# Chapter 2

## RDF Syntax 1

# RDF Overview

- **RDF data model**
- **RDF syntax**
- **RDF serializations: XML, Turtle, N3, ntriples**
- RDF Schema (RDFS)
- Semantics of RDF and RDFS
  - Axiomatic Semantics
  - Operational semantics based on rules
- Querying RDF via SPARQL

# Introduction

- Problem: What does an XML document mean?
  - XML is about data structures
  - The meaning (semantics) not apparent to machines
- RDF is more a data model than a language
  - It is realized in many different formats
- RDF defines very basic semantics
  - RDFS and OWL define more RDF vocabulary for building rich data models
- RDF remains domain independent

# RDF History

- An early version was developed in 1995 by R. V. Guha at Apple
- Draft versions published by W3C in 1997-1998
- W3C recommendation in 1999
- RDF 1.1 (2014) is most recent specification

# Topics

- Basic concepts of RDF
  - Resources, properties, values, statements, triples
  - URIs and URIrefs
  - RDF graphs
  - Literals, qnames
- Vocabularies and modeling
  - Vocabularies
  - Blank nodes, data modeling, types, reification
  - Lists, bags, collections
- Serialization of RDF graphs
  - XML, Turtle, Ntriples
- Critique of RDF

# RDF Basics

- Core idea: identify resources using **Web identifiers** and describing resources in terms of simple **properties** and property **values**
- RDF data model is as a “pure” graph model
- To identify resources, RDF uses **Uniform Resource Identifiers (URIs**) and **URI references (URIrefs).**
- **Definition:** A **resource** is anything that is identifiable by a URIref

# Graphs: pure and impure

# Pure graph model

- A pure graph model consists only of **edges** between pairs of **nodes**

  Can be directed or undirected; can be labeled or not

- A graph can be represented as an unordered collection of (subject, predicate, object) triples

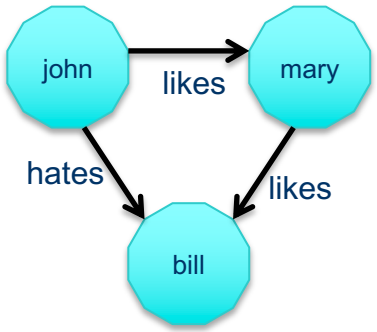
  If directed, predicate goes from subject to object

- Nodes not the subject or object of some triple are not allowed

  (John, likes, Mary),

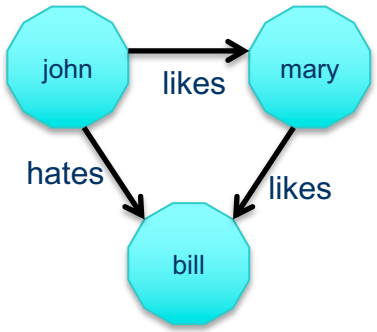
  (Mary, likes, Bill),

  ( John, hates, Bill)



# RDF graph model

- RDF is like this with a few caveats
  - Subjects and predicates are identified by URIs
  - Object can be a *URI* or a *literal*, i.e., a string or a number
- RDF defines some special URIs and gives them specific meaning, like this one for **type**
  - http://www.w3.org/1999/02/22-rdf-syntax-ns#type
- RDF has simple conventions for representing both ordered and unordered sequences and a few other data structures



# Property graphs

- Graph databases have become popular in past 10 years
- A common extension of the pure graph model is to allow both nodes and edges to have *properties*
- Simple version: properties are key/value pairs, e.g.
  - Age : 25
  - Date : “1990-09-21”
- We might give the **likes** edge from John to Mary two properties: *start* with value “1999-09-1” and *end* with value “2016-01-11”
  - Could mean the likes relation held between those two dates

# Some property graph technology

- Neo4J is perhaps the most widely used property graph
- OrientDB is another popular system that support for both a property graph and relational databases
- Apache's TinkerPop is an open source framework for querying and updating graph databases that is supported by most graph databases
- Amazon's Neptune is a graph database "built for the cloud" supporting both pure RDF and property graphs

# URIs and URIREFs

# Uniform Resource Identifiers (URIs)

- URIs identify resources on the Web
- Unlike URLs, they aren't limited to identifying things with network locations
- No organization controls who makes URIs or how they can be used
  - Some URI schemes (http: URLs) depend on centralized systems such as DNS name servers
  - Others are **completely decentralized**

# URI Reference (URIref)

- **URIref:** URI with optional fragment identifier at end, e.g:

  http://example.org/index.html#section2

- Fragment usecase:
  - HTML fragments refer to a place in a page
  - RDF fragments refer to resources in a RDF graph that the URI denotes, e.g., subjects, predicates or objects
    - http://www.w3.org/2004/02/skos/core : vocabulary for describing topics
    - http://www.w3.org/2004/02/skos/core#broader : the *broader* concept in SKOS Core vocabulary
- Like URLs, URIrefs may be either **absolute** or **relative**
  - Note: the empty URI refers to the resource it's in

# URIrefs in RDF

- RDF and Browsers use URIrefs to **identify things**, but interpret URIrefs slightly differently:
  - Browsers also use URIrefs to **retrieve** things
  - RDF uses URIrefs **only** to identify things and these might not even be retrievable
- **Linked Data** best practice is to use HTTP URIs that return RDF data for every URI

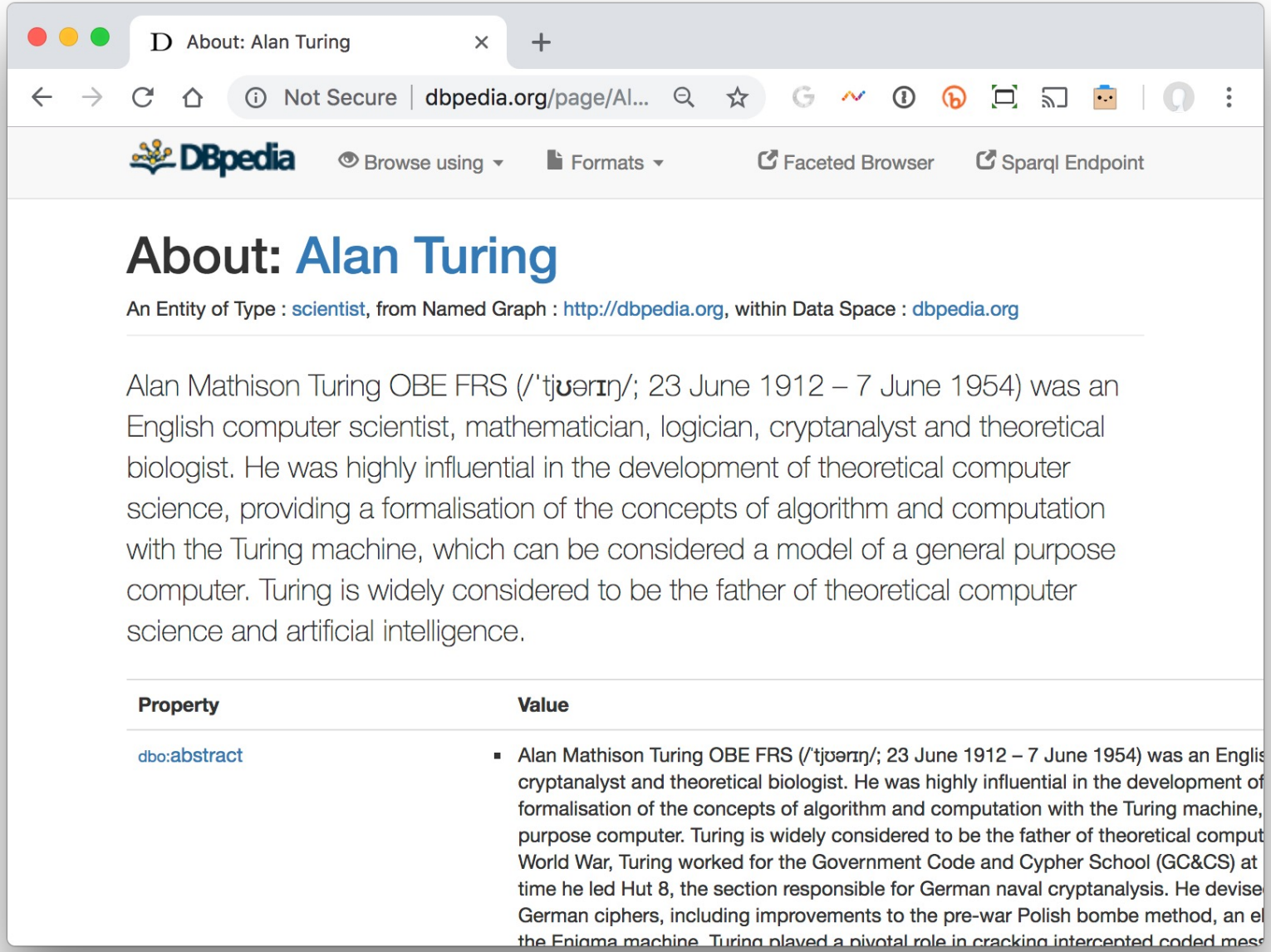
# Content Negotiation

- What does HTTP stand for?

# Content Negotiation

- What does HTTP stand for?
- HTTP == HyperText Transfer Protocol
- Lets Web client (browser, program) and server (apache) do many things (e.g., authentication)
- Can specify format of data returned, e.g., HTML, XML, RDF serialized in any of several forms, etc.
- Getting the same URL, http://dbpedia.org/resource /Alan_Turing, can produce content good for people or machines

# http://dbpedia.org/resource/Alan_Turing



# http://dbpedia.org/resource/Alan_Turing

- curl –L http://dbpedia.org/resource/Alan_Turing
  - -L says “follow redirects”
  - Returns default content version, typically html
- curl –LH "Accept: application/rdf+xml" *<url>*
  - Follow redirects
  - Return content as RDF serialized in xml if possible
- curl –LH "Accept: text/turtle, application/rdf+xml, text/ntriples, application/ld+json” *<url>*
  - *S*pecifies 4 possible content forms in preference order

# http://dbpedia.org/resource/Alan_Turing

**curl -LH "Accept: text/turtle" http://dbpedia.org/resource/Alan_Turing**

@prefix dbo: <http://dbpedia.org/ontology/> .
@prefix dbr: <http://dbpedia.org/resource/> .
dbr:Alan_turing dbo:wikiPageRedirects dbr:Alan_Turing .
<http://dbpedia.org/resource/A._Turing> dbo:wikiPageRedirects dbr:Alan_Turing .
dbr:Jack_Copeland dbo:knownFor dbr:Alan_Turing .
dbr:Joan_Clarke dbo:partner dbr:Alan_Turing .
dbr:Robin_Gandy dbo:doctoralAdvisor dbr:Alan_Turing .
dbr:Hilary_Putnam dbo:influencedBy dbr:Alan_Turing .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix yago: <http://dbpedia.org/class/yago/> .
dbr:Alan_Turing rdf:type yago:WikicatBritishCryptographers ,
yago:WikicatEnglishInventors ,
yago:Theorist110706812 ,
yago:Decoder109995398 .
@prefix umbel-rc: <http://umbel.org/umbel/rc/> .
dbr:Alan_Turing rdf:type umbel-rc:PersonWithOccupation .

...

# RDF Graphs

# RDF Graphs

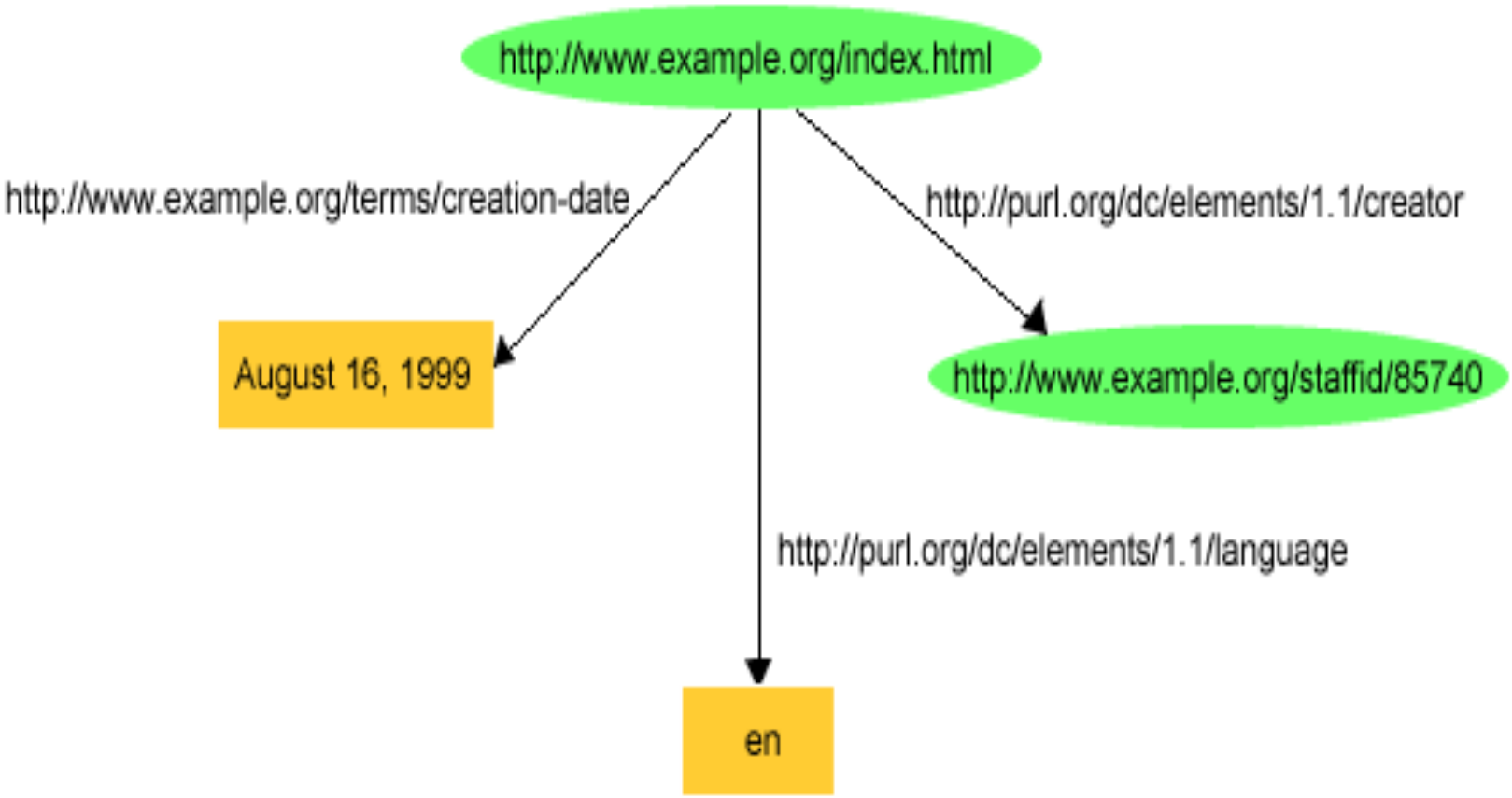
- RDF models statements by **nodes** and **arcs** in a **graph**
- A **statement** is represented by a node for the **subject**, a node for the **object** and an arc for the **predicate** (subject => object)
- A **node** may be identified by a **URIref** or it can be a **literal** or a **blank node**
- An **arc** is identified by a **URIref**
- **Note:** We will draw RDF graphs as **directed graphs**
  - But an arc can be the subject of an RDF statement
  - :has_parent  owl:inverseOf  :has_child

# Example

Consider the following statements:

- `http://www.example.org/index.html` has a creation-date whose value is August 16, 1999.
- `http://www.example.org/index.html` has a language whose value is English.
- `http://www.example.org/index.html` was created by hppt://example.org/staffed/85740

# The RDF Graph of the Example



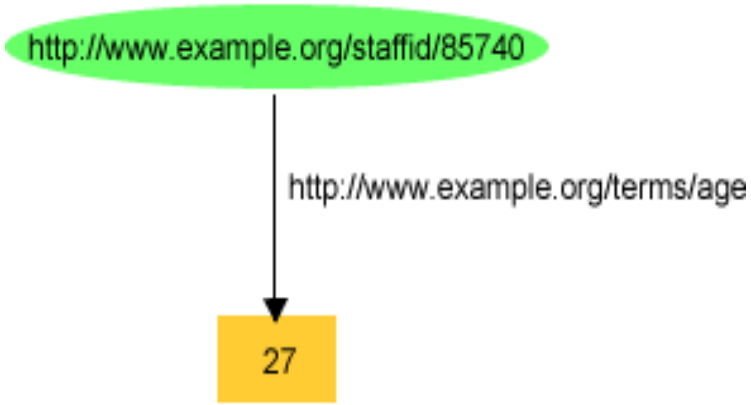

- Note: http://purl.org/dc/elements/1.1 is prefix for the Dublin Core vocabulary/ontology
- http://www.example.org/… is uses for examples

# RDF and Related Data Models

- In terms of the **relational model**, an RDF statement is like a **tuple in a relation** *Graph* with columns *Subject, Predicate, Object*
- For **first-order logic**, an RDF statement is like an **atomic formula** *triple(subj, pred, obj)* where *triple* is a FOL predicate and *subj, pred* and *obj* are constants
- More common view is to treat the triple's predicate as a logical predicate: *pred(subj, obj)*

# Literals and QNames

# Literals




What is 27? Number or string?

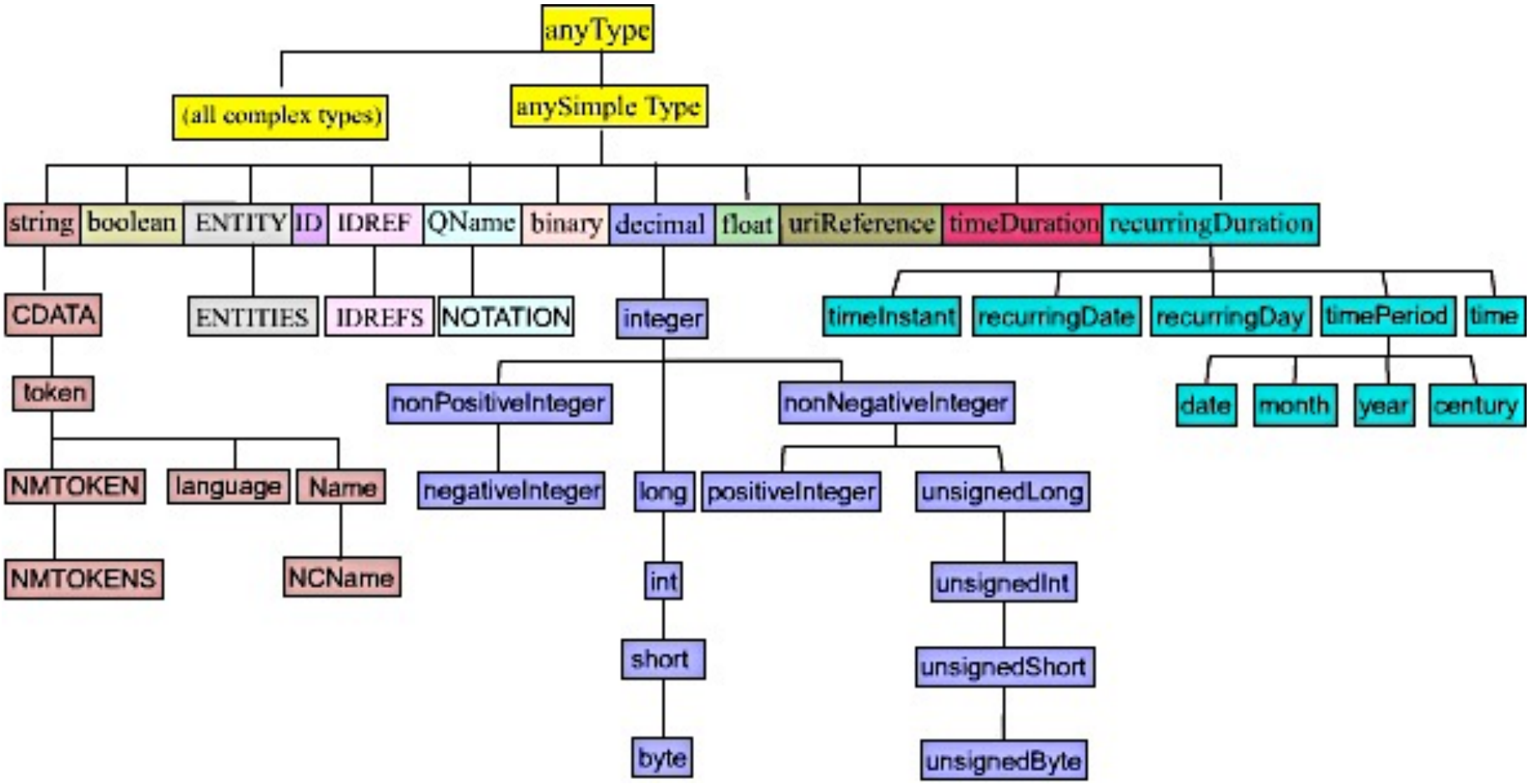
# Plain and Typed Literals

- RDF has two kinds of literals: **plain** and **typed**
- Plain literals have a **lexical form** (their lexical value) and optionally a **language tag**, e.g:
  - "27","Hello world"@en, "Bonjour le monde"@fr
- **RDF typed literals** are formed by pairing a string with a URIref for a particular XMLS **datatype**, e.g.:
  - “27”^^http://www.w3.org/2001/XMLSchema#integer
  - ”27"^^xsd:int

# Data Types for Literals

- In practice, the most widely used data typing scheme is the one by XML Schema
  - But **any** externally defined data typing scheme is allowed in RDF documents
- XML Schema predefines many data types
  - E.g. Booleans, integers, floating-point numbers, times, dates, etc.

# XMLSchema Datatypes




http://www.w3.org/TR/xmlschema-2/

# Qnames for URIrefs

- The ntriples notation results in very long lines
- We can use an **XML qualified name** (**QName**) w/o brackets for a full URI reference
  - http://dbpedia.org/page/Alan_Turing
  - dbp:Alan_Turing
- **Qnames** have a **prefix** that's been assigned to a **namespace URI**, a **colon** and a **local name**
  - How to assign a prefix to a URI varies by serialization
- The concepts of **names** and **namespaces** used in RDF originate in XML

# Topics Part 1

- Basic concepts of RDF
  - Resources, properties, values, statements, triples
  - URIs and URIrefs
  - RDF graphs
  - Literals, qnames
- Vocabularies and modeling
  - Vocabularies
  - Blank nodes, data modeling, types, reification
  - Lists, bags, collections
- Serialization of RDF graphs
  - XML, Turtle, Ntriples
- Critique of RDF