Building and Mining Knowledge graphs

(KEN4256)

Lecture 2: KG Construction from Structured Data

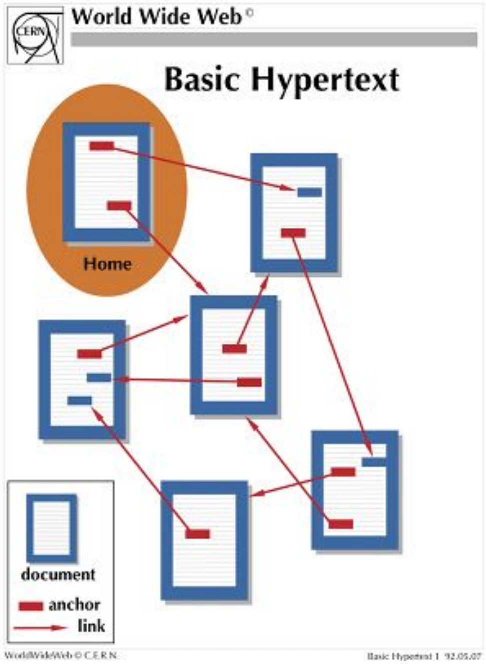
Maastricht University
Institute of Data Science

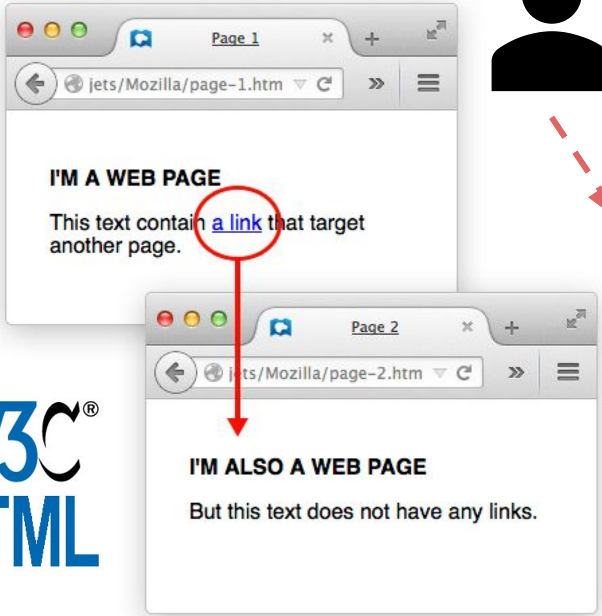
id: KEN4256_L2
version: 1.2024.0
created: February 19, 2020
last modified: March 26, 2024
published on: March 26, 2024

Long history of Knowledge Representation (KR)

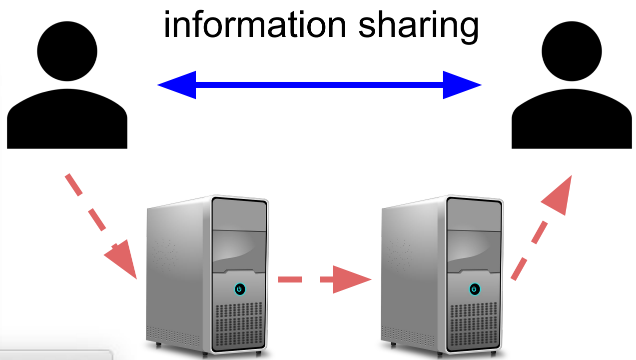
- Existential Graphs (1896; Peirce)
- Semantic Networks “semnets” (Introduced mid-1960s, hype 1980s),
- Conceptual Graphs
- Cognitive Semantic Networks
- Structured Inheritance Networks
- Multilayered Extended Semantic Networks (MultiNets)
- Basic Conceptual Graphs
- Full Conceptual Graphs
- Hierarchical Semantic Form
- **Resource Description Framework (RDF)**
- Property Graph

World Wide Web





W3C® HTML



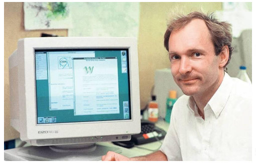
Humans have to make sense of (extract **knowledge** from) the information content

Web 2.0 vs Web 3.0

Semantic Web, the real Web 3.0

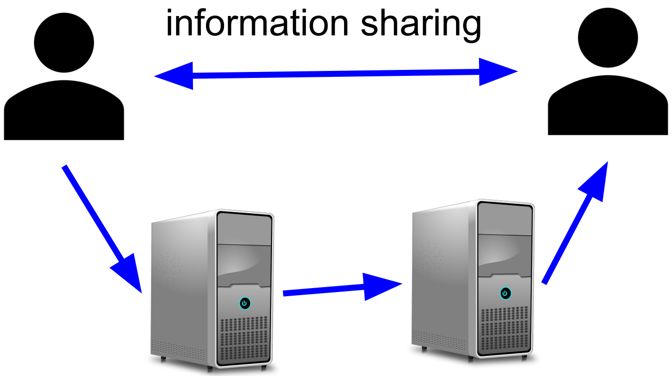
The Web 2.0 has been built to be Human readable, not machine readable.

Tim Berners-Lee, 2001



"The Semantic Web is not a separate Web but an extension of the current one, in which information is given well-defined meaning, better enabling **computers and people** to work in cooperation."

Scientific American, May 2001



Humans & computers alike can interpret the information and share a common understanding of its meaning

World Wide Web Consortium (W3C)

Working Group for developing and maintaining Web Standards

- Graphical interface standards (HTML, CSS): to make websites beautiful to human eyes.
- New data formats (XML, N-Triples, N-Quads, Turtle, TriG, JSON-LD): To store and exchange data in a well defined manner.
- Formal languages (RDF, RDFS, OWL) with mathematically defined symbols to represent and reason about knowledge
- New query languages (SPARQL, R2RML, RML) to query and transform data
- Ontologies and vocabularies (SKOS, FOAF, DCAT, PROV) to structure knowledge in a consistent manner.
- Reasoning algorithms to automatically infer implicit facts about data with background knowledge

All of these new standards should be compatible with / build on top of existing Web standards

Resource Description Framework (RDF)

RDF is a framework to represent information on the Web

RDF was developed by the W3C and first became a W3C recommendation in 1999. This was superseded by the RDF 1.0 specification in 2004, and the RDF 1.1 specification in 2014. The full **RDF 1.1 specification** consists of:

- An abstract data model with a set of implementing syntaxes (aka. formats) for storing and exchanging information represented in RDF
- a formal semantics that can be extended to create vocabularies and ontologies
- A first vocabulary RDF Schema which extends RDF with basic concepts and a hierarchical class system (label, subclass, subproperty…).
- A query language for RDF data - SPARQL

RDF is highly versatile

RDF can be used to describe things in a machine readable manner. This can include **digital objects** such as documents, **real world objects** such as people, and **concepts** such as Person or Organization.

RDF is a graph formalism in that the intention is to describe information about interconnected entities that can be depicted in a directed graph.

Not only used for KGs! RSS feed comes from RDF Site Summary

RDF spec (and a word on W3C specs in general)

w3.org/TR/rdf11-concepts/

W3C Recommendation

RDF 1.1 Concepts and Abstract Syntax

W3C Recommendation 25 February 2014

This version:
http://www.w3.org/TR/2014/REC-rdf11-concepts-20140225/
Latest published version:
http://www.w3.org/TR/rdf11-concepts/
Previous version:
http://www.w3.org/TR/2014/PR-rdf11-concepts-20140109/
Previous Recommendation:
http://www.w3.org/TR/rdf-concepts
Editors:
Richard Cyganiak, DERI, NUI Galway
David Wood, 3 Round Stones
Markus Lanthaler, Graz University of Technology
Previous Editors:
Graham Klyne
Jeremy J. Carroll
Brian McBride

Title, date created, version info etc.

Please check the **errata** for any errors or issues reported since publication.

The English version of this specification is the only normative version. Non-normative translations may also be available.

Abstract

Short description of the purpose of the document

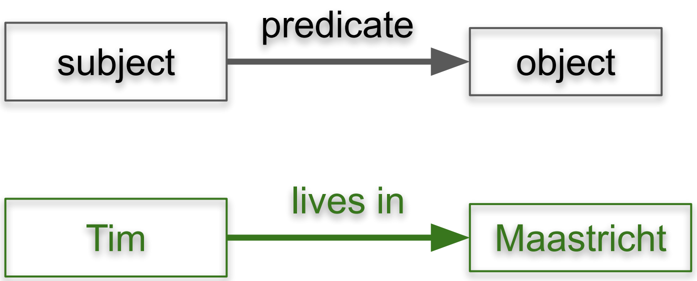
The Resource Description Framework (RDF) is a framework for representing information in the Web. This document defines an abstract syntax (a data model) which serves to link all RDF-based languages and specifications. The abstract syntax has two key data structures: RDF graphs are sets of subject-predicate-object triples, where the elements may be IRIs, blank nodes, or datatyped literals. They are used to express descriptions of resources. RDF datasets are used to organize collections of RDF graphs, and comprise a default graph and zero or more named graphs. RDF 1.1 Concepts and Abstract Syntax also introduces key concepts and terminology, and discusses datatyping and the handling of fragment identifiers in IRIs within RDF graphs.

W3C Recommendation

Table of Contents

The RDF Data Model

The fundamental unit of an RDF graph is a statement called a **triple**. An RDF graph G = (S, P, O) is a set of triples comprised of a **subject**, **predicate**, and **object** tuple. The subject and object represent *nodes* in the RDF graph and the predicate represents an *edge* between these nodes.



The subject and predicate must be an Internationalized Resource Identifier (IRI), while the object can be an IRI, blank node, or unicode string literal.

IRI or URI as identifiers for things

Uniform Resource Identifier: “string of characters that unambiguously (uniquely) identifies a particular resource…”

“... a **name** given to describe a thing. This thing can be anything in the world (**either physical or abstract**).”

Things (also called resources or entities) can have **multiple** URIs

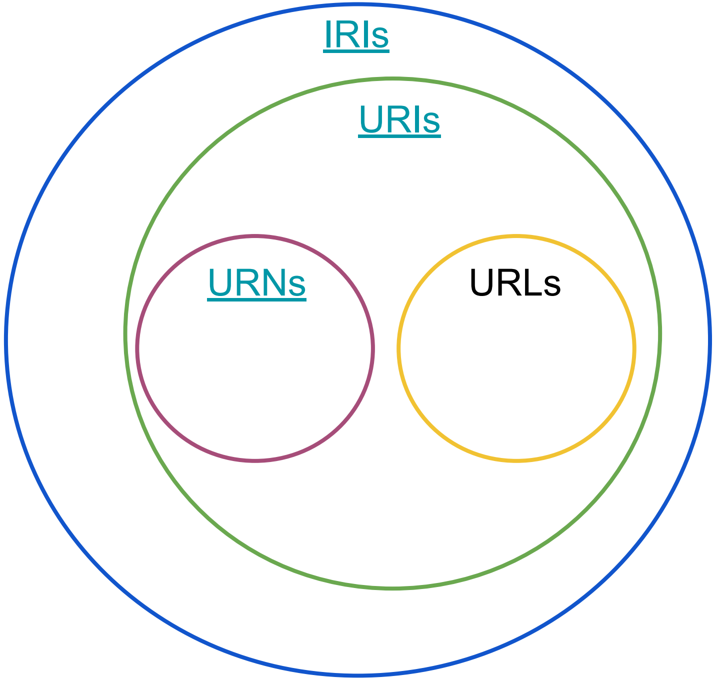
http://dbpedia.org/page/Tim_Berners-Lee

Refers to

https://www.wikidata.org/wiki/Q80

Refers to

IRIs, URIs, URLs, URNs



Identifiers in this family:

- **Internationalized Resource Identifier** (**IRI**): An IRI is a sequence of characters from the Universal Character Set (Unicode/ISO10646)
- **Uniform Resource Identifier (URI)**: Uses US-ASCII, a subset of ASCII, about 60 characters
- **Uniform Resource Locator** (**URL**): A compact string representation of the location for a resource that is available via the Internet.
- **Uniform Resource Name** (**URN**) Uniform Resource Name - a globally unique and persistent URI (even after a resource ceases to exist this identifier will remain). Protocol of URN is always “urn:” - **rare in practice, used for examples**
- Further reading on differences: link
- Using Linked Data best practices: URI = URL
- URI best practices reading: CoolURIs and here

Composition of an IRI

Namespace

https://springer.com/gp/book/9783030123741

scheme / protocol

domain name

resource name

This URI has a “Slash” namespace (last character preceding the resource name is a “/” character).

Can also have “Hash” namespaces in URIs e.g. **http://www.w3.org/1999/02/22-rdf-syntax-ns#type**

Prefixed name

A prefixed name is a prefix label and a local part, separated by a colon ":". A prefixed name is turned into an IRI by concatenating the IRI associated with the prefix and the local part.

springer:9883030123741

is equivalent to: https://springer.com/gp/book/9783030123741

when the prefix is declared in a RDF document,

PREFIX **springer**: <https://springer.com/gp/book/>

What is the namespace of the following resource: http://publications.europa.eu/ontology/cdm#ATTO_FD_010 ?

Click **Present with Slido** or install our Chrome extension to activate this poll while presenting.

RDF Subjects, Predicates, Objects

Starry night (June 30, 1889) by Vincent van Gogh

Subjects and **Predicates** (aka. properties) are denoted by [IRIs]()

Objects are either:

- an **IRI** (we talk about object property)
- or a literal value (data property)

Object property

Subject	Predicate	Object - IRI
<http://somenamespace.com/**starrynight**>	<http://somenamespace.com/**createdBy**>	<http://somenamespace.com/**Vincent_van_Gogh**> .

		Object - Literal
<http://somenamespace.com/**starrynight**>	<http://somenamespace.com/**dateCreated**>	“1889-06-30”^^xsd:date .

Data property

N-Triple format: IRIs are encapsulated with < >, literals are encapsulated “ ”, and ends with period .

RDF Literals

A **literal** can be accompanied by additional, and optional, details:

1) a **datatype IRI**, that determines how the lexical form maps to a literal value.
2) a **language tag only** when the datatype IRI is a rdf:langString

Many RDF syntaxes use **^^** (double caret) to associate the datatype of a literal with is lexical form, and the **@** (at sign) for specifying the language tag:

- “1.01”^^<https://www.w3.org/2001/XMLSchema#decimal>
- “1.01”^^xsd:decimal
- “Maastricht University”@en
- “Universiteit Maastricht”@nl

Data types of literals

The use of **XML Schema** data types is recommended (they are predefined and widely used)

Strings:	"Maastricht University"
Integers:	"1"^^<http://www.w3.org/2001/XMLSchema#integer>
Integers:	"1"^^xsd:integer
Decimals:	"1.23"^^xsd:decimal
Date:	"2014-9-11"^^xsd:date
Time:	"11:05:45"^^xsd:time
Date with time:	"2014-9-11T11:05:45"^^xsd:dateTime

RDF Blank Nodes

There are three kinds of object nodes in an RDF graph: IRIs, literals, and **blank nodes**.

Blank nodes are nodes that do not have a persistent identifier, and therefore cannot be reliably linked to by others. They are convenient for prototyping or complex data structure where there is no desire to assign a permanent identifier. They are used in some RDF syntaxes and some RDF store implementations.

⚠️ It is not recommended to use blank nodes. Instead assign IRIs. We expect your work to not use blank nodes.

IRIs can be abbreviated using defined prefixes

Full IRI:

`http://www.w3.org/1999/02/22-rdf-syntax-ns#type`

Define an abbreviation for your namespace (called a prefix):

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

Shortened IRI: `rdf:type`

ℹ️ Use https://prefix.cc to find out the URI associated to a prefix

RDF Predicates (built-in)

RDF has some predicates built into its vocabulary that the specification developers decided would be needed and widely reused, independent of the type of information captured.

ℹ️ http://www.w3.org/1999/02/22-rdf-syntax-ns#type aka. `rdf:type` or just “`a`”, is a special predicate defined by RDF which is used to specify a category (called a concept, class or type) to which an entity belongs (the “is a” relation). The category / type entities are usually defined in an ontology / vocabulary (e.g. schema.org). Entities which belong to a certain type are called instances of that type.

<http://somenamespace.com/**starrynight**> <http://www.w3.org/1999/02/22-rdf-syntax-ns#**type**> <http://somenamespace.com/**Painting**> .

Instance “Starry night is an instance of the Painting class” **Class**

🏷️ http://www.w3.org/2000/01/rdf-schema#label aka. `rdfs:label`, is a predicate defined by the RDF Shema vocabulary which can be used to attach a string label or name to an entity in the graph.

<http://somenamespace.com/**starrynight**> <http://www.w3.org/2000/01/rdf-schema#**label**> “The Starry Night”@en .

RDF Schema

RDF Schema is a vocabulary to enhance the meaning of RDF entities with human readable annotations and machine-understandable semantics

RDF Schema defines a vocabulary for:

- labels, descriptions, and pointers to other resources
- classes and class hierarchies
- properties and property hierarchies
- the domains and ranges of properties

RDF Schema Doc: https://www.w3.org/TR/rdf-schema/
RDFS: http://www.w3.org/2000/01/rdf-schema#

RDF(S) Annotation Properties

- Properties that improve human readability
  - **rdfs:label**, to associate a human friendly label (name) with the resource. Range: Literal
  - **rdfs:comment**, to add additional information about the resource. Range: Literal
  - **rdf:value** , to add a structured value for the resource. Range: Literal

- Properties that enables us to refer other RDF definitions
  - **rdfs:seeAlso**, to point to another resource that provides additional information about the resource. Range: IRI
  - **rdf:isDefinedBy**, to specify the resource that is responsible for its definition (e.g. an ontology or vocabulary). Range: IRI

RDFS Classes

RDF Schema contains the following classes:

rdfs:Resource, the class of all resources
rdfs:Class, the class of all classes

rdf:Property, the class of all Properties
rdf:langString, the class of language-tagged string values
rdfs:Literal, the class of all literals (strings)
rdfs:Datatype, the class of datatypes

rdf:Statement, the class of all reified statements (using the predicates rdf:subject, rdf:predicate and rdf:object)

RDFS Inference

RDF Schema offers basic inference with those properties:

rdfs:subClassOf
rdfs:subPropertyOf

rdfs:domain
rdfs:range

RDFS Inference

RDF Schema offers basic inference with those properties:

rdfs:subClassOf
rdfs:subPropertyOf

rdfs:domain
rdfs:range

ex:Bob foaf:knows ex:Alice
foaf:knows rdfs:domain foaf:Person .

RDFS Inference

RDF Schema offers basic inference with those properties:

rdfs:subClassOf
rdfs:subPropertyOf

rdfs:domain
rdfs:range

ex:Bob foaf:knows ex:Alice .
foaf:knows rdfs:domain foaf:Person .
->
ex:Bob rdf:type foaf:Person .

RDFS Inference

From

```
ex:Bob rdf:type ex:Human .
ex:Human rdfs:subClassOf ex:Mammal .
```

🧠 We can infer that Bob is a Mammal!

```
ex:Bob rdf:type ex:Mammal .
```

⚠️ **RDFS and ontologies are not built to constraint data!** Only to infer new statements.

✅ So, we **can infer** that Bob is a Mammal

❌ But you can’t make complicated inferences: An individual is either a bird or a person. all birds have feathers, Bob has no feathers, therefore Bob is not a Bird.

🌍 This kind of knowledge can be inferred with more sophisticated KR languages such as the Web Ontology Language

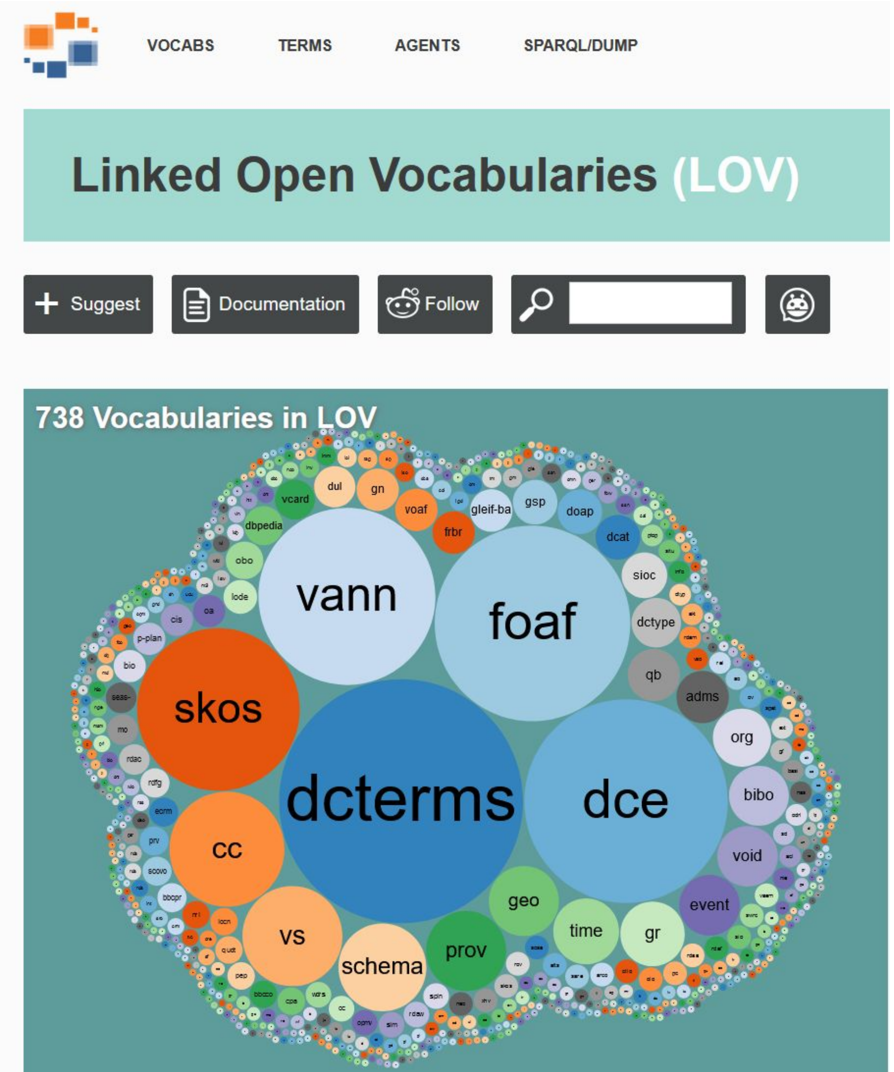
RDF Predicates and Classes (external)

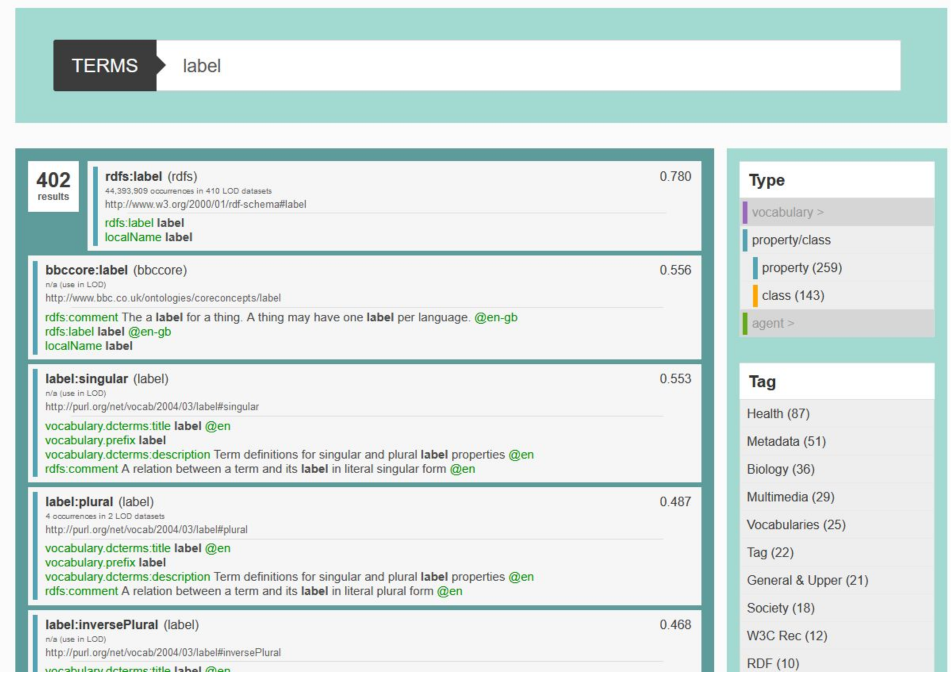
There are other predicates, **and classes**, outside of RDF defined in other **vocabularies** on the Web. There are many of these vocabularies, anyone can define and publish one. Some domain-specific e.g. biomedicine, and some domain-agnostic.

🔍 You can **search** for predicates and classes in the Linked Open Vocabularies service: **https://lov.linkeddata.es/dataset/lov**

E.g. the IRI `https://schema.org/employees` is a predicate defined by the popular Schema.org vocabulary, defined by a collaboration among the world's major search engines.

<http://somenamespace.com/**maastricht_university**> <https://schema.org/**employees**> <http://somenamespace.com/**Xu**> .





https://lov.linkeddata.es/dataset/lov/

From natural language to RDF data model to machine readable statements

natural language

RDF is a W3C standard

abstract syntax
(data model)

w3:RDF —*rdf:type*→ w3:standards

serialization

```
<https://www.w3.org/RDF>
    <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
        <https://www.w3.org/standards> .
```

RDF Syntaxes: N-Triples

N-Triples

```
<https://www.w3.org/RDF> <http://www.w3.org/2000/01/rdf-schema#label>  "Resource Description Framework"@en .
```

Each line is composed of subject, predicate, object
IRIs are encapsulated by angled brackets < >
Literals by quotes “” and optionally with language tags (@lang) or datatype IRIs (^^IRI)
lines terminated by periods .

ℹ️ Easier to stream or split the data in chunks, but less readable and larger files

RDF Syntaxes: Turtle

Turtle

```
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>
@prefix w3: <https://www.w3.org/ >
w3:RDF rdfs:label "Resource Description Framework "@en .
```

Each line composed of subject, predicate, object
Prefixes can be defined at the start, and used in a shorthand notation prefix:suffix that expands to valid IRIs
IRIs are encapsulated by angled brackets < >
Literals by quotes “” and optionally with language tags (@lang) or prefixed datatypes (^^xsd:datatype)
lines terminated by periods .

ℹ️ One of the most popular way to write RDF, compact and easy to read

Turtle: Multiple statements about one resource

- A resource may have **multiple properties** with values
  - Specified by separating the properties with values by semicolons
- A property of a resource may contain **multiple values**
  - Specified by separating the values by commas:

Turtle

```
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>
@prefix schema: <https://schema.org/>
@prefix um: <http://maastrichtuniversity.nl/>
um: a schema:CollegeOrUniversity ;
    rdfs:label "Maastricht University" ;
    schema:department um:FSE, um:FHML .
```

RDF syntaxes: RDF/XML

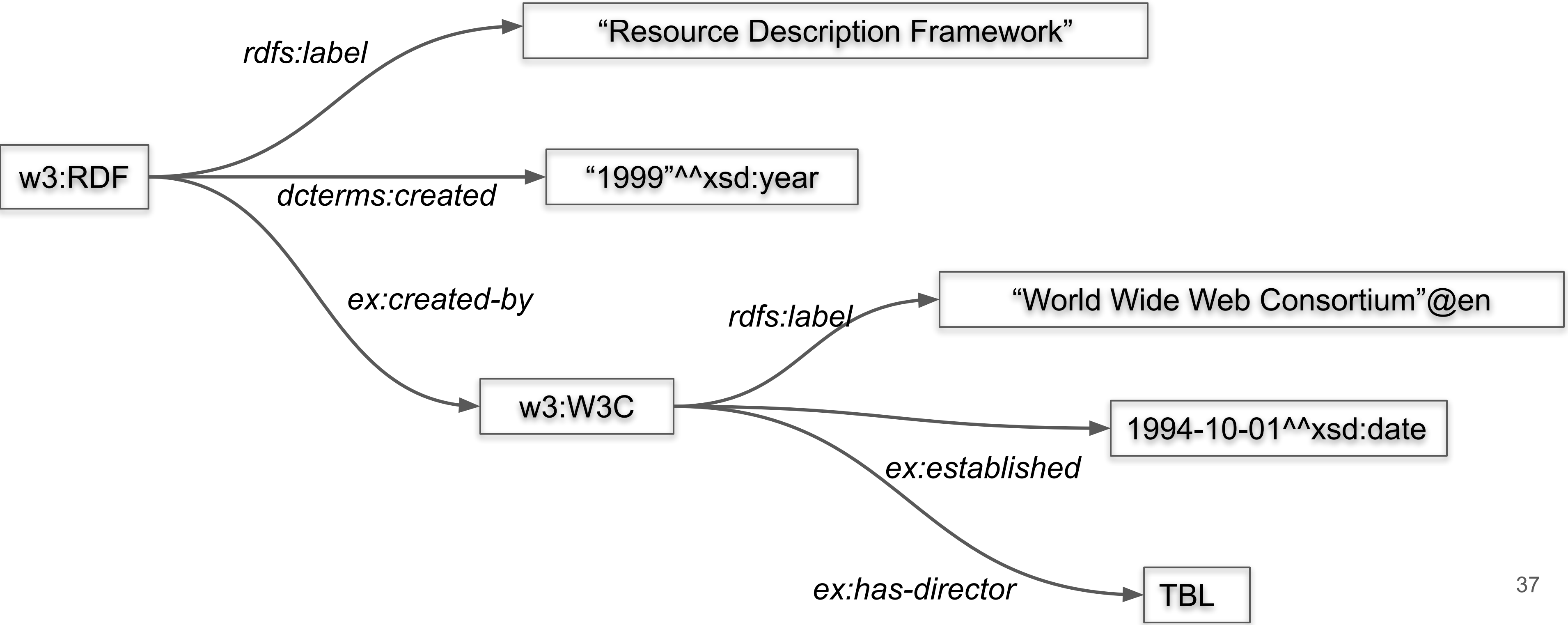
RDF/XML

```
<?xml version="1.0"?>
<rdf:RDF
   xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns# "
   xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema# "
   <rdf:Description rdf:about="https://www.w3.org/RDF ">
       <rdfs:label lang="en">Resource Description
Framework</rdfs:label>
   </rdf:Description>
</rdf:RDF>
```

Namespace declarations

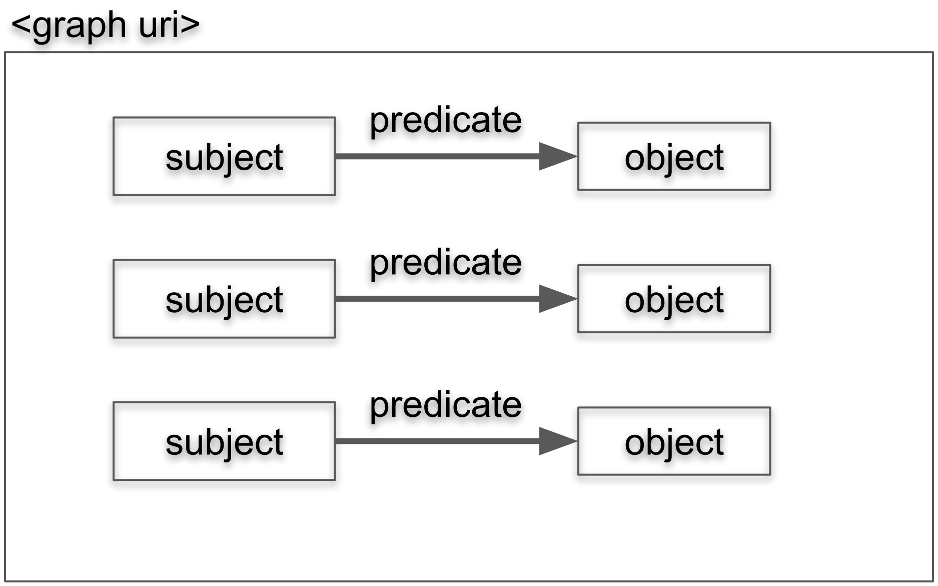
Uses the eXtensible Markup Language to represent RDF triples; can define and use namespaces.
A namespace in XML refers to a location on the web where resources are defined.

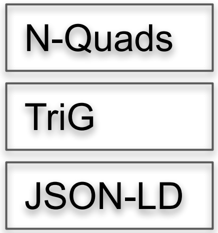
We can connect triples and a graph emerges where the node IRIs are shared



RDF Graphs

An **RDF Graph** is a **labeled, directed multigraph**. An **RDF Named Graph** is identified by an URI. The resources in this graph will share this URI as their common namespace





RDF Graph Syntaxes: N-Quads

N-Quads

```
<http://maastrichtuniversity.nl>
    <http://www.w3.org/2000/01/rdf-schema#label>
        "Maastricht University"
            <http://example.org/mygraph> .

<http://maastrichtuniversity.nl>
    <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
        <http://example.org/University>
            <http://example.org/mygraph> .
```

RDF Graph Syntaxes: TriG

TriG

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

:mygraph {
   <http://maastrichtuniversity.nl> a :University ;
       rdfs:label "Maastricht University" .
}
```

RDF Graph Syntaxes: JSON-LD

see
https://json-ld.org/playground/

JSON-LD

```
{
    "@context": {
        "ex": "http://example.org/ ",
        "rdfs": "http://www.w3.org/2000/01/rdf-schema# "
    },
    "@id": "ex:mygraph",
    "@graph": [
        {
            "@id": "http://maastrichtuniversity.nl ",
            "@type": "ex:University",
            "rdfs:label": "Maastricht University"
        }
    ]
}
```

slido

Which RDF syntax does not allow the specification of prefixes?

ⓘ Click **Present with Slido** or install our Chrome extension to activate this poll while presenting.

Building a KG from structured data

Report

crime	claimant	station	date
Pickpocketing	XY12SDA	Viña del Mar	2019-04-12
Assault	AB9123N	Arica	2019-04-12
Pickpocketing	XY12SDA	Rapa Nui	2019-04-12
Fraud	FI92HAS	Arica	2019-04-13

Claimant

id	name	country
XY12SDA	John Smith	U.S.
AB9123N	Joan Dubois	France
XI92HAS	Jorge Hernández	Chile

Figure 6.3: Relational database instance with two tables describing crime data

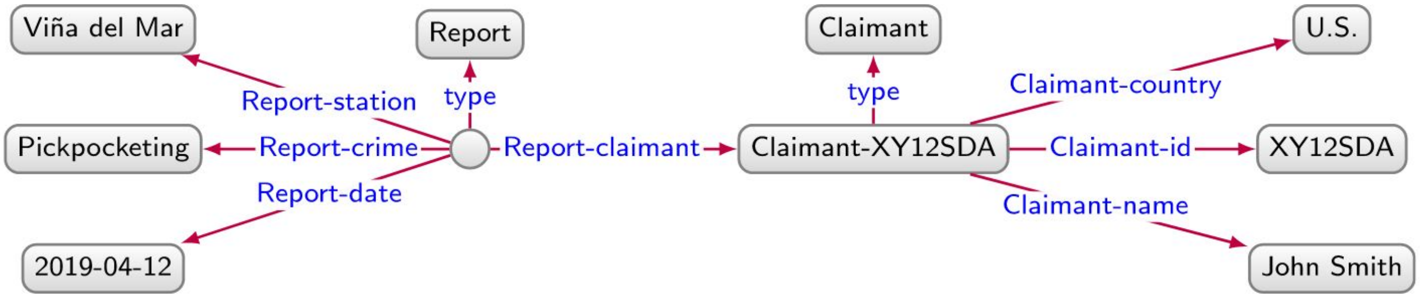


Figure 6.4: Direct mapping result for the first rows of both tables in Figure 6.3

Source: https://kgbook.org

Building a KG from structured data

Programming language and their RDF library. Allow to create URIs and triples, parse and save files in the different RDF formats

- Python: RDFLib
- PHP: EasyRDF
- JavaScript: various libraries listed at rdf.js.org
- R: rdflib (cf. introduction article)
- Java:
  - RDF4J: from the Eclipse foundation
  - Jena: from the Apache Foundation
  - SANSA stack: for large datasets using parallel processing on Spark, or Flink clusters
- Rust: oxigraph 🏗️

Building a KG from structured data

Mapping tools

- R2RML for SQL databases
  - Mapping language in RDF to convert SQL database to RDF
- RML and YARRRML for CSV, JSON, XML, SQL
  - Extension of R2RML to also convert popular data formats, such as CSV, JSON, XML
  - YARRRML is RML expressed as YAML, instead of RDF, making it easier to read and write for humans
- SPARQL-Generate to query structured files directly with SPARQL

The most popular currently is RML/YARRRML, with multiple different engines implemented to perform the mapping execution and generate the KG.

Building a KG from structured data

Common challenges:

- Graph Normal Form. fix typos, split multi-values
- map enumerations to controlled vocabularies, where possible. otherwise create your own vocabulary and publish it along with your data.
- Generate proper and persistent IRIs for your entities (i.e. purl, w3id)
- Find the right vocabulary terms and conceptual model to describe your data
- Link entities to others concepts/data on the web.

Storing and sharing knowledge

Data can be disseminated by **including it in HTML page**. For instance adding the JSON-LD describing a recipe in your website. That’s how Google recommend you to communicate them infos about your website.

But all this data can also be put as a whole into a **database** (aka. triplestore) for querying (with SPARQL)

ℹ️ A project to gather JSON-LD data from web pages: http://webdatacommons.org/structureddata

Linked Data Principles

1. Use **Uniform Resource Identifiers (URIs/URLs)** as identifiers for things
2. Use **HTTP URIs**, so that people can look up those entities
3. When someone looks up a URI, provide **useful information**, using Semantic Web standards
4. Include **links** to other URIs, so that they can discover more things

Summary