Lecture 1: Semantic Web in a nutshell

TIES4520 Semantic Technologies for Developers
Autumn 2018

University of Jyväskylä

Khriyenko Oleksiy

World Wide Web (WWW)

web page

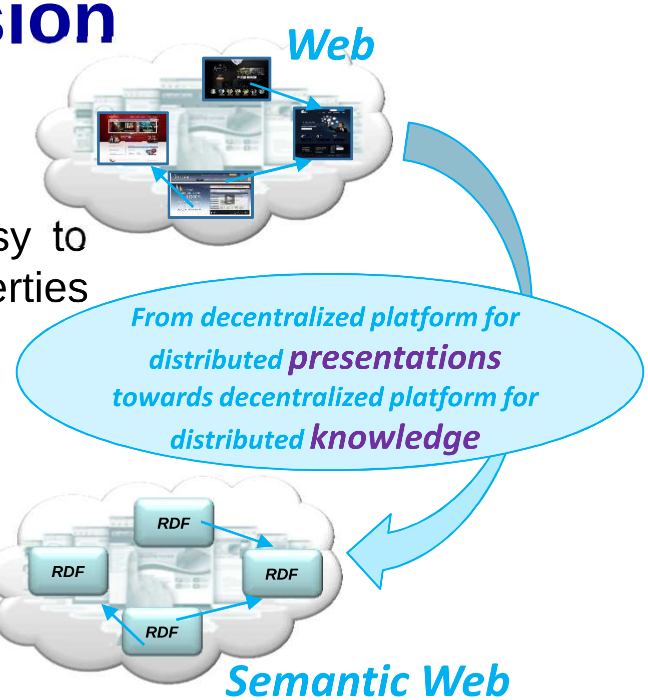
(document with some information)

World Wide Web

- **AAA principle** = Anybody can write Anything about Any topic
- Basic building block is a web page
- Any web page can refer to any other web page freely
- No central point of control
- No central repository. Documents scattered across the whole Web…

- ***Problem***: Web page is a ***document for humans***. For computers (machines) web pages are too *difficult to understand*

Semantic Web vision



- **Solution**
  - Let's produce Web data in a form that is easy to "digest" by a machine without losing good properties of WWW
- **How?**
  - Switch: informal representation => formal model
  - Connect information, but stay consistent
  - Distribute information (no central repository)
- **Semantics**
  - Relation between signs, words, symbols and the *things* (documents, people, places, events, organizations, concepts, etc.) to which they refer.
  - Relation of the things to each other.

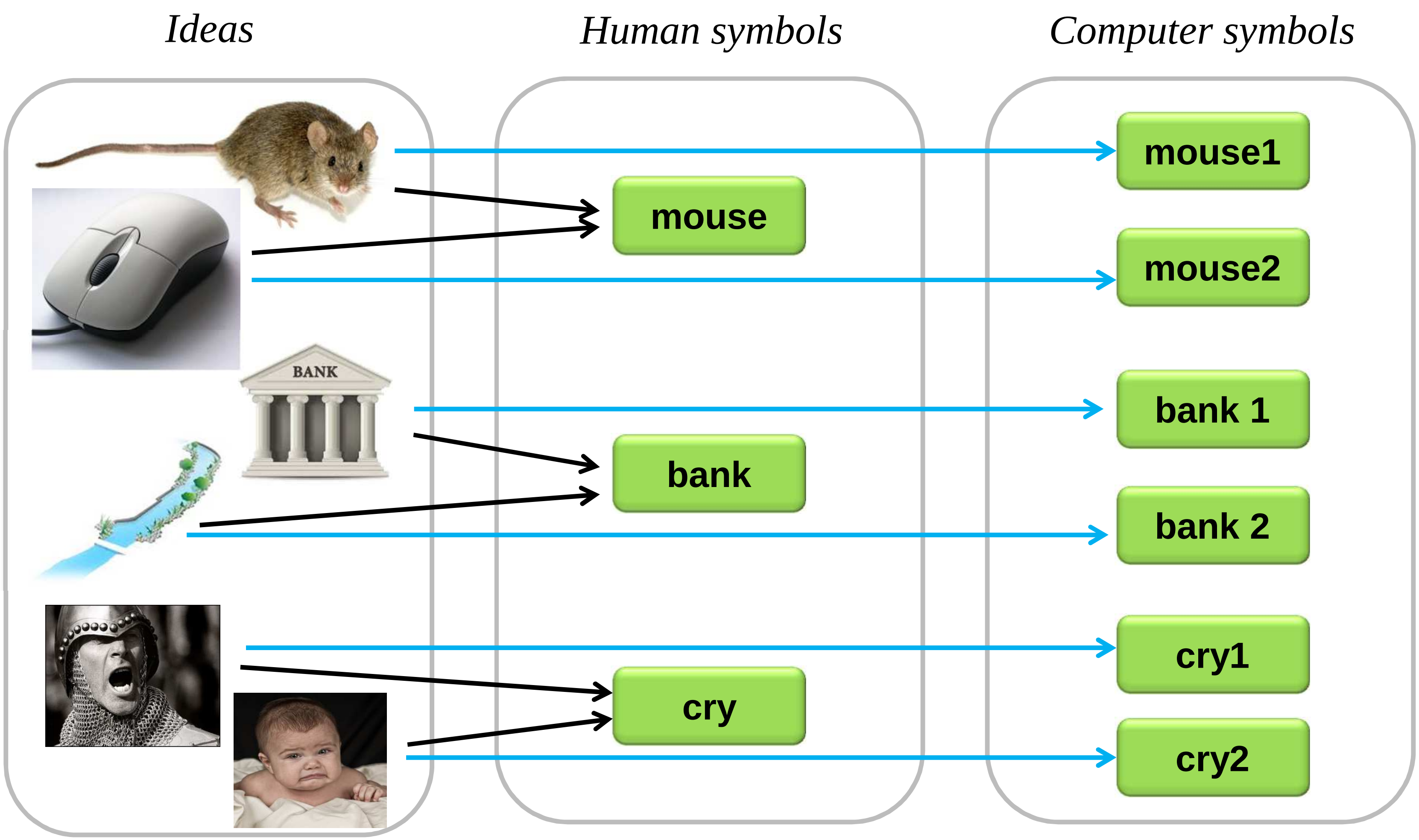
Syntax: *I love technology* *I ♥ technology*

Semantics: I am enjoying about learning and using new technology…

Challenges of Semantic Web

- We want:
  - Anybody can express data/knowledge about anything and connect it to anything
  - Web of distributed knowledge where the logical pieces of meaning can be manipulated by machines in a smart way
- We face these main problems:
  - How can I express some data so that it is disambiguous?
  - How can I refer to data (= connect it)?

Disambiguity of data (1)



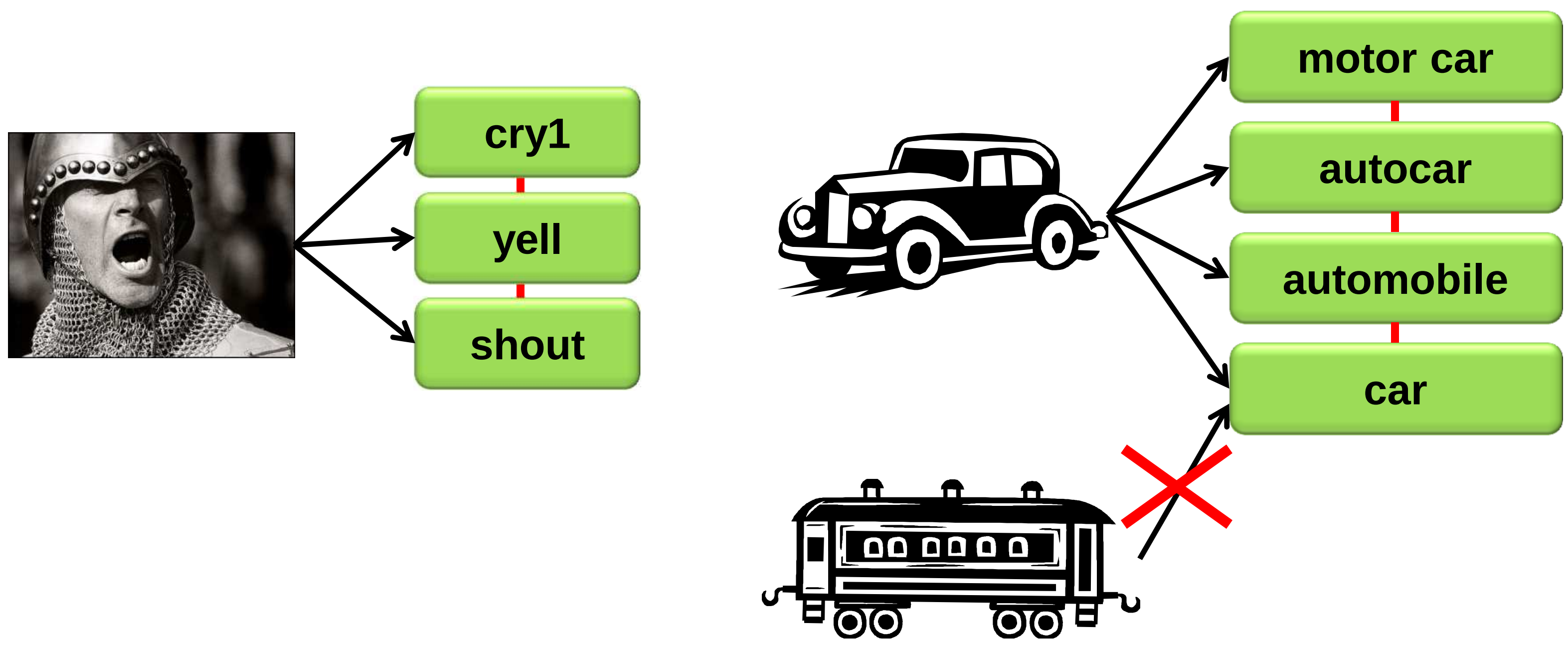
Disambiguity of data (2)

- Disambiguity of referencing to things:
  - Example: mouse, windows, cry, …
  - Every thing should have its *unique* name
  - Solution: URIs (Uniform Resource Identifiers)

- Disambiguity of concepts
  - Example:
    - John is *attracted to* Mary.
    - North pole is *attracted to* south pole.
  - There has to be some common understanding of the domain in question
  - Solution: *Ontology* – a precise explanation of terms and reasoning in a subject area.

Non-unique Naming Assumption

- Some resource (abstract idea or concrete thing) may have several names
- We have to *explicitly* tell the computer that these names mean the same thing



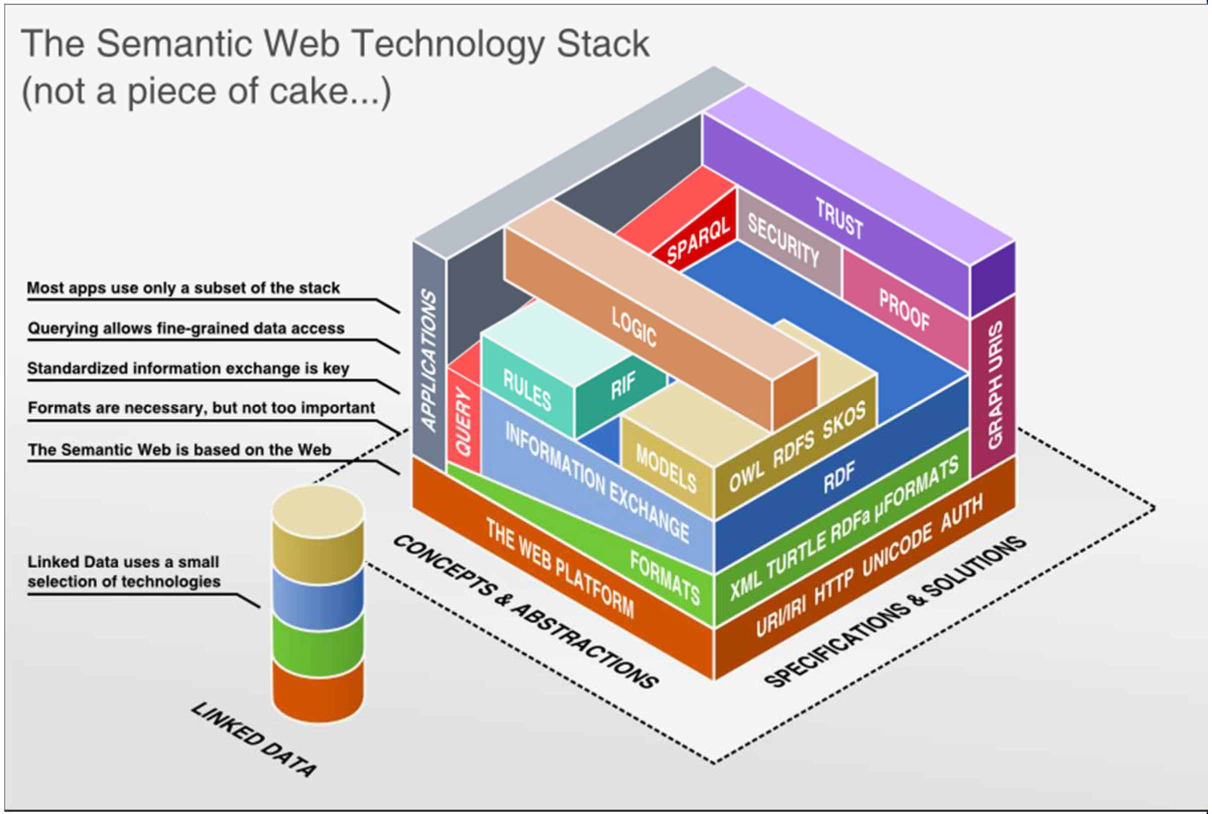
Open world assumption

- Any information can come at any point in the future
- We never know everything about the world. There can be true facts that are not contained in the knowledge base

	Open world	Closed world
Data	Helsinki and Tampere are in Finland. Paris is not in Finland.	
Q1	Is Helsinki in Finland?	
A1	YES	YES
Q2	Is Paris in Finland?	
A2	NO	NO
Q3	Is Jyväskylä in Finland?	
A3	I don’t know	NO

Half way summary

- Semantic Web
  - Anybody can express data about anything and connect it to anything
  - Data is readable and manipulated by machines
- Problems
  - Disambiguity of data => URIs + ontologies
- Properties
  - Non-unique Naming Assumption
  - Open World Assumption



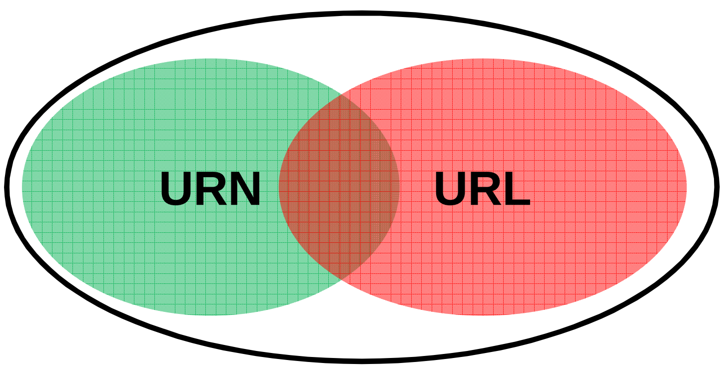
URI (Uniform Resource Identifier)

- **URI** globally identifies a certain entity (abstract or real)
- Special cases
  - ***URN*** (Uniform Resource Name)
    - Identifies the resource by naming it (URN is the name)
  - ***URL*** (Uniform Resource Locator)
    - Identifies the resource by locating it (URL is the address)
- Examples:

  http://bakery.com/breads/baguette

  isbn:0-12-385965-4

  http://users.jyu.fi/~olkhriye/ties4520/lectires/Lecture01.pdf



Namespaces

- Problem:
  - In one document you have these URIs:
    - *http://www.jyu.fi/people/students/john/assignments/assignment1*
    - *http://www.jyu.fi/people/students/john/assignments/assignment2*
    - *http://www.jyu.fi/people/students/john/assignments/assignment3*
  - Too long representation
- Solution:
  - Introduce a namespace as a prefix of the short (qualified name):

Full name: *http://www.jyu.fi/people/students/john/assignments/assignment1*

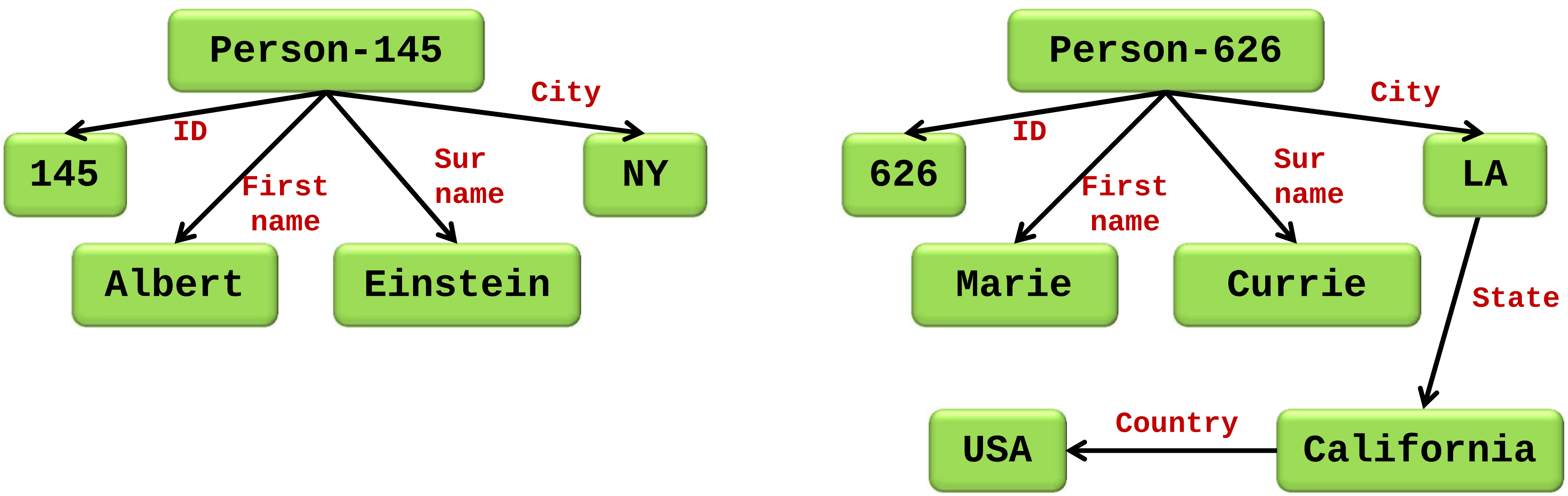
Prefix (for example as:) Rest of the name

  - Use qualified names (qnames):
    - *as:*assignment1
    - *as:*assignment2
    - *as:*assignment3
- Benefits:
  - Improved readability
  - Saves space

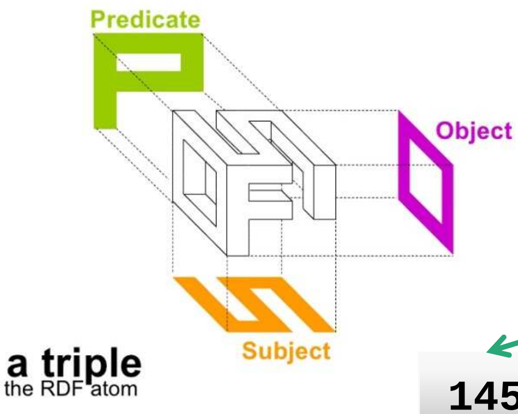
RDF (Resource Description Framework)

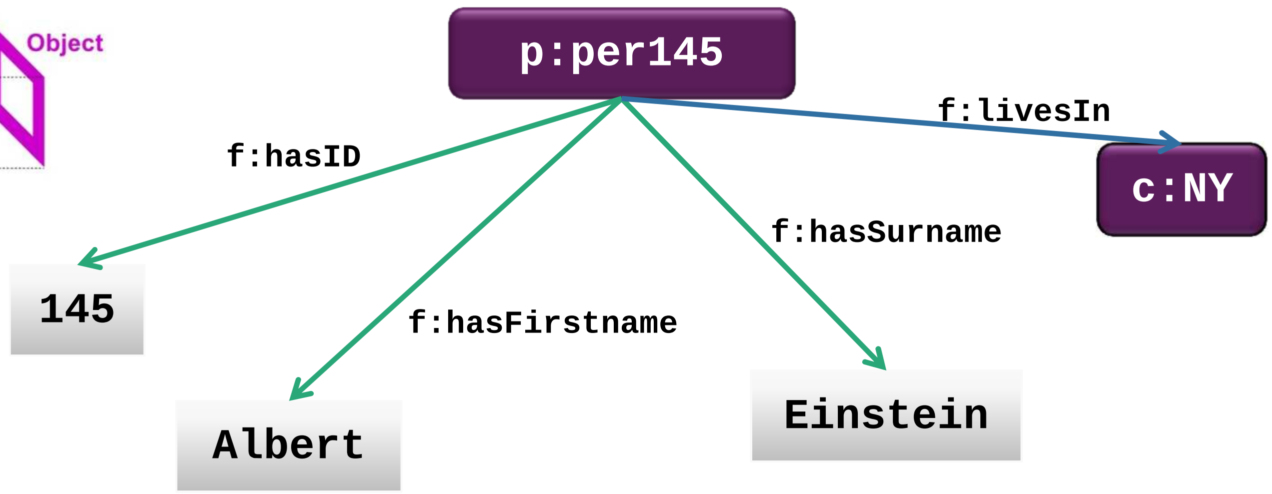
- RDF is a general method to decompose knowledge into small pieces with rules about the meaning of those pieces. It is a method to *describe facts in a short form*.
- Everything is a *Resource*
  - Anything that we can talk about and has identity in a form of URI.
  - Example: human, building, weather
- RDF represents graphs

ID	Firstname	Surname	City
145	Albert	Einstein	NY
626	Marie	Currie	LA



RDF as graph, RDF as text

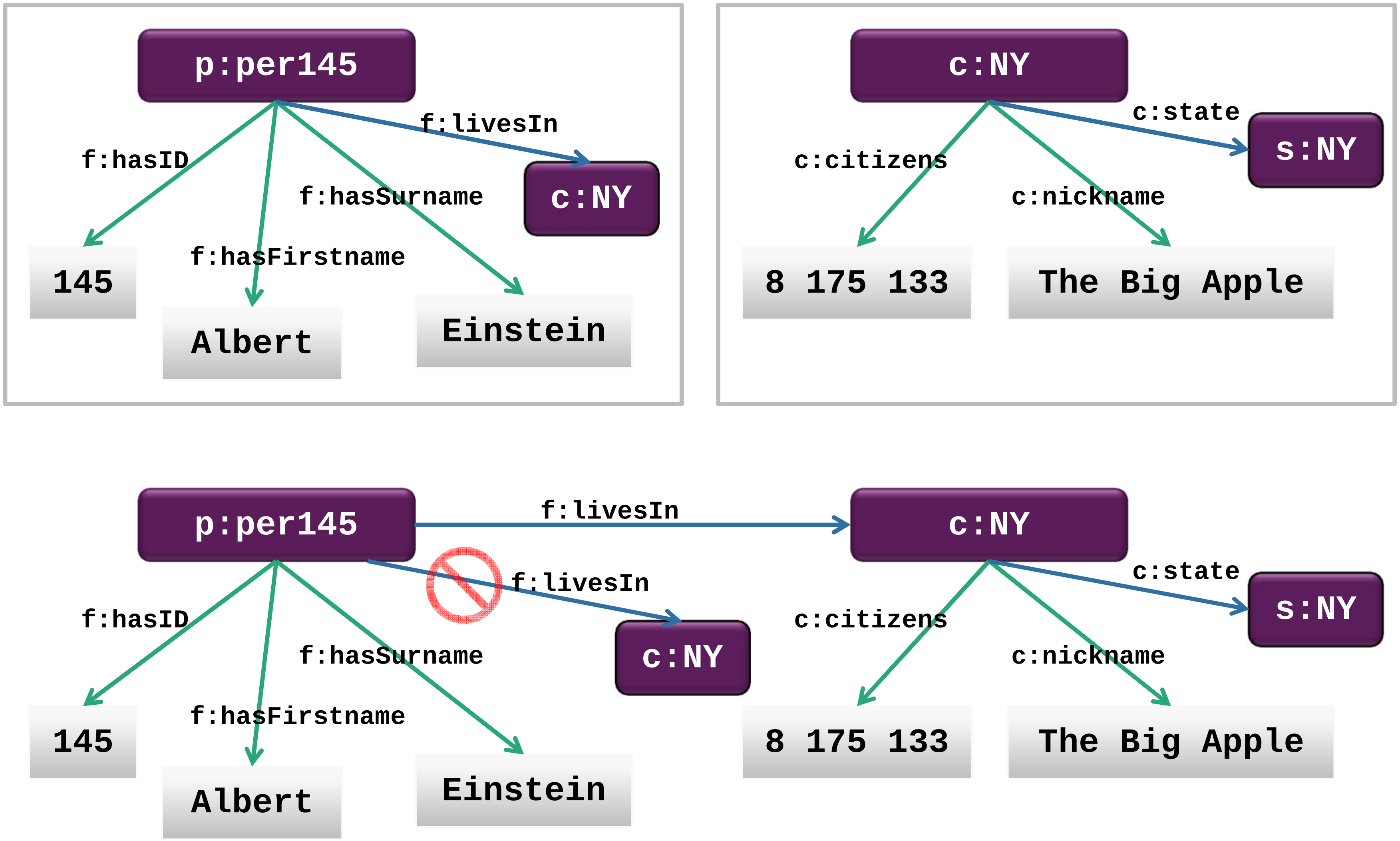




All the data in RDF is described in statements/triples:
subject* – *predicate* – *object

```
p:per145 f:hasID "145" .
p:per145 f:hasFirstName "Albert" .
p:per145 f:hasSurname "Einstein" .
p:per145 f:livesIn c:NY .
```

Graph matching and merging



Serialization

- The way of representing the graph in textual form
- Serializations (notations):
  - ***RDF/XML***
  - ***TriX***
  - ***N-triples***
  - ***Turtle*** *(Terse RDF Triple Language)*
  - ***Notation 3***

RDF/XML

- Suitable for machines
- Many XML parsers exist
- Difficult for humans to see *subject*-*predicate*-*object* triples

```
<rdf:RDF
  xmlns="http://data.gov/ontology/edu#"
  xmlns:log="http://www.w3.org/2000/10/swap/log#"
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
    <rdf:Description rdf:about="http://www.jyu.fi/courses/TIES4520">
        <credits>5</credits>
    </rdf:Description>
    <rdf:Description rdf:about="http://www.jyu.fi/people/Mary">
        <studies rdf:resource="http://www.jyu.fi/courses/TIES4520"/>
        <livesIn xmlns="http://data.gov/ontology/urban#"
                 rdf:resource="http://www.geo.com/city/Turku"/>
    </rdf:Description>
</rdf:RDF>
```

TriX notation (1)

- Another way of writing RDF in XML
- Contains named graphs
  - Named graphs contain triples
- Syntactically extendable
- More info: (*http://www.hpl.hp.com/techreports/2004/HPL-2004-56.html*)

- Why is RDF/XML not good enough?
  - RDF embedded in XHTML and other XML documents is hard to validate + other validation problems
  - Some unresolved syntactic issues (blank nodes as predicates, reification, literals as subjects, etc.)

TriX notation (2)

```
<TriX xmlns="http://www.w3.org/2004/03/trix/trix-1/">
  <graph>
    <uri>http://example.org/graph1</uri>
    <triple>
      <uri>http://example.org/John</uri>
      <uri>http://example.org/loves</uri>
      <uri>http://example.org/Mary</uri>
    </triple>
    <triple>
      <uri>http://example.org/John</uri>
      <uri>http://example.org/hasName</uri>
      <plainLiteral>John</plainLiteral>
    </triple>
    <triple>
      <uri>http://example.org/John</uri>
      <uri>http://example.org/hasAge</uri>
      <typedLiteral datatype="http://www.w3.org/2001/XMLSchema#integer">
        32
      </typedLiteral>
    </triple>
  </graph>
</TriX>
```

Graph URI optional

S

P

O

N-triples

- Simple textual serialization of RDF statements
- Each statement consists of *subject*, *predicate* and *object* separated by a white space
- Statements are separated by dots (**.**)
- Resources are referred to with full URIs in **<>** brackets
- Literals are wrapped into double quotes (**“ ”**)

```
<http://www.jyu.fi/people/Mary> <http://data.gov/ontology/urban#livesIn>
   <http://www.geo.com/city/Turku> .
<http://www.jyu.fi/people/Mary> <http://data.gov/ontology/edu#studies>
   <http://www.jyu.fi/courses/TIES4520> .
<http://www.jyu.fi/courses/TIES4520> <http://data.gov/ontology/edu#credits>
   “5” .
```

Turtle (1)

- ***Turtle*** is similar to N-triples, but even more compact
- Uses ***@prefix*** to define the prefix and later on use just qualified names (e.g. *fam:John*)

```
@prefix   p: <http://www.jyu.fi/people/> .
@prefix   u: <http://data.gov/ontology/urban#> .
p:Mary u:hasAge "25" .
```

- Abbreviated form of triples with *semantic sugar*:
  - Semicolon (**;**) to separate statements about the same subject

```
x:Mary x:hasAge "25"; x:gender x:female; x:likes x:chocolate .
```

  - Comma (**,**) to separate statements about the same subject with the same predicate

```
x:Mary x:likes x:chocolate, x:cheese, x:bread.
```

- And more features: *http://www.w3.org/TeamSubmission/turtle/*
  http://www.w3.org/TR/turtle/

Turtle (2)

- Same example as before:

```
<http://www.jyu.fi/people/Mary> <http://data.gov/ontology/urban#livesIn>
   <http://www.geo.com/city/Turku> .
<http://www.jyu.fi/people/Mary> <http://data.gov/ontology/edu#studies>
   <http://www.jyu.fi/courses/TIES4520> .
<http://www.jyu.fi/courses/TIES4520> <http://data.gov/ontology/edu#credits>
   "5" .
```

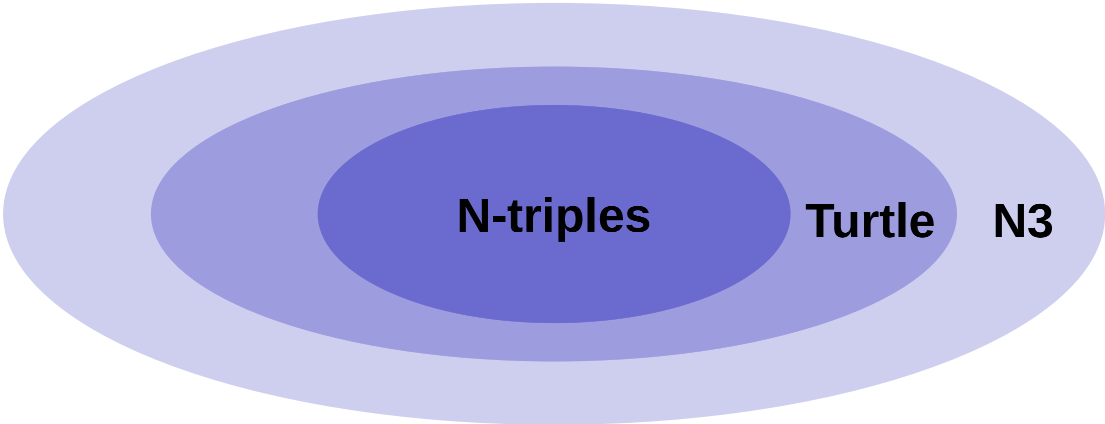
```
@prefix   p: <http://www.jyu.fi/people/> .
@prefix   u: <http://data.gov/ontology/urban#> .
@prefix edu: <http://data.gov/ontology/edu#> .
@prefix  co: <http://www.jyu.fi/courses/> .
@prefix  ci: <http://www.geo.com/city/> .

p:Mary u:livesIn ci:Turku .
p:Mary edu:studies co:TIES4520 .
co:TIES4520 edu:credits "5" .
```

```
p:Mary u:livesIn ci:Turku ; edu:studies co:TIES4520 .
co:TIES4520 edu:credits "5" .
```

Notation 3

- ***Notation 3*** (superset of Turtle) is the most important RDF notation, most clearly captures the abstract graph.
- Introduces formula *{ … }*
  - E.g. to make statements about statements
- New logic-related predicates (e.g. *=>*, *@forSome*, etc.)
- Variables allowed (*?varName*)
- Suffix: **.n3*
- Relationship to other notations:



Three rules of RDF

1. Every fact is expressed as a triple (subject, predicate, object)
2. Subjects, predicates and objects are names for entities represented as URIs
3. Objects can be literal values as well (***subjects and predicates are not!!!***)

```
<urn:isbn:1-12-456789-0> <http://www.jyu.fi/library#hasAuthor> “John” .
```

Subject — ***(URI)***

Predicate — ***(URI)***

Object — ***(URI or Literal value)***

```
c:TIES4520 co:hasLecture lec:Lecture1 .
```

```
c:TIES4520 co:hasCode “TIES4520” .
```

```
p:per673 fam:hasSurname “Doe” .
```

```
“John” fam:hasAge “25” .
```

Literals: Language Tags

Literal values are raw text that can be used instead of objects in RDF triples.

- *Plain literal* represents string:
  - *language tag*, to specify what language the raw text is written in;

```
:jone :name  "John" .
```
*name is a language-less, datatype-less **raw text value**.*

```
:jone :name  "John"@en .
```
*name **in English**, is John.*

```
:jone :name  "Jacque"@fr .
```
*name **in French**, is Jacque.*

```
:a :b "The first line\nThe second line\n more" .
:a :b """The first line
      The second line
      more""" .
```
"simple literal".

"""long literal""".

Literals: Datatype

Literal values are raw text that can be used instead of objects in RDF triples.

- *Datatype literal* represents non-string values (e.g. boolean, numbers, date or time, etc.) with *datatype* that indicates how to interpret the raw text.
  - Lexical form of the literal + URI of the datatype (datatypes defined in *XML Schema* are used by convention: *xsd:integer, xsd:decimal, xsd:float, xsd:double, xsd:string, xsd:boolean, xsd:dateTime*, etc.).

`"1234" .`	*An untyped literal value. No datatype.*
`"1234"^^xsd:integer .`	*A typed literal value using a namespace.*
`"1234"^^<http://www.w3.org/2001/XMLSchema#integer> .`	*The same with the full datatype URI.*

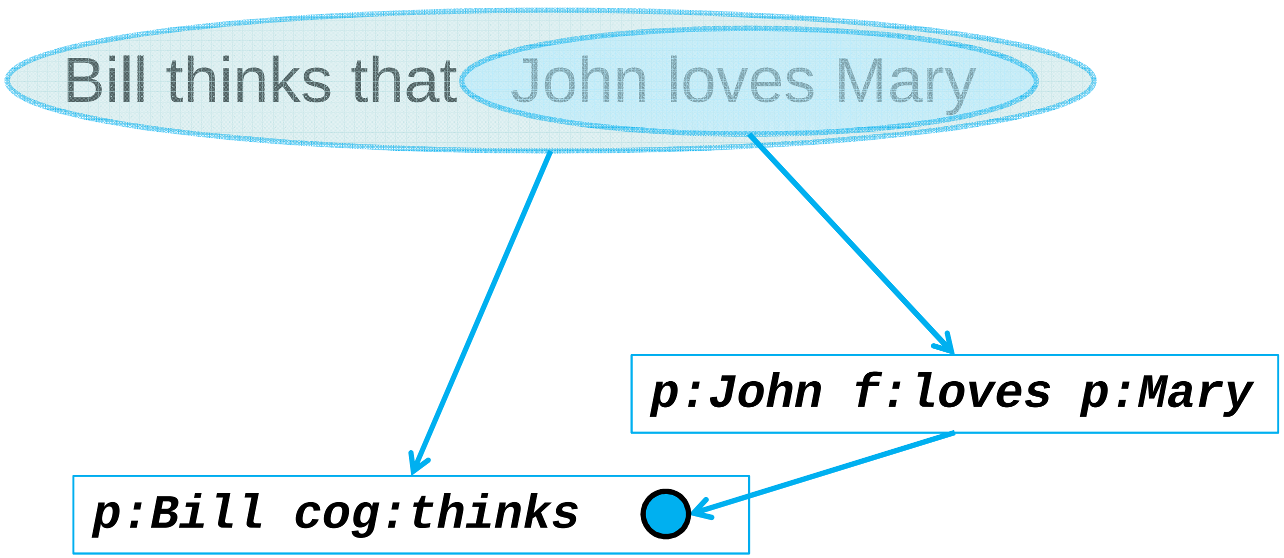
	Datatype	Value space (informative)
Core types	xsd:string	Character strings
	xsd:boolean	true, false
	xsd:decimal	Arbitrary-precision decimal numbers
	xsd:integer	Arbitrary-size integer numbers
IEEE floating-point numbers	xsd:double	64-bit floating point numbers incl. ±Inf, ±0, NaN
	xsd:float	32-bit floating point numbers incl. ±Inf, ±0, NaN
Time and date	xsd:date	Dates (yyyy-mm-dd) with or without timezone
	xsd:time	Times (hh:mm:ss.sss…) with or without timezone
	xsd:dateTime	Date and time with or without timezone
	xsd:dateTimeStamp	Date and time with required timezone
Recurring and partial dates	xsd:gYear	Gregorian calendar year
	xsd:gMonth	Gregorian calendar month
	xsd:gDay	Gregorian calendar day of the month
	xsd:gYearMonth	Gregorian calendar year and month
	xsd:gMonthDay	Gregorian calendar month and day
	xsd:duration	Duration of time
	xsd:yearMonthDuration	Duration of time (months and years only)
	xsd:dayTimeDuration	Duration of time (days, hours, minutes, seconds only)
Limited-range integer numbers	xsd:byte	-128…+127 (8 bit)
	xsd:short	-32768…+32767 (16 bit)
	xsd:int	-2147483648…+2147483647 (32 bit)
	xsd:long	-9223372036854775808…+9223372036854775807 (64 bit)
	xsd:unsignedByte	0…255 (8 bit)
	xsd:unsignedShort	0…65535 (16 bit)
	xsd:unsignedInt	0…4294967295 (32 bit)
	xsd:unsignedLong	0…18446744073709551615 (64 bit)
	xsd:positiveInteger	Integer numbers >0
	xsd:nonNegativeInteger	Integer numbers ≥0
	xsd:negativeInteger	Integer numbers <0
	xsd:nonPositiveInteger	Integer numbers ≤0
Encoded binary data	xsd:hexBinary	Hex-encoded binary data
	xsd:base64Binary	Base64-encoded binary data
Miscellaneous XSD types	xsd:anyURI	Absolute or relative URIs and IRIs
	xsd:language	Language tags per [BCP47]
	xsd:normalizedString	Whitespace-normalized strings
	xsd:token	Tokenized strings
	xsd:NMTOKEN	XML NMTOKENs
	xsd:Name	XML Names
	xsd:NCName	XML NCNames

Important:

`:jone :age "10"^^xsd:float .`	is the **same** as	`:jone :age "10.000"^^xsd:float .`
`:jone :age "10" .`	is **not** the **same** as	`:jone :age "10.000"^^xsd:float .`
`:jone :taxesPayed "true" .`	is **not** the **same** as	`:jone :taxesPayed "true"^^xsd:boolean .`

Reification problem

- Making statement about a statement or more statements
- Used to represent the context
- *Example:*



Solutions to reification

- Referring to the statement as a resource:

```
p:Bill cog:thinks s:st1 .
s:st1 rdf:type rdf:Statement.
s:st1 rdf:subject p:John .
s:st1 rdf:predicate f:loves .
s:st1 rdf:object p:Mary .
```

valid in Turtle
valid in Notation3

- Using special context brackets:

```
p:Bill cog:thinks {p:John f:loves p:Mary} .
```

***not** valid in Turtle*
valid in Notation3

Blank nodes

- When we want to annotate a resource whose identity is unknown we use *blank node*.
- Example: Larry has a fast red car.

```
p:Larry p:owns ? .
?   rdf:type v:Car; v:color c:red; v:speed s:fast .
```

Use abbreviation (*[ ]*) for anonymous blank node, if there is no need to reuse the blank node:

```
p:Larry p:owns [
  rdf:type v:Car
  ; v:color c:red
  ; v:speed s:fast .] .
```

Use a blank node identifier (***_:name***), that allows the same blank node to be referred to in more then one place:

```
p:Larry p:owns _:xe46na54an .
_:xe46na54an rdf:type v:Car
   ; v:color c:red
   ; v:speed s:fast .
```

Turtle abbreviations

- ***@prefix*** – abbreviation for prefix definition
- ***a*** – abbreviation for the property<http://www.w3.org/1999/02/22-rdf-syntax-ns#type>

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

p:Larry <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> p:Human .
p:Larry rdf:type p:Human .
p:Larry a p:Human .
```

- ***,*** – is used to repeat the *subject* and the *predicate* of triples that only differ in the *object*

```
p:Larry p:owns v:Car , v:bike.
```

- ***;*** – is used to repeat the *subject* of triples that vary in the *predicate* and the *object*

```
p:Larry a p:Human ; p:owns v:Car .
```

- ***[ ]*** – abbreviation for the blank node

```
p:Larry p:owns [rdf:type v:Car ; v:color c:red ; v:speed s:fast .].
```

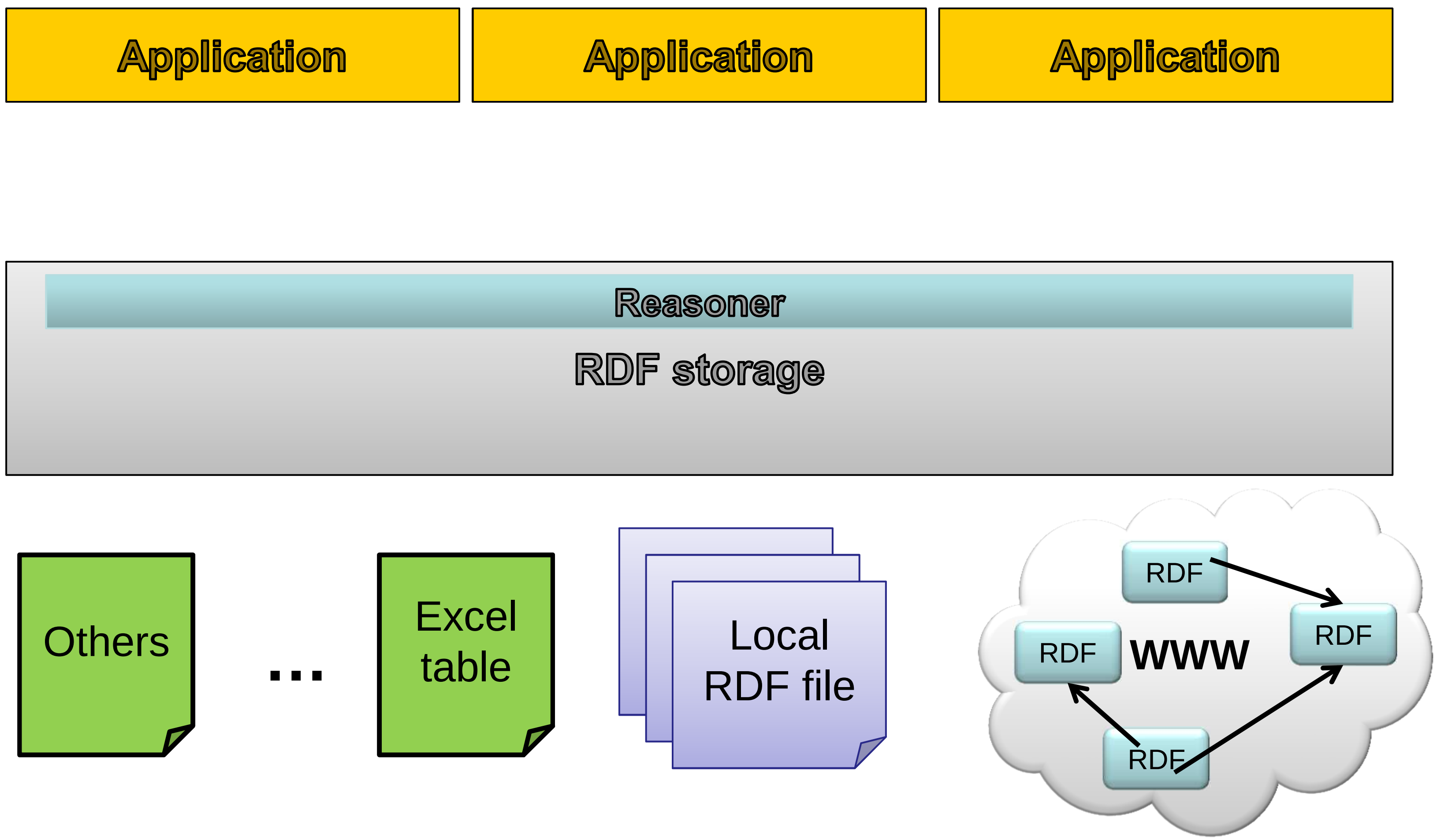
- ***( )*** – abbreviation for RDF Collection (structure for lists of RDF nodes)

```
cd:Beatles cd:artist ( b:George b:John b:Paul b:Ringo ).
```

Manipulation of RDF data

- Storing
  - RDF file on the web
  - Specialized RDF storage (“RDF database”)
  - Other form (*.xls, DB, …) exposed as RDF
- Querying
  - Like in relational DB there is a query language (SPARQL)
  - Can query from several sources (web sources, local RDF storages, etc.)
- Reasoning
  - Can derive new facts from already existing facts
  - Can check consistency of the model
  - Does not exist in relational DB !!!

RDF for a machine



Simple Task

- Solve small tasks about RDF and its serializations
  - Convert it from N-triple to Turtle (try to save space by using abbreviated syntax)
  - Convert it to a graph (e.g. draw on a piece of paper)

```
<http://transport.data.gov.uk/id/stop-point/9100BAYFORD>
<http://transport.data.gov.uk/def/naptan/station>
<http://transport.data.gov.uk/id/station/BAY>.
<http://transport.data.gov.uk/id/stop-point/9100BAYFORD>
<http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
<http://transport.data.gov.uk/def/naptan/StopPoint>.
<http://transport.data.gov.uk/id/stop-point/9100BAYFORD>
<http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
<http://transport.data.gov.uk/def/naptan/RailAccessArea>.
<http://transport.data.gov.uk/id/station/BAY>
<http://transport.data.gov.uk/def/naptan/crsRef> "BAY" .
<http://transport.data.gov.uk/id/station/BAY>
<http://transport.data.gov.uk/def/naptan/tiplocRef> "BAYFORD" .
<http://transport.data.gov.uk/id/station/BAY>
<http://www.w3.org/2004/02/skos/core#prefLabel> "Bayford Rail Station" .
```

Further reading

- RDF: **What is RDF and what is it good for?**
  https://github.com/JoshData/rdfabout/blob/gh-pages/intro-to-rdf.md#
- Semantic web: *http://www.youtube.com/watch?v=OGg8A2zfWKg*
- N-triples: *http://www.w3.org/2001/sw/RDFCore/ntriples/*
- Turtle: *http://www.w3.org/TeamSubmission/turtle/* ; *http://www.w3.org/TR/turtle/*
- Notation3: *http://www.w3.org/DesignIssues/Notation3.html*
- TriX: *http://www.hpl.hp.com/techreports/2004/HPL-2004-56.html*
- Datatypes: *https://www.w3.org/TR/2013/WD-rdf11-concepts-20130115/#xsd-datatypes*
- Linked data: *http://www.youtube.com/watch?v=qMjkI4hJej0*