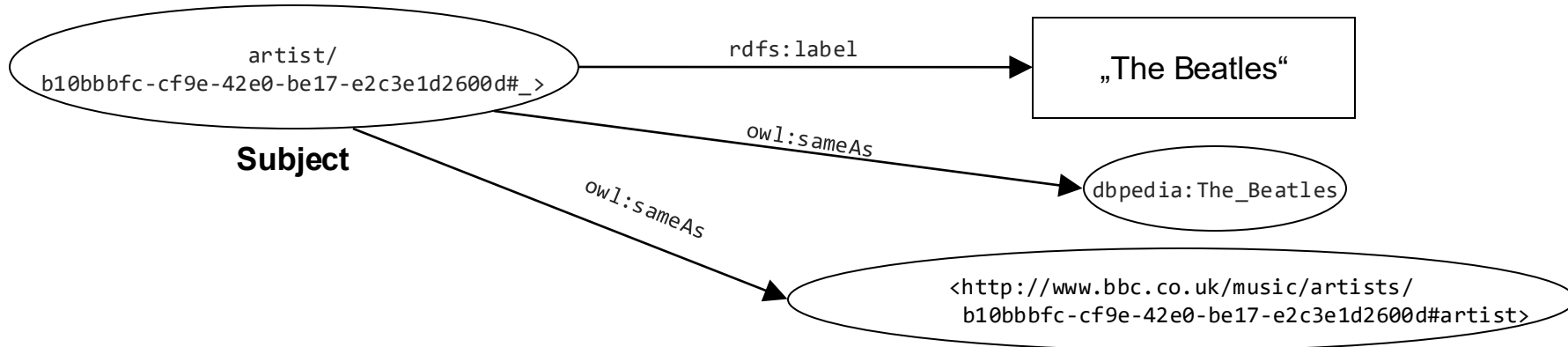
When multiple statements apply to **same subject (and predicate)** they can be abbreviated:

```
<artist/b10bbbfccf9e-42e0-be17-e2c3e1d2600d#_>
  rdfs:label "The Beatles" ;
  owl:sameAs dbpedia:The_Beatles ,
    <http://www.bbc.co.uk/music/artists/
    b10bbbfccf9e-42e0-be17-e2c3e1d2600d#artist> .
```

Turtle

Same subject

Same subject and predicate



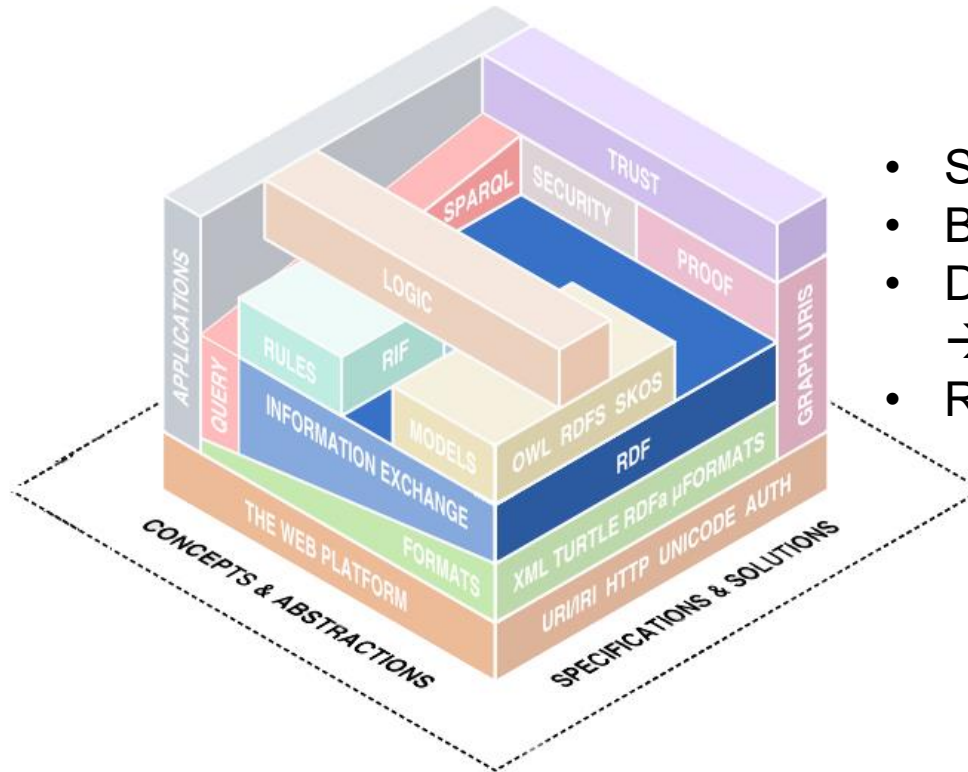
Turtle also provides a simple syntax for data types and language tags for literals:

```
<recording/5098d0a8-d3c3-424e-9367-1f2610724410#_>
  a mo:Signal;
  rdfs:label "All You Need Is Love" ;
  mo:duration "PT3M48S"^^xsd:duration .

dbpedia:The_Beatles dbpedia-owl:abstract
  "The Beatles were an English rock band formed (...) "@en,
  "The Beatles waren eine britische Rockband in den(...) "@de .
```

Turtle

Standard information exchange is key



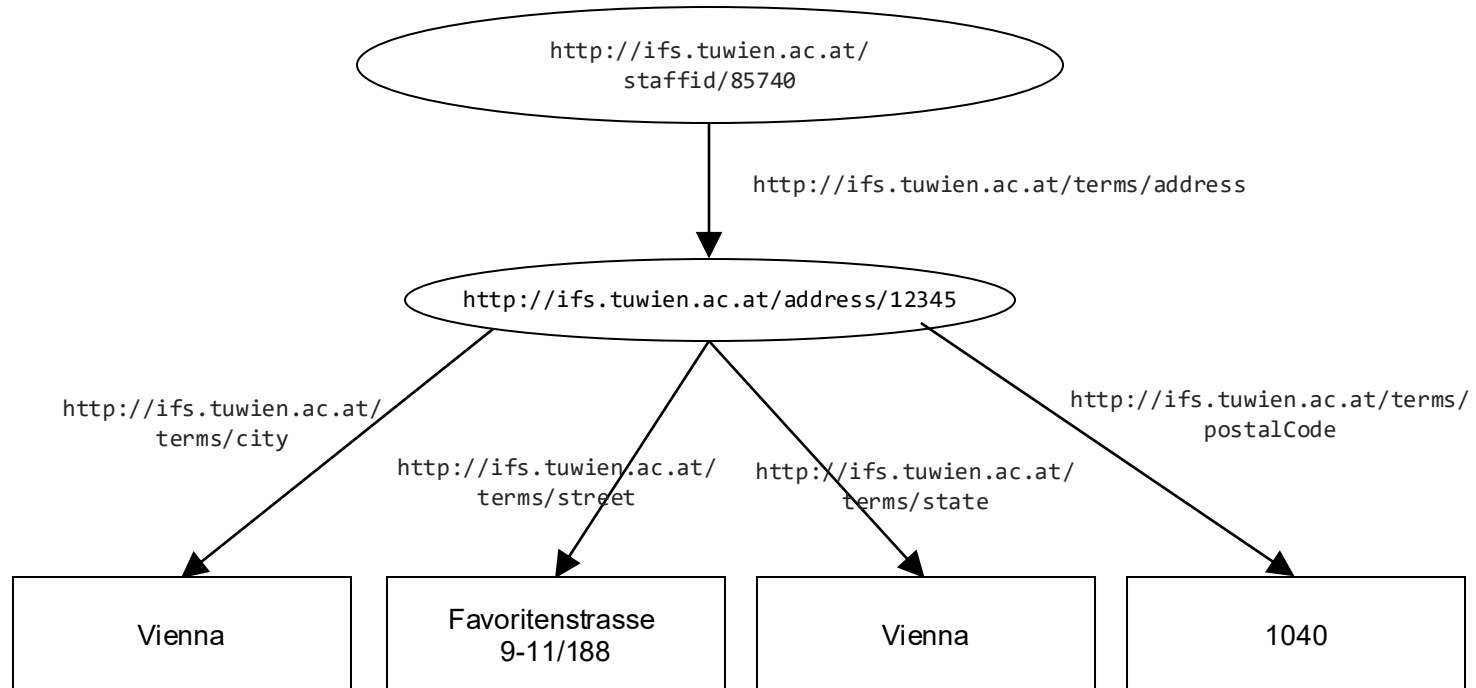
- Structured values
- Blank nodes
- Data structures (container, collection)  
→ see appendix
- Reification

Consider the triple:

```
exstaff:85740
    exterms:address
        "Favoritenstrasse 9-11/188, 1040 Vienna" .
```

How do we represent address as a **structure** consisting of separate *street*, *city*, *state*, and *postal code* values? (N-ary relation)

→ Consider the **"aggregated thing" to be described** (like an address) **as a resource**, and then make statements about that new resource

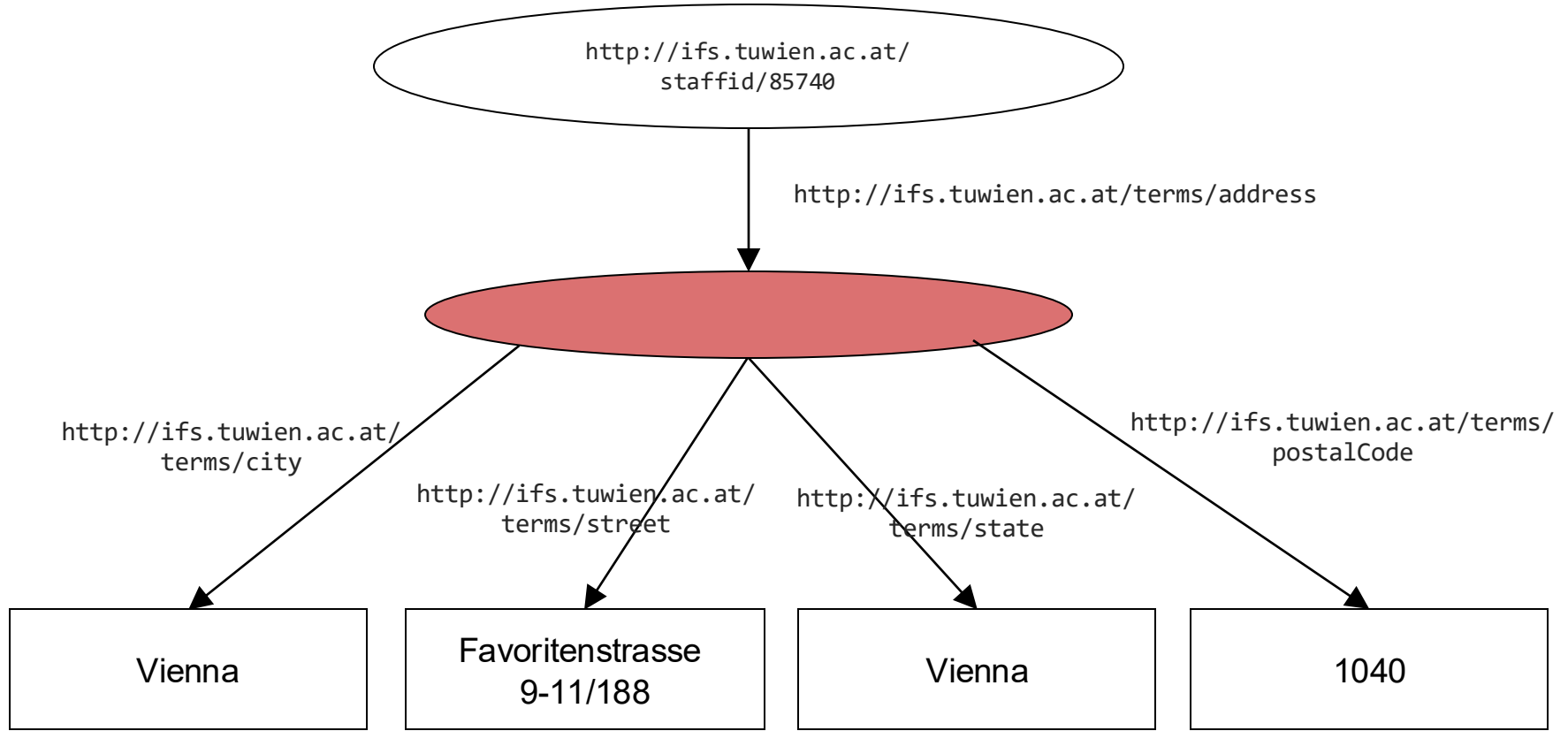


Or in triples notation:

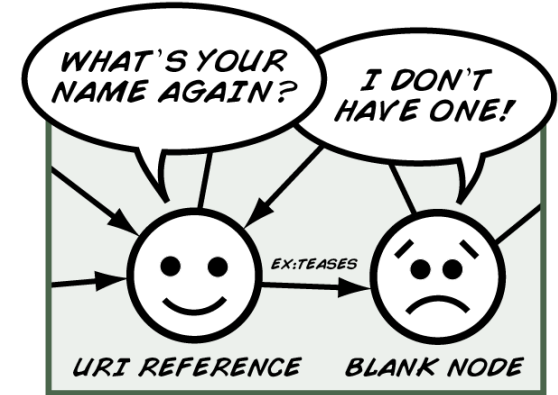
```
exstaff:85740 exterms:address exaddressid:85740 .  
exaddressid:85740 exterms:street "Favoritenstrasse 9-11/188" .  
exaddressid:85740 exterms:city "Vienna" .  
exaddressid:85740 exterms:state "Vienna" .  
exaddressid:85740 exterms:postalCode "1040" .
```

```
exstaff:85740 exterms:address exaddressid:85740 .  
exaddressid:85740 exterms:street "Favoritenstrasse 9-11/188" ;  
                    exterms:city "Vienna" ;  
                    exterms:state "Vienna" ;  
                    exterms:postalCode "1040" .
```

Turtle



- Represent unnamed resources
- Used widely, with varying intentions:
  - Make statements about resources that do not have URIs (but that are described in terms of relationships with other resources that do)
  - group related information
  - represent n-ary relationships



<http://milicicvuk.com/blog/?p=4>

## Issues:

- "Breaks" reasoning, because simple entailment in the presence of existentials is NP-complete [but: often not a big deal in practice\*]
- Use in Linked Data discouraged [but: common in practice\*]

\* A. Hogan, M. Arenas, A. Mallea, and A. Polleres, "Everything you always wanted to know about blank nodes," *Web Semantics: Science, Services and Agents on the World Wide Web*, vol. 27, pp. 42–69, 2014.

```
exstaff:85740 exterms:address ? .  
? exterms:street "1501 Grant Avenue" .  
? exterms:city "Bedford" .  
? exterms:state "Massachusetts" .  
? exterms:postalCode "01730" .
```

Turtle

Is this a valid notation for a blank node?

```
exstaff:85740  exterms:address _:johnaddress .
_:johnaddress  exterms:street "1501 Grant Avenue" .
_:johnaddress  exterms:city  "Bedford" .
_:johnaddress  exterms:state  "Massachusetts" .
_:johnaddress  exterms:postalCode "01730" .Turtle
```

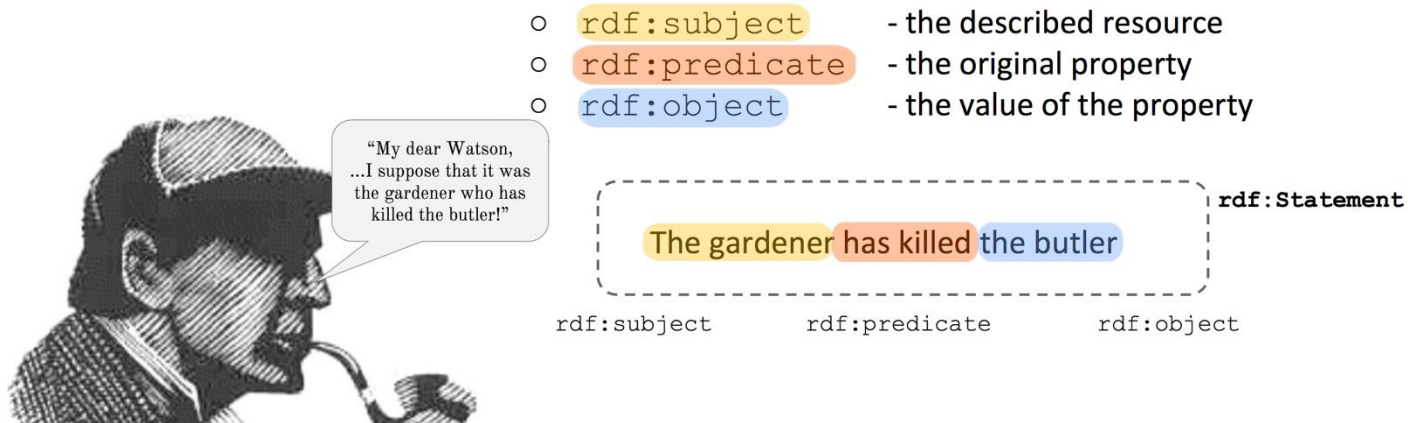
- Each distinct blank node must be given a different blank node identifier
- Blank node identifiers have significance only within a single graph
- Blank node identifiers may only appear as subjects or objects in triples (not predicates)

Used for making statements about statements to

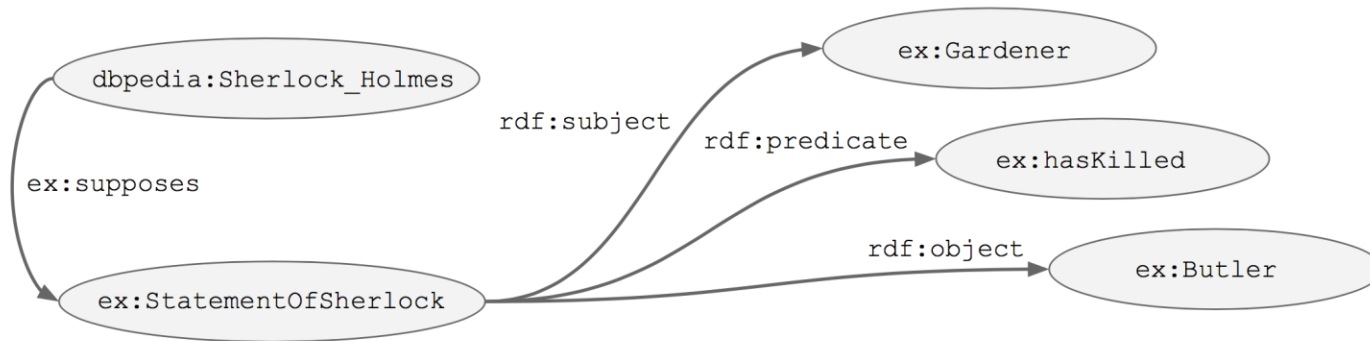
- model data provenance
- formalize statements about reliability and trust
- define metadata about statements

Caveat:

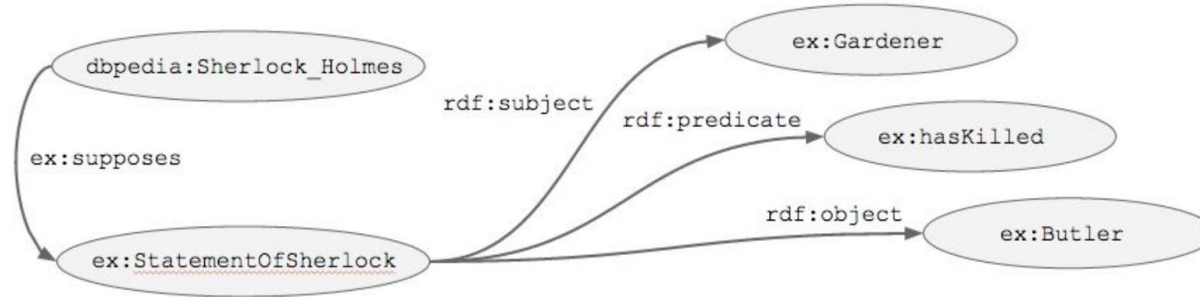
- definition of infinite recursions and cycles



Sherlock Holmes supposes that the gardener has killed the butler



```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix dbpedia <http://dbpedia.org/resource/> .
@prefix ex: <http://example.org/Crimestories#> .
```



```

@prefix dbpedia: <http://dbpedia.org/resource/> .
@prefix rdf:     <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix ex:      <http://example.org/Crimestories#> .

dbpedia:SherlockHolmes ex:supposes ex:StatementOfSherlock .
ex:StatementOfSherlock a rdf:Statement ;
    rdf:subject      ex:Gardener ;
    rdf:predicate   ex:hasKilled ;
    rdf:object      ex:Butler .
  
```

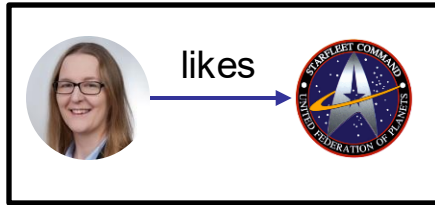
Turtle

# Menti 11

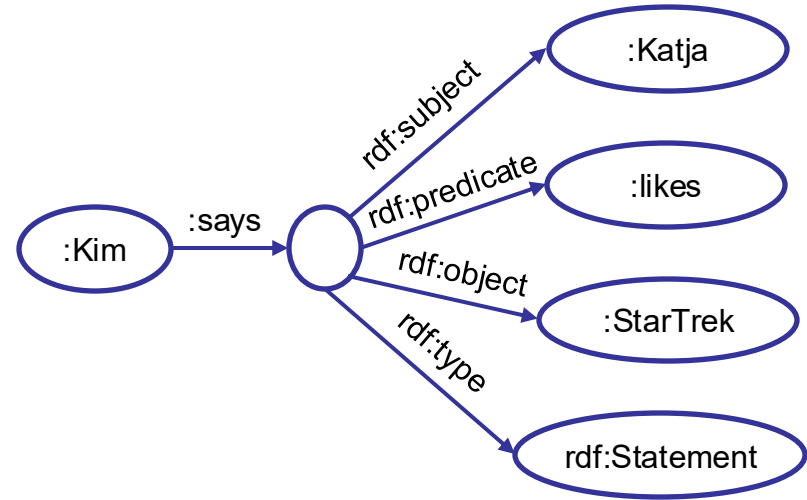


### Standard reification

Kim says



:Katja :likes :StarTrek



```

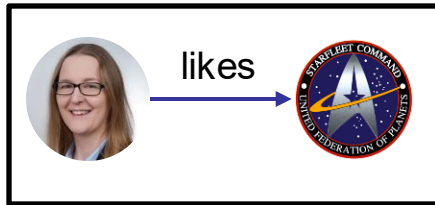
:Kim :says _:s1
_:s1  rdf:type    rdf:Statement
_:s1  rdf:subject :Katja
_:s1  rdf:predicate :likes
_:s1  rdf:object  :StarTrek
  
```

Pros: it works on any basic RDF/SPARQL engine

Cons: number of triples and query triple patterns is at least quadrupled

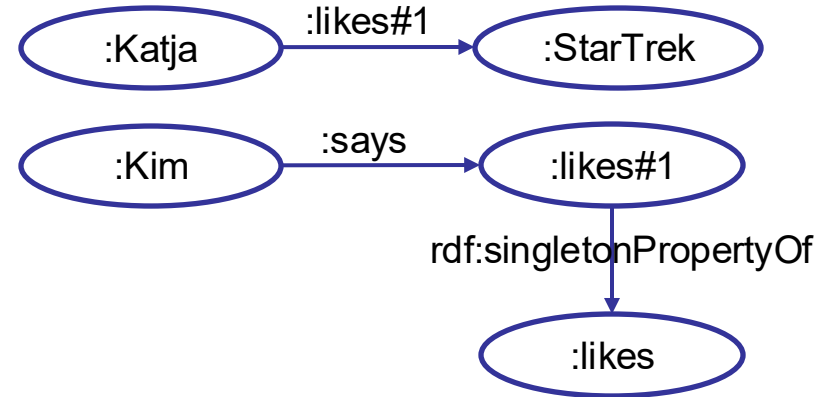
## Singleton property

Kim says



:Katja :likes :StarTrek

## Reification Alternatives



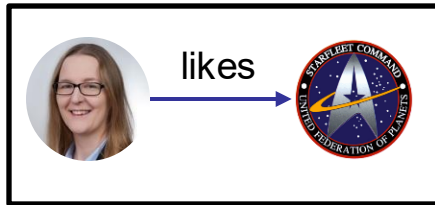
```
:Katja  :likes#1  :StarTrek
:Kim    :says     :likes#1
:likes#1 rdf:singletonPropertyOf :likes
```

Pros: space efficient

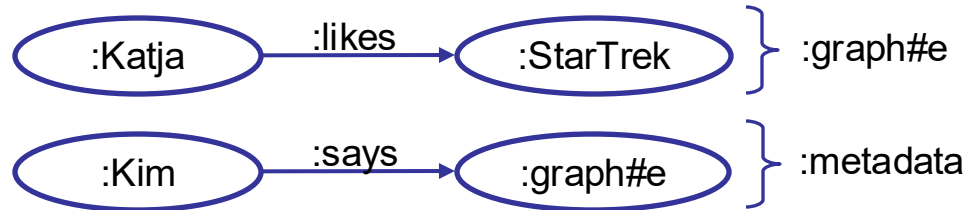
Cons: it requires additional text indexes for predicates

### Named graphs

Kim says



:Katja :likes :StarTrek



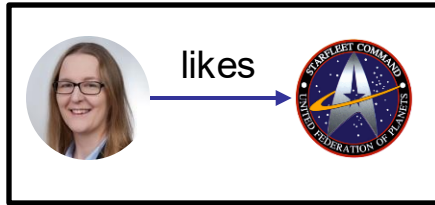
:Katja	:likes	:StarTrek	:graph#e
:Kim	:says	:graph#e	:metadata

Pros: relatively space efficient

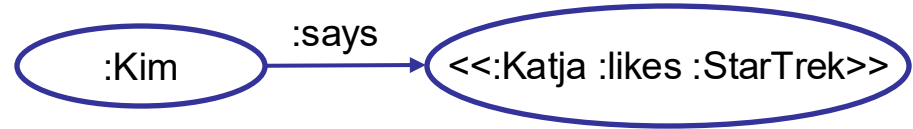
Cons: it requires efficient graph indexing – not always supported

### RDF-star (RDF\*)

Kim says



:Katja :likes :StarTrek



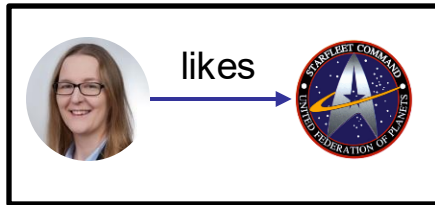
:Kim :says <<:Katja :likes :StarTrek>>

Pros: nested triples treated as first-class citizens

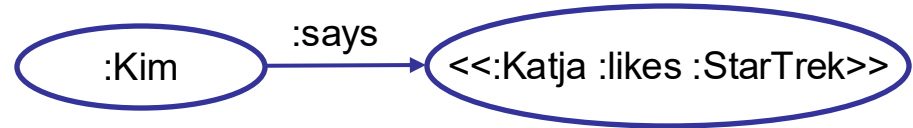
Cons: not supported by all systems

## RDF-star (RDF\*)

Kim says



:Katja :likes :StarTrek



:Kim :says <<:Katja :likes :StarTrek>>

W3C recommendation – not yet a standard

Currently not supported by all triple stores.

Supported (to a certain degree) by GraphDB, Stardog, Jena, Oxigraph

Details:

G. Abuoda, C. Aebeloe, D. Dell'Aglio, A. Keen, K. Hose:

StarBench: Benchmarking RDF-star Triplestores

In: QuWeDa icw. ISWC 2023

## Classes

- **rdf:XMLLiteral** - the class of XML literal values
- **rdf:Property** - the class of properties
- **rdf:Statement** - the class of RDF statements
- **rdf:Alt**, **rdf:Bag**, **rdf:Seq** - containers of alternatives, unordered containers, and ordered containers\*
- **rdf:List** - the class of RDF Lists
- **rdf:nil** - an instance of **rdf:List** representing the empty list

## Properties

- **rdf:type** - an instance of **rdf:Property** used to state that a resource is an instance of a class
- **rdf:first** - the first item in the subject RDF list
- **rdf:rest** - the rest of the subject RDF list after **rdf:first**
- **rdf:value** - idiomatic property used for structured values
- **rdf:subject** - the subject of the subject RDF statement
- **rdf:predicate** - the predicate of the subject RDF statement
- **rdf:object** - the object of the subject RDF statement

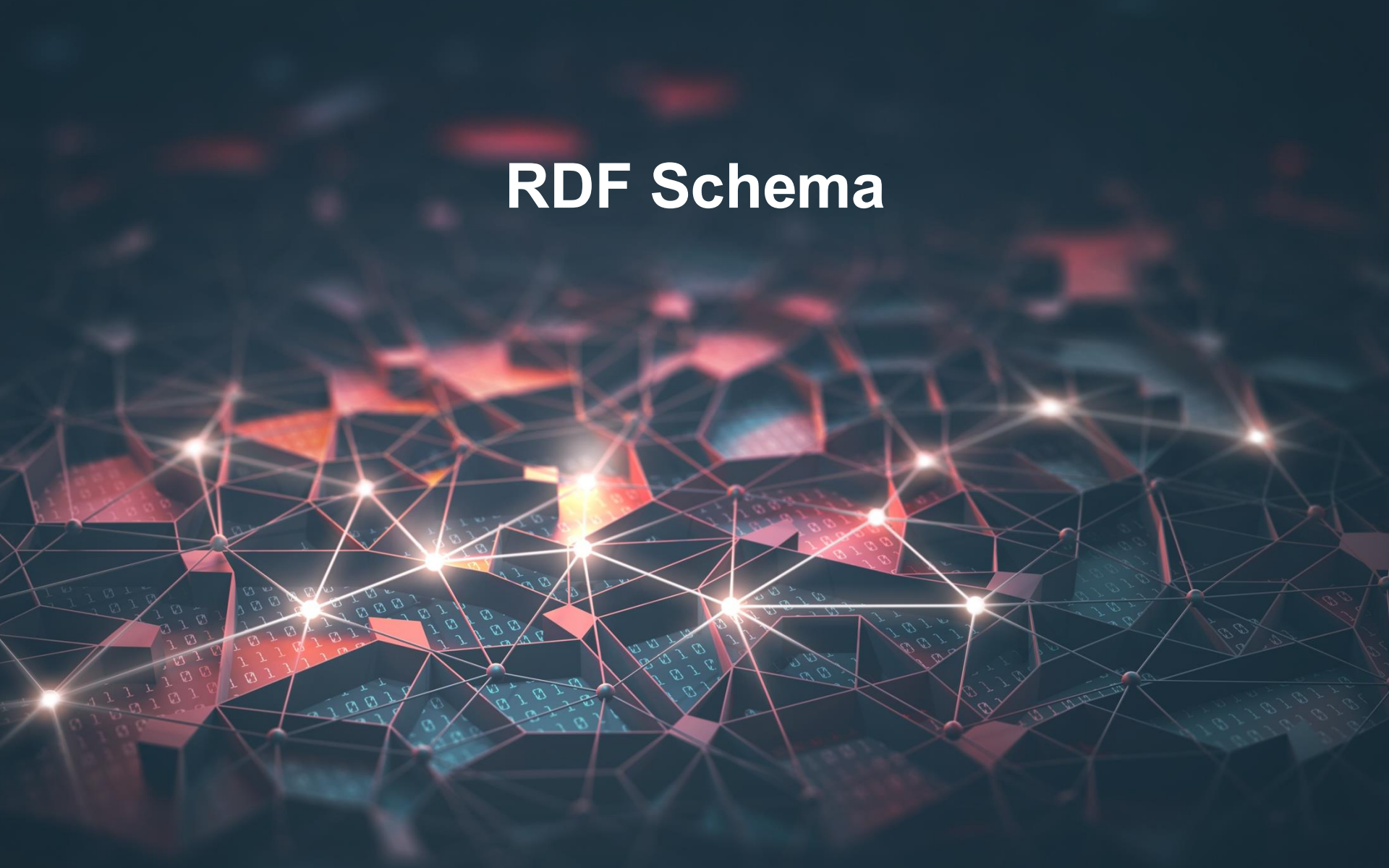
\* **rdfs:Container** is a super-class of the three

- RDF 1.2

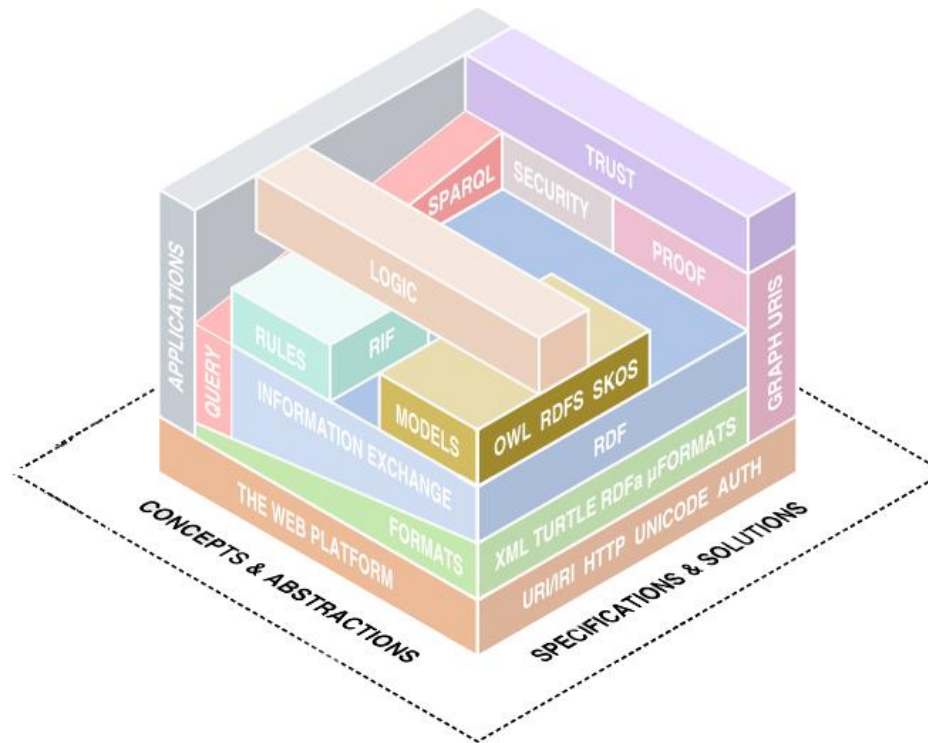
<https://www.w3.org/TR/rdf12-primer/>

- Introduces Quoted Triples (RDF-star)
- More expressive data modeling
- *Under development*

# RDF Schema



Specifying conceptual models:  
specifying **lightweight** ontologies with RDFS.



- On the Web “Anyone can say Anything about Any topic” – **AAA Slogan**
- **Non-unique Naming Assumption**
  - The same entity could be known by more than one name
  - E.g., PersonA can be the same instance as PersonB
  - Disjointness needs to be specified explicitly
- **Open World Assumption**
  - “Missing information is not evaluated as negative information!”
  - Assumes incomplete information by default, in this way, it allows for reasoning on incomplete models.
  - If fact  $f$  in a knowledge base is missing, not  $f$  cannot be inferred
  - E.g., likes(PersonA, DrinkB)
    - CWA: likes(PersonA, DrinkC) – No
    - OWA: likes(PersonA, DrinkC) – don’t know; PersonA may also like other drinks...

- First version in April 1998, W3C recommendation in Feb. 2004
- Every RDF Schema document is an RDF document

## **Purpose:**

### **1. Nominate**

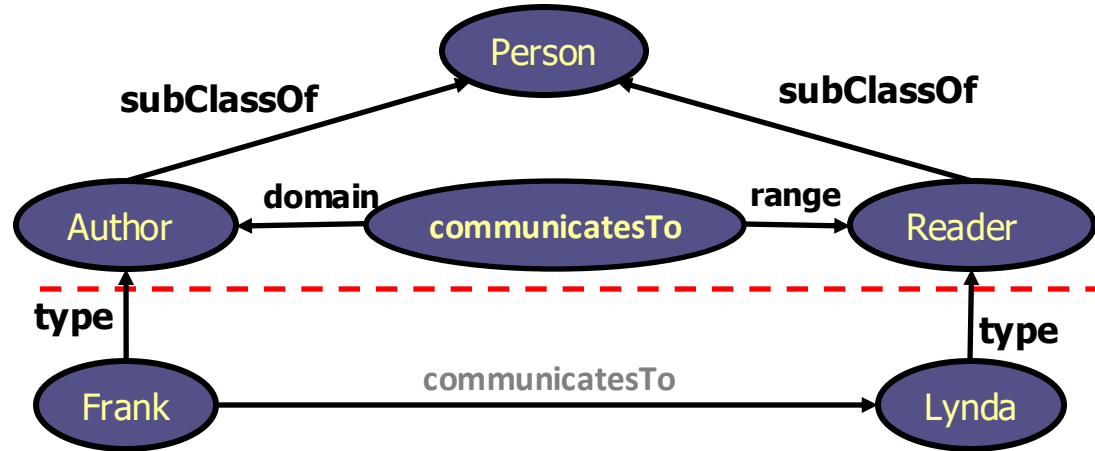
- the ‘types’ (i.e., classes) of things we might make assertions about, and
- the properties we might apply, as predicates in these assertions, to capture their relationships

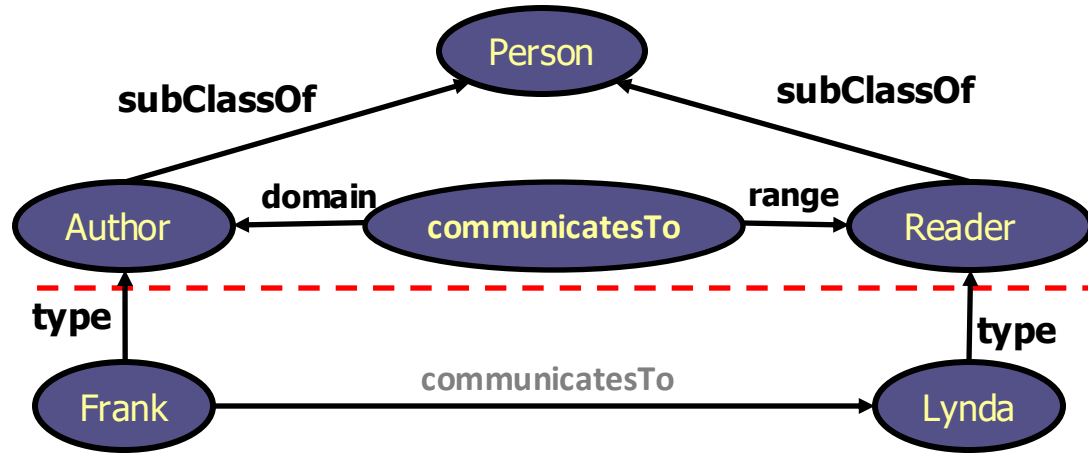
### **2. Inference**

Given a set of assertions, using these classes and properties, specify what should be inferred about assertions that are implicitly made

# What does RDF Schema add?

- Defines the **vocabulary** for RDF
- Organizes this vocabulary in a **typed hierarchy**
  - class, subClassOf, type
  - property, subPropertyOf
  - domain, range





```
<Person, rdf:type, rdfs:Class>
<Reader, rdfs:subClassOf, Person>
<Author, rdfs:subClassOf, Person>
<communicatesTo, rdf:type, rdfs:Property>
<communicatesTo, rdfs:domain, Author>
<communicatesTo, rdfs:range, Reader>
<Frank, communicatesTo, Lynda>
```



```
<Frank, rdf:type, Author>
<Lynda, rdf:type, Reader>
```

## **rdfs:domain**

- Restricts the scope of a property

```
mo:member rdfs:domain mo:MusicGroup .
```

Turtle

## **rdfs:range**

- Used to restrict the type of values of a property

```
mo:member rdfs:range foaf:Agent .
```

Turtle

## RDF and RDFS Vocabularies

### RDF Classes:

`rdf:Property`, `rdf:Statement`,  
`rdf:XMLLiteral`  
`rdf:Seq`, `rdf:Bag`, `rdf:Alt`,  
`rdf>List`

### RDF Properties:

`rdf:type`, `rdf:subject`,  
`rdf:predicate`, `rdf:object`,  
`rdf:first`, `rdf:rest`, `rdf:_n`  
`rdf:value`

### RDF Resources:

`rdf:nil`

### RDFS Classes

`rdfs:Resource`  
`rdfs:Class`  
`rdfs:Literal`  
`rdfs:Datatype`  
`rdfs:Container`  
`rdfs:ContainerMembershipProperty`

### RDFS Properties

`rdfs:domain`  
`rdfs:range`  
`rdfs:subPropertyOf`  
`rdfs:subClassOf`  
`rdfs:member`  
`rdfs:seeAlso`  
`rdfs:isDefinedBy`  
`rdfs:comment`  
`rdfs:label`

## Classes:

- **rdfs:Resource** all resources within RDF are implicitly members of this class
- **rdfs:Class** declares a resource as a class (type or category) for other resources; instantiation of classes with `rdf:type`
- **rdfs:Literal** literal values such as strings and integers (plain or typed)
- **rdfs:Datatype** class of datatypes (both an instance of and a subclass of `rdfs:Class`; each instance of `rdfs:Datatype` is a subclass of `rdfs:Literal`)

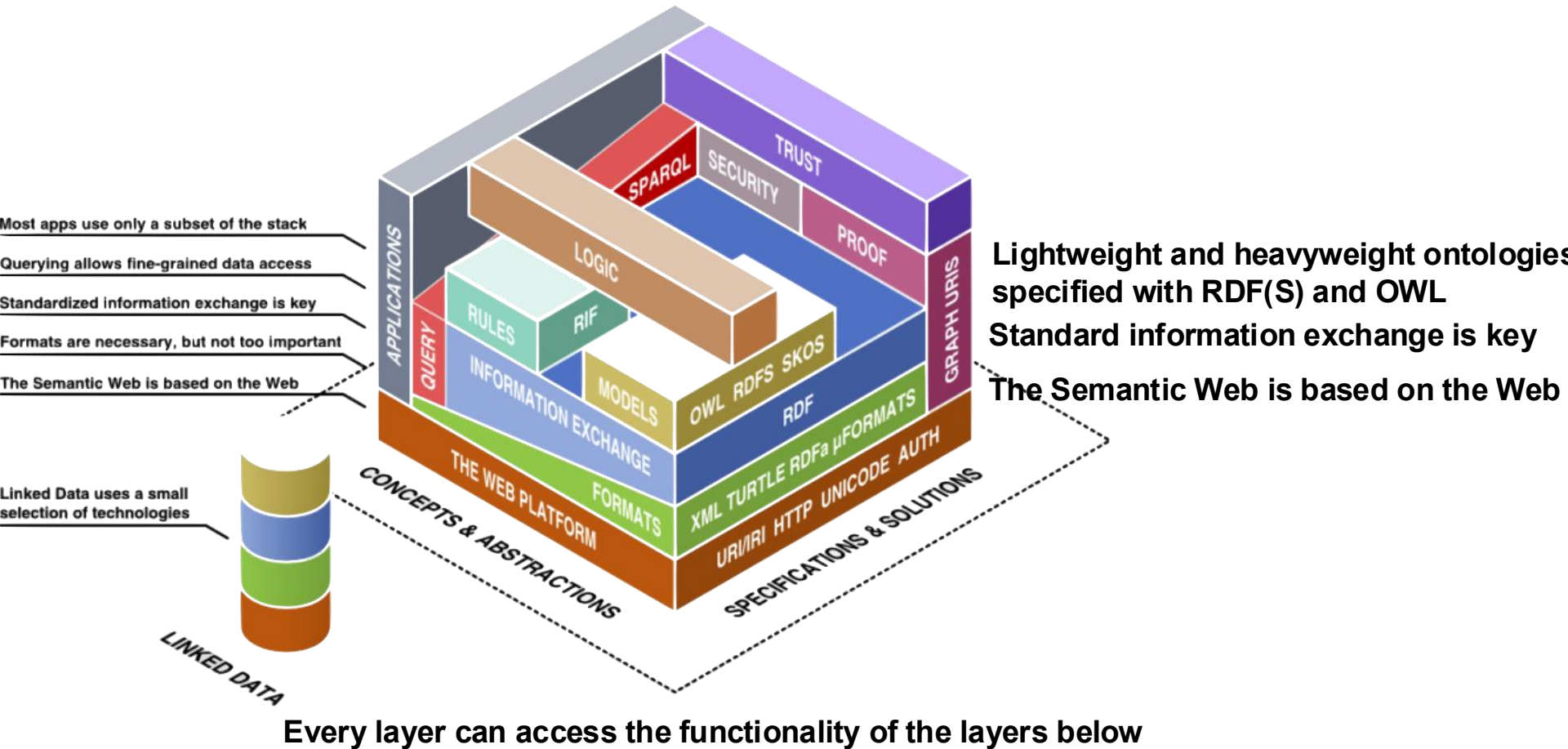
## Properties:

- **rdfs:subClassOf** allows to declare hierarchies of classes; properties of super class are inherited
- **rdfs:subPropertyOf** all resources related by one property are also related by another
- **rdfs:domain** defines the scope of a property (see slide)
- **rdfs:range** defines the range of values of a property (see slide)
- **rdfs:comment** used to provide a human-readable description of a resource
- **rdfs:seeAlso** used to indicate a resource that might provide additional information
- **rdfs:isDefinedBy** indicates a resource defining the subject source

(no inference)

- Allows basic inference
- Poor expressivity
- Does not allow complex expressions besides membership

## Summary: The Semantic Web Technology Stack



- RDF and Linked Data
- RDFS
- Outlook: OWL (in a later lecture)

Some advanced topics that might be helpful for your projects are mentioned in the appendix.

# Appendix 1

## RDF



```
exstaff:85740  exterms:address _:johnaddress .
_:johnaddress  exterms:street "1501 Grant Avenue" .
_:johnaddress  exterms:city  "Bedford" .
_:johnaddress  exterms:state  "Massachusetts" .
_:johnaddress  exterms:postalCode "01730" .Turtle
```

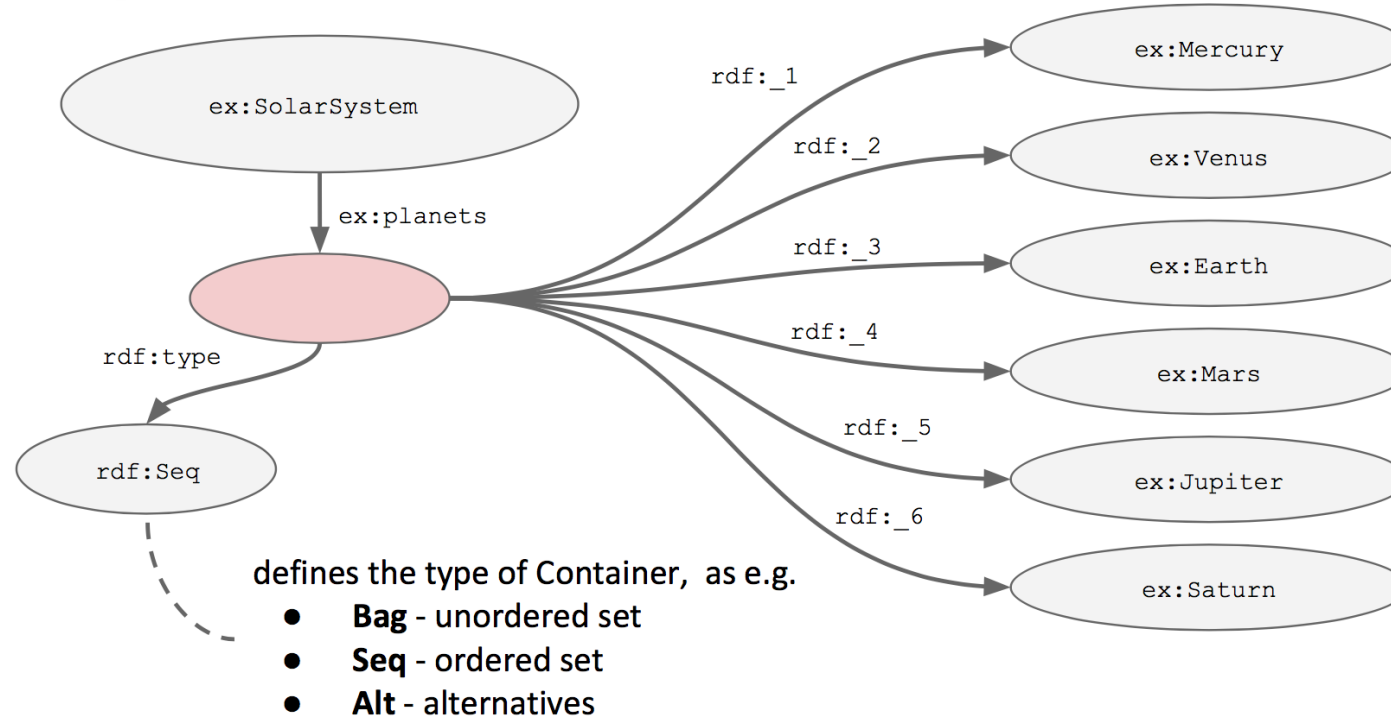
- Each distinct blank node must be given a different blank node identifier
- Blank node identifiers have significance only within a single graph
- Blank node identifiers may only appear as subjects or objects in triples (not predicates)

- General Data structure to enumerate resources or literals
- Only shortcuts without additional semantic expressivity (syntactic sugar)

## Distinguish:

- **Container:**  
open list, i.e. extension (new entries) possible
- **Collections:**  
closed list, i.e. no extension possible

## RDF Container (Open Lists)



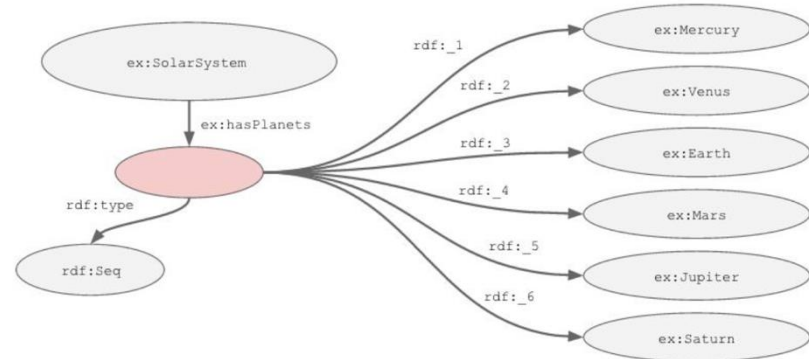
## RDF Container (Open Lists)

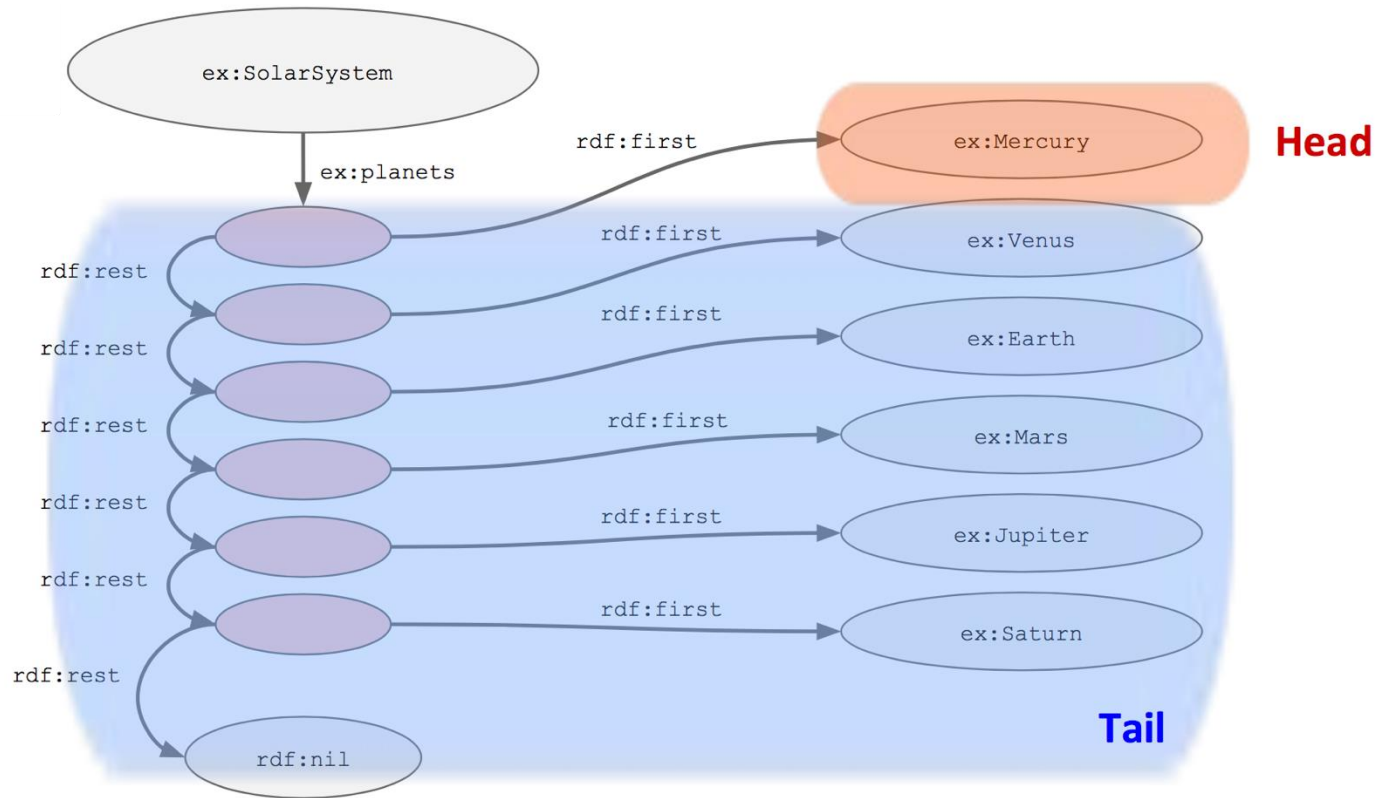
```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix ex: <http://example.org/test#> .
```

```
ex:SolarSystem ex:planets [
  a rdf:Seq ;
  rdf:_1 ex:Mercury ;
  rdf:_2 ex:Venus ;
  rdf:_3 ex:Earth ;
  rdf:_4 ex:Mars ;
  rdf:_5 ex:Jupiter ;
  rdf:_6 ex:Saturn
] .
```

Turtle

= **rdf:type** rdf:Seq

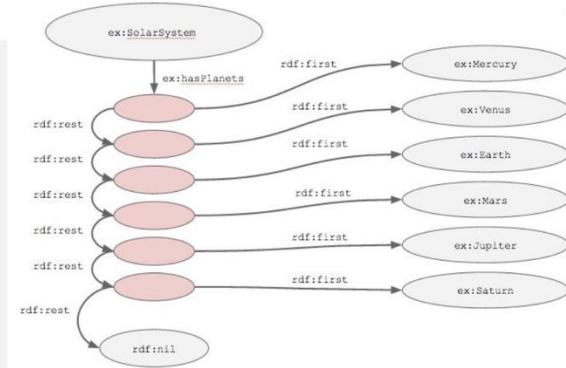




## RDF Collection (Closed Lists)

```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix ex: <http://example.org/test#> .
```

```
ex:SolarSystem ex:planets [
  rdf:first ex:Mercury ; rdf:rest [
    rdf:first ex:Venus ; rdf:rest [
      rdf:first ex:Earth ; rdf:rest [
        rdf:first ex:Mars ; rdf:rest [
          rdf:first ex:Jupiter ; rdf:rest [
            rdf:first ex:Saturn ;
            rdf:rest rdf:nil
          ]
        ]
      ]
    ]
  ]
]
] .
```



```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix ex: <http://example.org/test#> .

ex:SolarSystem ex:planets (
  ex:Mercury ex:Venus ex:Earth ex:Mars ex:Jupiter ex:Saturn
) .
```

**Turtle**

Turtle provides a nice syntax for RDF collections: simply put elements into parentheses