Internet Engineering

Tomasz Babaczyńskii, Zofia Kruczkiewicz
Tomasz Kubik

Information systems modelling – UML and service description languages

Choose yourself and new technologies

HUMAN CAPITAL
HUMAN – BEST INVESTMENT!

Wrocław University of Technology

EUROPEAN SOCIAL FUND

Project co-financed from the EU European Social Fund

RDF -Resource Description Framework

RDF

- It is a framework for representing information about resources in a graph form
  - primarily intended to describe metadata about WWW resources
  - can be used to describe anything that can be **identified** on the Web, even if it is not accessible
- RDF description can be read and understand by computers, and exchanged without loss of meaning
- W3C maintains RDF in the scope of its Semantic Web Activity

HUMAN CAPITAL
HUMAN – BEST INVESTMENT!
Wrocław University of Technology
EUROPEAN SOCIAL FUND
Project co-financed from the EU European Social Fund

RDF facts

- W3C Recommendation on 10 February 2004
  - RDF-CONCEPTS,
  - RDF-SYNTAX,
  - RDF-VOCABULARY,
  - RDF-SEMANTICS,
  - RDF-TESTS, RDF-PRIMER

HUMAN CAPITAL
HUMAN – BEST INVESTMENT!
Wrocław University of Technology
EUROPEAN SOCIAL FUND
Project co-financed from the EU European Social Fund

Resource

- RDF can represent anything that can be named: an object, act, or concept:
  - Abstract ideas as “classess”, “properties”, “groups”, “sets”
  - Abstract terms, as “love” or “money”
  - Real world objects, as “rock” or “sand”
  - Web resourvces
  - …
- In RDF sense “named” means “identified by URI”
- RDF uses URIs to identify resources, or, more precisely, URIRefs.

URI

- Uniform Resource Identifier is a global, rigid resource identifier with the following syntax:

```
scheme ":" hier_part ["?" query ] [ # fragment ]
```

`scheme` – string (a letter followed by letters, digits, and `["+"|"."|"-"]`)

`hier_part` – has the following syntax:

```
[userInfo "@"] hostname [:port_number] [path]
```

`query` – optional information, commonly organized as a sequence of: `<key>=<value>` pairs, separated by `";"` or `"&"`

`fragment` – optional part (local reference)

Example:

```
http://example.org/family.rdf#fatherOf
```

URIRef

- The syntax of URI has been defined in [RFC2396] and updated in [RFC2732]
- The current generic URI syntax specification is [RFC3986].
- URI can be absolute or relative:
  - Absolute : a resource is identified with full and context independent resource reference
  - Relative : a reference has not given full information to identify a resource and missing information must be derived from the context
- A URIRef is relative form of URI
  - consists of URI and optional `fragment` preceded by #
  - absolute URI of `#section2` from the document `http://www.example.org/index.html` is `http://www.example.org/index.html#section2`

IRI

- Internationalized Resource Identifier is a a complement to URI
- provides wider repertoire of characters allowed
  - Unicode/ISO10646 characters beyond U+007F
  - private characters of that set can occur only in query parts
- Standardized in [RFC3987] that defines "internationalized" versions corresponding to other constructs from [RFC3986], such as URI references.
- In many cases URI and IRI are used interchangeably, but practical replacement of URIs (or URI references) by IRIs (or IRI references) depends on the application.

Statement

- RDF is based on *Graph data model*
- Represented by the *triple*: *<subject, predicate, object>*.
  - The triple links one object (*subject*) to another object (*object*) or a literal via a property (*predicate*).
  - In other words: a resource (*subject*) has a property (*predicate*) valued by property value (*object*).
- RDF requires that:
  - *subject* has URI or is *b-node*;
  - *predicate* has URI;
  - *object* has URI, is *b-node* or is literal.
- The same URI can be assigned to a node and to an arc as well.

Remarks on statements

- RDF model = set of RDF triples
- triple = (subject, predicate, object) = statement
  - subject = resource
  - predicate = property (of the resource)
  - object = value (of the property)
- URIRefs identify subject, predicate, and object

Remarks on namespaces

- Some domain names appearing in the URL authority part have been reserved for testing or other similar uses [RFC2606]
- `example.com`, `example.net`, or `example.org` do not refer to any existing resources but serve for illustrative purposes

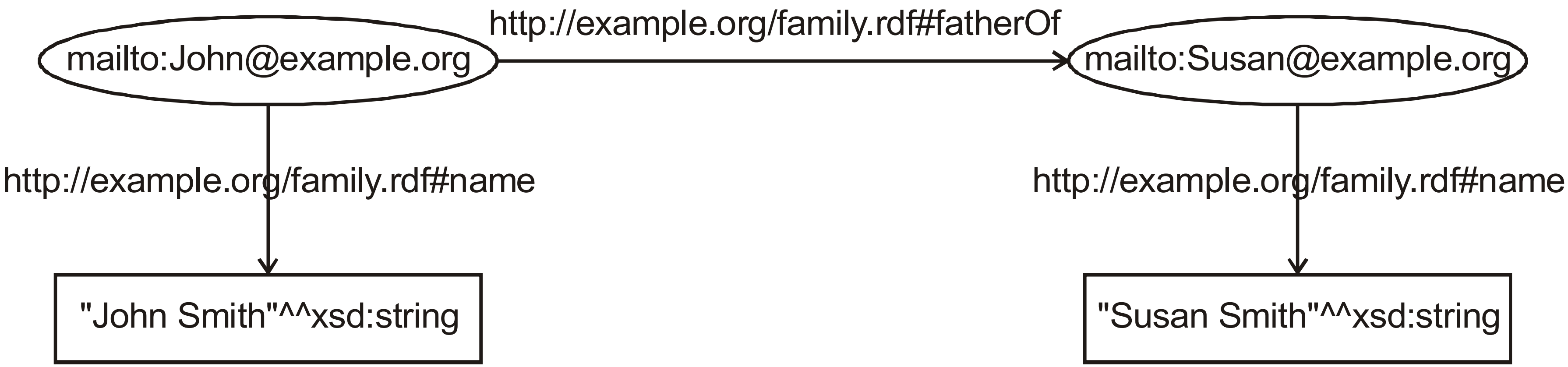
RDF Graph

- Graphical representation of a triple
  - subject and object nodes are resources

  - object node is a literal

RDF graph example

- “John Smith is a father of Susan Smith”

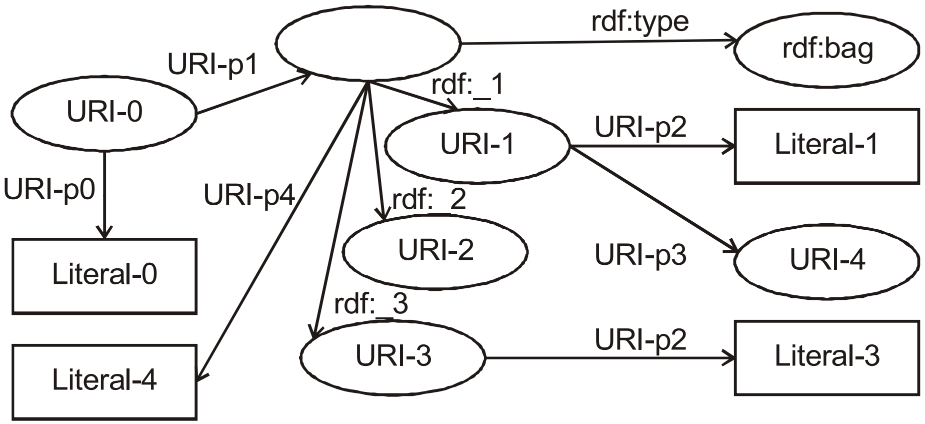


- Both persons are identified by their e-mail addresses and their names are provided as typed literals.

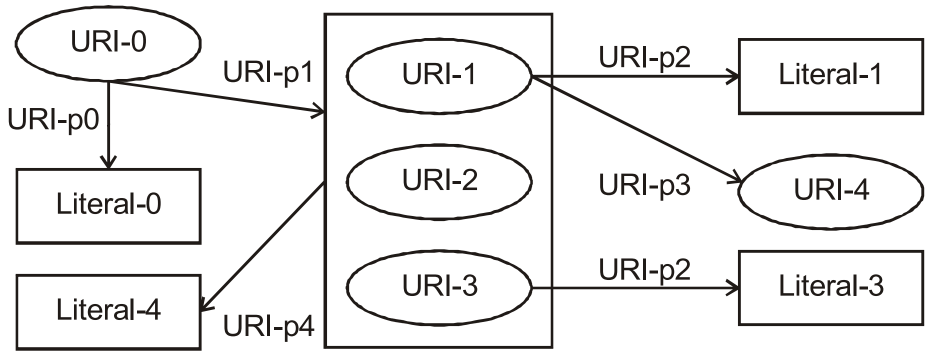
Complex values

- Each triple in RDF graph is known as a ‘property’.
- Nodes may have more than one arc originating from them, indicating that multiple propertyTypes are associated with the same resource.
- Groups of multiple properties are known as ‘descriptions’.
- PropertyTypes may point to simple atomic values (strings and numbers) or to more complex values that are themselves made up of collections of properties.
- syntactically, the values can be embedded (lexically in-line) or referenced (linked)

More complicated RDF graph



- informal model with a bag



- formal model with a bag and a blank node

blank node (*b-node*)

- is a node of RDF graph, which is not identified by a URI and is not a literal
- is a graph scoped identifier that cannot be directly referenced from outside
- is used mainly for graph branching as for representing higher arity relations
- it can be used only as a subject or an object of a RDF triple.
- it cannot be used as a predicate (in in some syntaxes like N3 it is acceptable to use a blank node as a predicate)
- Using *b-nodes* may cause problems in merging or querying
  - possible node ID conflicts (merging)
  - temporary node ID assignments (querying)

RDF graph labeling

- URI can be a subject, a predicate or an object of a triple
- only URI can be a predicate of a triple
- only URI and *bnode* can be a subject of a triple
- *bnode*, URI or literal can be an object of a triple

Project co-financed from the EU European Social Fund

Literals

- All literals are Unicode strings and represents value such as string or number
- Literals cannot be the subjects of statements, only the objects (target nodes in the RDF graphs).
- Literals can be either plain literals (without type) or typed literals
  - Plain literals can have an optional language tag assigned in form of a suffix starting with @ followed by the language code string (as defined by [RFC-3066], normalized to lowercase).
  - Typed literals have a lexical form ending with a suffix being RDF datatype URI reference (as in XML Schema Datatypes or an URI of custom datatype defined). The suffix starts with two caret characters ^^.

Typed literals

- Not all XML Schema datatypes are suitable for the use in RDF.
  - xsd:duration – does not have a well-defined value space;
  - xsd:QName and xsd:ENTITY – require an enclosing XML document context;
  - xsd:ID and xsd:IDREF – are for cross references within an XML document;
  - xsd:NOTATION – is not intended for direct use;
  - xsd:IDREFS, xsd:ENTITIES and xsd:NMTOKENS – are sequence-valued datatypes which do not fit the RDF datatype model.

RDF datatype

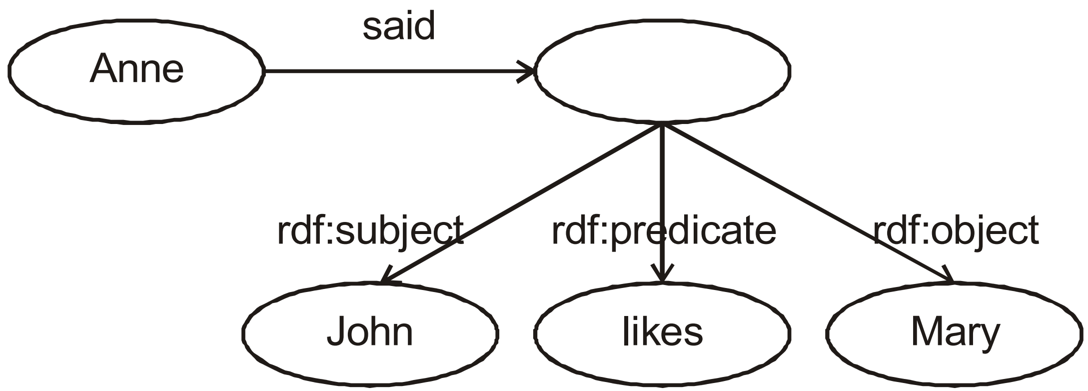
- There is no built-in concept of numbers or dates or other common values in RDF
- RDF predefines just one datatype: rdf:XMLLiteral, which is used for embedding XML in RDF
- There is no mechanism for defining new datatypes as well
  - It is expected that any new datatypes will be provided separately, and identified with URI references, as XML Schema datatypes defined in [XML-SCHEMA2].

Higher-order statements

- One can make RDF statements about other RDF statements
  - “Ralph believes that the web contains one billion documents”
  - “the creator of the statement that *the creator of the document on RDF syntax is Ora Lassila* is the Library of Congress”
- Higher-order statements
  - allow us to express beliefs (and other modalities)
  - are important for trust models, digital signatures,etc.
  - also: metadata about metadata
  - are represented by modeling RDF in RDF itself reification

Reification

- Allows to build higher-order statements, i.e. statements about other RDF statements
- Models of other statements must be created.
- New statements becomes new and accessible resources
- RDF built-in predicate vocabulary for reification:
  - rdf:subject
  - rdf:predicate
  - rdf:object
  - rdf:type



"Anne said that John likes Mary"

RDF serialization

- Serialization provides a way to convert between the abstract RDF model to a concrete format, such as a file or other byte stream. The most popular methods of RDF graphs serialization are:
  – RDF/XML, Terse RDF Triple Language (Turtle), and N-Triples.
- Serialization preserves the constructs of the original RDF graph.

RDF and RDF Schema (RDFS)

- A set of URIRefs is known as a vocabulary
- RDF vocabulary contains basic terms for expressing simple statements about resources, using named properties and values [RDF-CONCEPTS].
- RDFS vocabulary extends RDF vocabulary providing mechanisms for describing properties and relationships between these properties and other resources [RDF-VOCABULARY].
- RDF and RDFS vocabularies can describe relationships between items from multiple vocabularies developed independently, usually with the aid of XML namespace names.

OOP and RDF

- In object oriented-programming languages class definition:
  - implies the characteristic of the instances
  - is done in terms of the properties its instances may have
- In RDF properties definitions implies the class membership of the instance.
  - RDF describes properties in terms of the classes of resource to which they apply
  - If an instance has a certain property asserted with a domain defined, this domain specifies the class of this instance
- Naming convention
  - the class's names starts with an upper case letter
  - properties names starts with a lower case letter

RDF vocabulary

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

RDF property and type vocabulary	RDF reification vocabulary	RDF container vocabulary	RDF collection vocabulary
rdf:Property rdf:type rdf:XMLLiteral rdf:value	rdf:Statement rdf:subject rdf:predicate rdf:object	rdf:Seq rdf:Bag rdf:Alt rdf:_1 rdf:_2 ...	rdf:List rdf:first rdf:rest rdf:nil

HUMAN CAPITAL
HUMAN – BEST INVESTMENT!
Wrocław University of Technology
EUROPEAN SOCIAL FUND
Project co-financed from the EU European Social Fund

rdf:Property

- is a property (or, more formally, a class of RDF properties) used for the definition of predicates in triples.
- Each definition of a property might include restrictions regarding domain and range (using concepts from RDFS vocabulary). Even though properties are classes, they are defined and used independently of RDFS classes (defined with rdfs:Class from RDFS vocabulary).

rdf:type

- is a property (or, more formally, an instance of rdf:Property) used to assert a type to a resource.
- The value of this property is a URI identifying a class (or, more formally, an instance of rdfs:Class defined with RDFS vocabulary).
- A triple of the form: R rdf:type C states that C is an instance of rdfs:Class and R is an instance of C. Asserting the same type to several resources is possible, as asserting any other predicates.

rdf:XMLLiteral

- is a special built-in datatype delivered for assigning XML content as a possible literal value to a target nodes in the RDF graph.
- In the specification this datatype is described as an instance of rdfs:Datatype and a subclass of rdfs:Literal (using RDFS vocabulary).

rdf:value

- is an instance of rdf:Property that may be used in describing structured values.
- It delivers the actual value for the subject with several properties
  - distance can have rdf:value property with a value "15"^^xsd:decimal, and ex:unit property with a value, for example "meter"^^xsd:string
- rdf:value has no meaning on its own. It is provided as a piece of vocabulary that may be used in such idioms.
- The rdfs:domain and rdfs:range of rdf:value is rdfs:Resource (using RDFS vocabulary).

rdf:Statement

- is a resource reifying a triple
- it must have at least 3 properties valued by the corresponding resources:
  - rdf:subject
  - rdf:object
  - rdf:predicate

rdf:Alt, rdf:Bag, rdf:Seq

- Concepts used in the description of containers
  - rdf:Bag represents a container of unordered elements with duplicates allowed.
  - rdf:Alt is a container of alternative elements, possibly with a preference ordering, from which one is to be selected.
  - rdf:Seq is a container of ordered elements.
- They characterize the types of containers and provide the information on partial enumeration of their items rather then construct these containers.
- All they use the rdf:_n to establish the containment relationship with other resources.

rdf:_1, rdf_2, ...

- these are blank nodes of RDF graph, which are not absolutely identified by URIs.
- They represent anonymous resources at which RDF graph branches.
- They are properties, that associate a container as the subject with a resource it contains as the object.

rdf:List

- An instance of rdfs:Class that can be used to create collections known as list or list-like structures.
- Declaration of such collections is similar as in the programming languages, with a head and the tail and terminator declarations (for which the concepts of rdf:first, rdf:last, rdf:nil are used, respectively).

rdf:first

- this concept is used in the description of list and other list-like structures. It appears in the triples of the form L rdf:first O. The meaning of such triple is following: there is a first-element relationship between L and O. rfd:first is an instance of rfd:Property. The rdfs:domain of rdf:first is rdf:List and its rdfs:range is rdfs:Resource.

rdf:rest

- Concept used in the description of list and other list-like structures.
- It appears in the triples of the form L rdf:last O. The meaning of such triple is following:
  - there is a rest-of-list relationship between L and O.
- rfd:last is an instance of rfd:Property. The rdfs:domain of rdf:last is rdf:List and its rdfs:range is rdfs:List.

rdf:nil

- is an instance of rdf:List, representing the empty list or list-like structure.
- rdf:nil appears in the triples of the form: L rdf:rest rdf:nil
  - L is an instance of rdf:List that has one item, that can be indicated using rdf:first property.

RDSF

- Vocabulary for custom vocabularies creation
- Provides a type system for RDF
  - custom classes and properties definitions are possible
  - characteristics of other resources, such as domains (to indicate that a resource is of a particular RDF class) and ranges (to indicate that a resource is of a specific data type) of properties can be provided.
- Validation against RDF Schema is not the same as validation against XML Schema
  - there is no syntax check against schema
  - a graph consistency check is done on demand by reasoning engine
  - inconsistent triples may be added to a graph being not detected unless a consistency check is performed
  - traversing class inheritance in order to access and analyze their properties requires engine with reasoning capabilities

RDFS vocabulary

rdfs:domain	rdfs:Resource
rdfs:range	rdfs:Literal
rdfs:subClassOf	rdfs:Datatype
rdfs:subPropertyOf	rdfs:Class
rdfs:member	rdfs:Container
rdfs:comment	rdfs:ContainerMembershipProperty
rdfs:seeAlso	
rdfs:isDefinedBy	
rdfs:label	

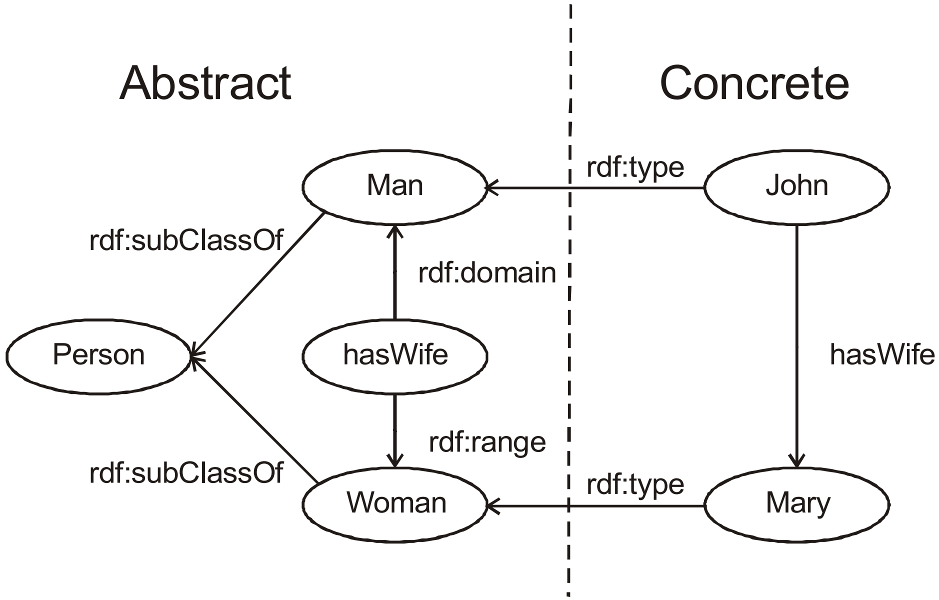
Project co-financed from the EU European Social Fund

Vocabularies for classes and properties

- vocabulary for classes:
  - **rdfs:Class** (a resource is a class)
  - **rdf:type** (a resource is an instance of a class)
  - **rdfs:subClassOf** (a resource is a subclass of another resource)
- vocabulary for properties:
  - **rdf:Property** (a resource is a property)
  - **rdfs:domain** (denotes the subject of a property)
  - **rdfs:range** (denotes the object of a property)
  - **rdfs:subPropertyOf** (expresses inheritance of properties)

Declarations templates

- P rdf:type C
  - C is an instance of rdfs:Class
  - P is an instance of C.
- P rdfs:domain C
  - P is an instance of the class rdf:Property
  - C is a instance of the class rdfs:Class
  - subjects of triples whose predicate is P are instances of the class C.
- P rdfs:range C
  - P is an instance of the class rdf:Property
  - C is an instance of the class rdfs:Class
  - objects of triples whose predicate is P are instances of the class C.



rdfs:domain

- an instance of rdf:Property used in resource definitions in similar manner to the type declaration
- Usage:
  - P rdfs:domain C.
- Meaning:
  - P is an instance of the class rdf:Property, and C is an instance of the class rdfs:Class, and the resources denoted by the subjects of triples whose predicate is P are instances of the class C.
- The property P can have more than one rdfs:domain property assigned
  - subjects of any triples with predicate P are instances of all the classes stated by the rdfs:domain properties (are instances of more then one class)
- The domain of rdfs:domain is rdf:Property class. The range of rdfs:domain is rdfs:Class class.

rdfs:range

- An instance of rdf:Property used for range of property values definition in similar manner to type declaration
- rdfs:range states that the values of a property are instances of one or more classes.
- Usage:
  - P rdfs:range C.
- Meaning:
  - P is an instance of the class rdf:Property, C is an instance of the class rdfs:Class and the resources denoted by the objects of triples whose predicate is P are instances of the class C.
- The property P can have more than one rdfs:range property assigned
  - the resources denoted by the objects of triples with predicate P are instances of all the classes stated by the rdfs:range properties.
- The domain of rdfs:range is rdf:Property class. The range of rdfs:range is the rdfs:Class.

rdfs:Resource

- A class, instances of which can be all things described by RDF as resources.
- It is the class of everything.
- All other classes are subclasses of this class.
- rdfs:Resource is an instance of rdfs:Class.

rdfs:Literal

- A class representing values such as numbers and dates used as the range of properties.
- Literals may be
  - plain ("October, 1, 2010"@en)
  - typed ("2010-01-10"^^xsd:date).
- String in literals are recommended to be in Unicode Normal Form C [http://www.w3.org/TR/2004/REC-rdf-concepts-20040210/#ref-nfc].
- Typed rdfs:Literal is an instance of rdfs:Class,
- rdfs:Literal is a subclass of rdfs:Resource

rdfs:Datatype

- A class used to assert that a literal should be interpreted in a particular way
- A datatype is defined abstractly by
  - two domains: one of lexical forms and one of values
  - and a mapping from lexical forms to values.
- A datatype is identified by one or more URI references
- Some external mechanism recognises a datatype URI, accessing and making use of appropriate representations of the domains and map.
- Each instance of rdfs:Datatype is a subclass of rdfs:Literal. rdfs:Datatype is both an instance of and a subclass of rdfs:Class

```
ex:octalnumber rdf:type rdfs:Datatype .
Judy ex:age _:y .
_:y ex:octalnumber "35" .
```

Value Space	{T, F}
Lexical Space	{"0", "1", "true", "false"}
Lexical-to-Value Mapping	{<"true", T>, <"1", T>, <"0", F>, <"false", F>}

rdfs:Class

- Classes are themselves resources.
- Once declared, the RDF class can be used as a value of rdf:type property.
  - The subject of the corresponding triple becomes implicitly an instance of the class.
- The members of a RDF class are instances of the class
  - the set of instances is the extension of the class, and two different classes may contain the same set of instances
  - class and a set of class' instances do not have to be the same
- rdfs:Class is an instance of rdfs:Class, and is the class of classes. The group of resources that are RDF Schema classes is itself a class called rdfs:Class.

rdfs:subClassOf

- A property used to form a taxonomy of classes by extending existing classes.
- It might be used to state that one class is a subclass of another.
  - Extension reuses (and thus shares) existing definition(s). A class can have multiple superclasses.
  - If a class C is a subclass of a class C', then all instances of C will also be instances of C'.

rdfs:subPropertyOf

- A property used to form a taxonomy of properties in a similar way as rdfs:subClassOf in a classes case.

rdfs:member

- A property that is a super-property of all the container membership properties, each container membershi property has an rdfs:subPropertyOf relationship to the property rdfs:member.

rdfs:Container

- A class used to represent the core RDF Container classes, ie. rdf:Bag, rdf:Seq, rdf:Alt.

rdfs:ContainerMembershipProperty

- A class, instances of which are properties: rdf:_1, rdf:_2, rdf:_3 ... stating, that a resource is a member of a container.
- rdfs:ContainerMembershipProperty is a subclass of rdf:Property.
- Each instance of rdfs:ContainerMembershipProperty is an rdfs:subPropertyOf the rdfs:member property.
- Container membership properties might be applied to resources other than containers.

HUMAN CAPITAL
HUMAN – BEST INVESTMENT!
Wrocław University of Technology
EUROPEAN SOCIAL FUND
Project co-financed from the EU European Social Fund

rdfs:comment

- An instance of rdf:Property that may be used to provide a human-readable description of a resource, clarifying its meaning. Multilingual documentation is supported through use of the language tagging facility of RDF literals.

rdfs:seeAlso

- An instance of rdf:Property that is used to indicate a resource that might provide additional information about the subject resource.

rdfs:isDefinedBy

- An instance of rdf:Property that is used to indicate a resource defining the subject resource.
- This property might be used to indicate an RDF vocabulary in which a resource is described. rdfs:isDefinedBy is a subproperty of rdfs:seeAlso.

rdfs:label

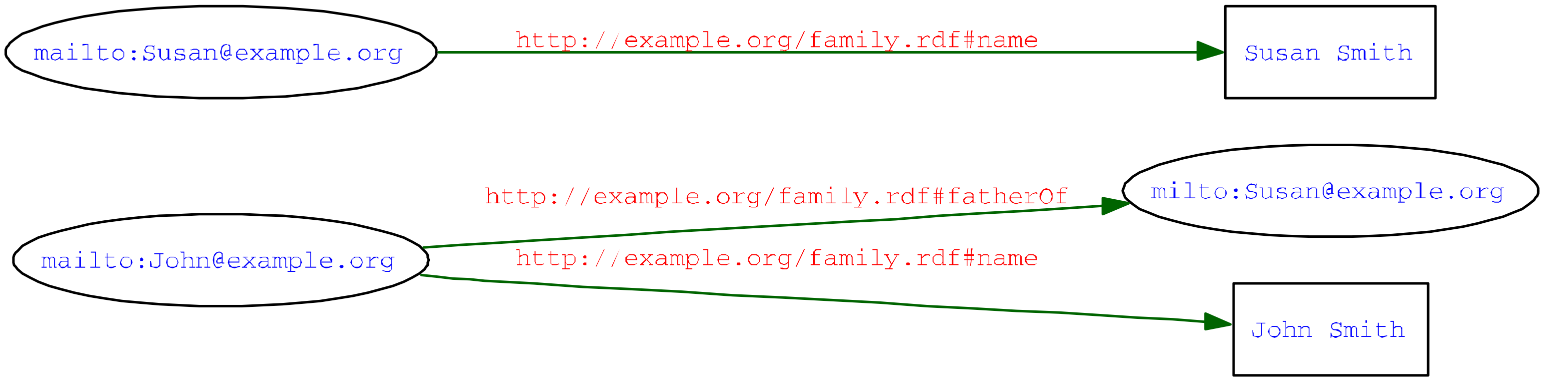
- An instance of rdf:Property that may be used to provide a human-readable version of a resource's name.
- Multilingual labels are supported using the language tagging facility of RDF literals.

Concepts details

Element	Class of	rdfs:subClassOf	rdf:type
rdfs:Resource	all resources	rdfs:Resource	rdfs:Class
rdfs:Class	all classes	rdfs:Resource	rdfs:Class
rdfs:Literal	literal values	rdfs:Resource	rdfs:Class
rdfs:Datatype	datatypes	rdfs:Class	rdfs:Class
rdf:XMLLiteral	XML literal values	rdfs:Literal	rdfs:Datatype
rdf:Property	properties	rdfs:Resource	rdfs:Class
rdf:Statement	statements	rdfs:Resource	rdfs:Class
rdf:List	lists	rdfs:Resource	rdfs:Class
rdfs:Container	containers	rdfs:Resource	rdfs:Class
rdf:Bag	unordered containers	rdfs:Container	rdfs:Class
rdf:Seq	ordered containers	rdfs:Container	rdfs:Class
rdf:Alt	containers of alternatives	rdfs:Container	rdfs:Class
rdfs:Container MembershipProperty	rdf:_1... properties expressing membership	rdf:Property	rdfs:Class

Roles and restrictions

Element	Role	rdfs:domain	rdfs:range
rdfs:range	restriction on subject	rdf:Property	rdfs:Class
rdfs:domain	restriction on object	rdf:Property	rdfs:Class
rdf:type	instance declaration	rdfs:Resource	rdfs:Class
rdfs:subClassOf	subclass declaration	rdfs:Class	rdfs:Class
rdfs:subPropertyOf	subproperty declaration	rdf:Property	rdf:Property
rdfs:label	human readable label	rdfs:Resource	rdfs:Literal
rdfs:comment	human readable comment	rdfs:Resource	rdfs:Literal
rdfs:member	container membership	rdfs:Resource	rdfs:Resource
rdf:first	first element declaration	rdf:List	rdfs:Resource
rdf:rest	rest of a list declaration	rdf:List	rdf:List
rdf:_1,rdf:_2, ...	container membership	rdfs:Container	rdfs:Resource
rdfs:seeAlso	additional information	rdfs:Resource	rdfs:Resource
rdfs:isDefinedBy	subject definition info	rdfs:Resource	rdfs:Resource
rdf:value	used for structured values	rdfs:Resource	rdfs:Resource
rdf:object	object declaration	rdf:Statement	rdfs:Resource
rdf:predicate	predicate declaration	rdf:Statement	rdfs:Resource
rdf:subject	subject declaration	rdf:Statement	rdfs:Resource



TURTLE serialization

```
@prefix ex: <http://example.org/family.rdf#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
# This is a comment.
<mailto:John@example.org>
ex:fatherOf <milto:Susan@example.org> ;
ex:name "John Smith"^^xsd:string .
<mailto:Susan@example.org> ex:name "Susan
Smith"^^xsd:string .
```

N-Triples serialization

```
<mailto:John@example.org>
<http://example.org/family.rdf#fatherOf>
<milto:Susan@example.org> .
<mailto:John@example.org>
<http://example.org/family.rdf#name> "John
Smith"^^<http://www.w3.org/2001/XMLSchema#string> .
<mailto:Susan@example.org>
<http://example.org/family.rdf#name> "Susan
Smith"^^<http://www.w3.org/2001/XMLSchema#string> .
```

RDF/XML serialization

```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
xmlns:ex="http://example.org/family.rdf#">
 <rdf:Description rdf:about="mailto:John@example.org">
   <ex:fatherOf rdf:resource="milto:Susan@example.org" />
   <ex:name
rdf:datatype="http://www.w3.org/2001/XMLSchema#string">John
Smith</ex:name>
 </rdf:Description>
 <rdf:Description rdf:about="mailto:Susan@example.org">
  <ex:name
rdf:datatype="http://www.w3.org/2001/XMLSchema#string">Susan
Smith</ex:name>
 </rdf:Description>
</rdf:RDF>
```

RDF/XML

- RDF/XML is a normative language for serializing RDF graph in a computer-readable form
- RDF/XML document starts with rdf:RDF element declaration having series of rdf:Description elements embedded.
- The list of attributes of rdf:RDF element contains XML namespace declarations.
- The definitions of elements used in RDF/XML serialization are given in RDF/XML document, which is available under the link http://www.w3.org/1999/02/22-rdf-syntax-ns. That document defines RDF itself (RDF Schema for the RDF vocabulary defined in the RDF namespace).

Statements in RDF/XML

- Are declared in the scope of <rdf:Description> elements.
  - <rdf:Description> element can group several statements with the same subject and different predicates and objects.
  - names of nested elements or names of element’s atrributes represent predicates
  - rdf:about attribute declares subject
- If an object of the statement is a resource, it is represented by the value of rdf:resource predicate’s attribute.
- If an object of the statement is a literal, it is represented by the predicate’s content (if the predicate is an element) or by the predicate’s value (if the predicate is an attribute).

Statement example

```
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
 <rdf:Description
rdf:about="http://example.org/myFamily.rdf#John">
   <rdf:type rdf:resource="http://example.org/terms#Father" />
 </rdf:Description>
</rdf:RDF>
```

rdf:type

- This predicate is used to assign types to nodes as presented
- A shorthand syntax for declaration of typed node elements:

  <*type* rdf:about="*resource*" />

  - *type* is URI of the value of the rdf:type predicate assigned to *resource*
  - in a case of multiple rdf:type predicates only one can be used in this way, the others must remain as property elements or property attributes
- Further abbreviation incorporates the use of xml:base attribute
  - this attribute is used to resolve relative RDF URI references
  - the base URI set applies to rdf:about, rdf:resource, rdf:ID and rdf:datatype

rdf:ID

- The rdf:ID attribute on a node element (not property element, that has another meaning) can be used instead of rdf:about
  - When used, it gives a relative RDF URI reference equivalent to # concatenated with the rdf:ID attribute value.
  - rdf:ID is useful for defining a set of distinct, related terms relative to the same RDF URI reference.
  - Such terms can not appear more than once in the scope of an xml:base value (or document, if none is given) what is automatically checked by the XML editing tools.

```
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xml:base="http://example.org/myFamily.rdf"
    xmlns:ex="http://example.org/terms#" >
  <ex:Father rdf:about="#John" />
  <ex:Father rdf:ID="John" />
 </rdf:RDF>
```

HUMAN CAPITAL
HUMAN – BEST INVESTMENT!
Wrocław University of Technology
EUROPEAN SOCIAL FUND
Project co-financed from the EU European Social Fund

Resource

- Resources can appear:
  - as elements' names (subjects or predicates)
    - the URI identifying a resource have to be abbreviated using standard XML namespace conventions (like ex:Father).
  - values of attributes (objects).
    - the URI can be abbreviated applying XML entity declarations (like rdf:about="&base;#John"). The example below shows both cases.

Resource example

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE rdf:RDF [
<!ENTITY base "http://example.org/myFamily.rdf">
]>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xml:base="&base;"
    xmlns:ex="http://example.org/terms#" >
 <ex:Father rdf:about="&base;#John" />
 <ex:Father rdf:ID="John" />
</rdf:RDF>
```

Literals

- Can appear in an every place where property value is expected: as attributes of elements or as contents of property nodes but with some restrictions.
  - The plain literals can appear as either property attributes or property nodes values, while the typed literals (XML or custom typed) only as property node contents.
- In the example below the literal "John Smith" appeared as a value of ex:name property attribute (declared in the namespace xmlns:ex="http://example.org/terms#").

```
<ex:Father rdf:about="#John" ex:name="John Smith"/>
```

- The same effect can be received by inserting this literal into the content of a property node <ex:name> as in the example below:

```
<rdf:Description
  rdf:about="http://example.org/myFamily.rdf#John">
    <rdf:type
  rdf:resource="http://example.org/terms#Father" />
  <ex:name>John Smith</ex:name>
 </rdf:Description>
```

Plain literals

- when declared as a property node content, can have an optional indicator of the content's language. This indicator is provided as a value of an optional rdf:lang attribute. In fact, this attribute can be used on any node element or property element to indicate that the included content is in the given language.
- The set of valid language indicators must be lowercased and match language tags as defined by RFC 4646

Typed literals

- Holds the content of the type declared using rdf:datatype attribute.
- The value of this attribute should be a datatype URI as defined in XML Schema or a custom datatype URI.
- In the example below "33" string will be interpreted as an integer number.

```
<ex:age
rdf:datatype="http://www.w3.org/2001/XMLSchema#int">33</ex:age>
```

XML literal

- For an XML literal, an attribute rdf:parseType should be used with a value set to "Literal" string. If so, the contents of the property node can be any XML document (as shown in the example below).

```
<rdf:Description rdf:about="http://example.org/myFamily.rdf#John's_car">
    <ex:record rdf:parseType="Literal" >
          <ex:color>red</ex:color>
          <ex:checkDates>
                    <ex:lastCheck>2009-02-03</ex:lastCheck>
                    <ex:nextCheck>2012-02-02</ex:nextCheck>
          </ex:checkDates>
          </ex:record>
 </rdf:Description>
</rdf:RDF>
```

Comments

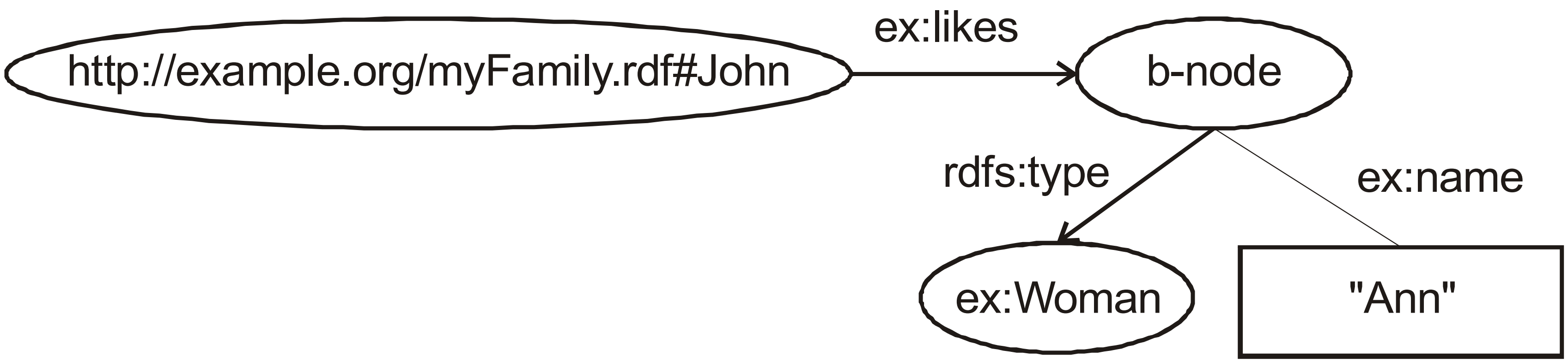
- As in any other XML document the comments can be provided within tags composed from characters <!-- and -->. But the comments are not a part of RDF graph and can disappear in serializing-deserialising round trip.

Project co-financed from the EU European Social Fund

Blank node

- Can be represented by a nested element.

```
<Description
rdf:about="http://example.org/myFamily.rdf#John">
 <ex:likes>
   <ex:Woman ex:name="Ann" />
 <ex:likes>
</Description>
```



rdf:nodeID

- An attribute rdf:nodeID allows multiple use of the same blank node
  - rdf:nodeID="b-node identifier" replaces rdf:about="RDF URI reference" when declaring blank node (the place for declaration of b-node with an identifier is Description element) or
  - replaces rdf:resource="RDF URI reference" when declaring property element (the place where reference to b-node identified is used is nested element).

```
<Description rdf:about="http://example.org/myFamily.rdf#John">
 <ex:likes rdf:nodeID="b1"/>
</Description>
<Description rdf:about="http://example.org/myFamily.rdf#Adam">
 <ex:likes rdf:nodeID="b1"/>
</Description>
<Description rdf:nodeID="b1">
        <ex:name>Meryl Streep</ex:name>
</Description>
```

Containers and collections

- Are declared as nested rdf:Bag, rdf:Seq, rdf:Alt elements – nodes of the RDF graph with a type property reflecting container's type.
- The rdf:about attribute can provide URIs identifyig these nodes. Without this attribute any given container becomes b-node.
- The elements nested in a container are rdf:_n or rdf:li . These elements are interpreted as properties of the container's node with values defined by rdf:resource attribute.

Containers example

```
<?xml version="1.0"?>
<rdf:RDF xmlns:ex="http://example.org/" xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
          <rdf:Description rdf:about="http://example.org/Tournament">
                    <ex:hasCompetitors>
                      <rdf:Bag rdf:about="http://example.org/Competitors">
                              <rdf:li rdf:resource="http://example.org/Adam" />
                              <rdf:li rdf:resource="http://example.org/Witold" />
                              <rdf:li rdf:resource="http://example.org/Tomasz" />
                      </rdf:Bag>
                    </ex:hasCompetitors>
                    <ex:hasStages>
                      <rdf:Seq rdf:about="http://example.org/Stages">
                              <rdf:li rdf:resource="http://example.org/Preliminary" />
                              <rdf:li rdf:resource="http://example.org/Group" />
                              <rdf:li rdf:resource="http://example.org/Final" />
                      </rdf:Seq>
                    </ex:hasStages>
                    <ex:hasPlace>
                      <rdf:Alt rdf:about="http://example.org/Playgrounds">
                              <rdf:li rdf:resource="http://example.net/Playground1" />
                              <rdf:li rdf:resource="http://example.net/Playground2" />
                      </rdf:Alt>
                    </ex:hasPlace>
          </rdf:Description>
</rdf:RDF>
```

Terse RDF Triple Language (Turtle)

- The simplest and most concise serialization syntax for RDF used in many textbooks and tutorials.
- Its human-friendly and readable syntax was designed specifically for RDF.
- Turtle is not an XML language, and therefore it has no support from XML editors.

Statements

- All parts of the statement, as subject, predicate, and object, should be written on a line, separated by white spaces and terminated with a period.
- The statements can be written in consecutive line if there are multiple statements about the same subject. This shorthand way relies on writing shared subject followed by a sequence of pairs composed from predicate and object of the statements, separated with a semicolon and terminated with a period.

```
exf:John rdf:type ext:Father .
exf:John ext:name "John Smith" .
An equivalent, shortened form of these declarations is as
follows:
exf:John rdf:type ext:Father ;
ext:name "John Smith" .
```

Further statement shortening

- Statements having the same both subject and predicate can be shortened.
  - the shared subject and predicate should be followed by the objects of statements, separated with a comma and terminated with a period

```
exf:John ext:likes "Meryl Streep" .
exf:John ext:likes "Another name".
```

- can be written as:

```
exf:John ext:likes "Meryl Streep",
 "Another name" .
```

Reification

- The reification of statements can be written shortly as in example:

```
[ a rdf:Statement;
rdf:subject exf:John;
rdf:predicate ext:likes;
rdf:object "Meryl Streep" ] .
```

Resources

- Are identified with URIs that are either
  - fully qualified identifiers or enclosed within sharp braces: < and >
  - identifiers build from a declared prefix and extension .
- The declaration of the prefix should be written on a line, starting with @prefix keyword, followed by the prefix name and a leading part of URIref enclosed within sharp braces. All these three parts should be separated by white spaces.

Typed resources

- The type of the resource can be declared with rdf:type predicate, written on line between the resource and the type URI.
- Shortened syntax it is similar to the use of rdf:type predicate, with a character a used instead of rdf:type.

```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix ext: <http://example.org/terms#> .
@prefix exf: <http://example.org/myFamily.rdf#> .
# the statement example comment
ex:John rdf:type ex:Father .
```

- The equivalent statement using a

```
ex:John a ex:Father .
```

- and using fully qualified URIs

```
<http://example.org/terms#John>
  <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
  <http://example.org/terms#Father> .
```

Literals

- Writing literals in Turtle depends on whether they are plain, data-typed or language tagged
- Literals are written as strings enclosed in double quotes (in a case of string without line break character) or as strings limited by the set of three double quotes on both sides (in a case of string containing line break).
- Since double quote is a special character, it appears in the literal-string values written as \" (U+0022). Similar escapes are used to encode surrounding syntax, non-printable characters and to encode Unicode characters by codepoint number (although they may also be given directly, encoded as UTF-8).

```
@prefix ex: <http://example.org/> .
ex:Book ex:hasMotto ""“First line of the  \"motto\",
and next line,
and final line.""" .
```

Datatyped and language tagged literals

- Datatyped literals are written with ^^ suffix, followed by any legal URI form giving the datatype URI.
- The language tagged literals are written with @ suffix followed by the valid character language tag. Literals might be given either a language suffix or a datatype URI but not both.

```
@prefix ex: <http://example.org/> .
ex:Bridge ex:numberOfCards
  "52"^^<http://www.w3.org/2001/XMLSchema#int> ;
ex:Bridge ex:name "Bridge"@en ;
ex:Bridge ex:name "Brydż"@pl.
```


Project co-financed from the EU European Social Fund

Comments

- Lines starting with # character
- The Turtle parser will ignore all text after this character to the end of the line.

Blank nodes

- b-node identifier starts with a prefix, which is colon, followed by a node ID
- It is possible to define a blank node without b-node identifier using of a pair of square brackets [and]
- All the statements written within these brackets have an unnamed b-node as the subject

```
ex:Furniture ex:hasDescription :Furniture-01 .
:Furniture-01
ex:name "chair"@en, "krzesło"@pl;
ex:color "brown"@en, "brązowy”@pl;
ex:productionDate "2010-12-01"^^^<http://www.w3.org/2001/XMLSchema#date> .

ex:Furniture ex:hasDescription [ex:name "chair"@en, "krzesło"@pl;
ex:color "brown"@en, "brązowy”@pl;
ex:productionDate "2010-12-01"^^^<http://www.w3.org/2001/XMLSchema#date>]
    .

@prefix : < http://example.org/myFamily.rdf > .
:#John a ex:Father .
: #John rdf:type ex:Father .
```

Containers and collections

- Starts with a line containing the subject, keyword a and one of the following terms: rdf:Bag, rdf:Seq and rdf:Alt. The consecutive line includes list of predicates of the form, rdf: 1, rdf: 2, rdf: 3, . . . , rdf n together with associated resources.
- All lines should end with a semicolon, except last, ended by dot. The numbers near by rdf: terms can be ignored in the declarations of rdf:Bag and rdf:Alt containers, but not in rdf:Sequence.

```
ex:Stages a rdf:Seq ;
rdf:_1 ex:Preliminary ;
rdf:_2 ex:Group ;
rdf:_3 ex:Final .
```

```
ex:Playgrounds a rdf:Alt ;
rdf:_1 <http://example.net/Playground1> ;
rdf:_2 <http://example.net/Playground2> .
```

```
ex:Stages a rdf:Seq ;
rdf:_1 ex:Preliminary ;
rdf:_2 ex:Group ;
rdf:_3 ex:Final .
```

```
ex:Tournament ex:hasCompetitors
ex:Competitors .
ex:Tournament ex:hasStages ex:Stages .
ex:Tournament ex:hasPlace ex:Playgrounds .
```

Ordering of elements in rdf:Sequence

- If the sequence is declared once, using predicates as rdf: n is straightforward. However, inserting any new elements into existing structure can cause some problems. If this did happen, several predicates would need to be re-enumerated.
- The solution is the use of rdf:li predicate. This predicates substitutes any of rdf: n predicates.
- When used, the order in which rdf:li predicates appear in the document is significant. The first resource of the group associated with rdf:li becomes rdf: 1, the second rdf: 2, and so on.
- Resources declared in such a way will not be altered even when different RDF graphs are merged.

Collections

- Are based on the concept of head-tail links. Starts with a subject, keyword a, followed by the rdf:List term, rdf:first (head) and rdf:rest (tail) predicates with identifiers.
- rdf:first usually refers to the object, rdf:rest to the b-node being a subjects of another collection declaration.
- The objects of rdf:first predicates are the members of collection being declared.
- The terminator of such recursive declaration is rdf:rest predicate, whose object is rdf:nil (tail referring to nil).

```
ex:Sponsors a rdf:List ;
rdf:first cmp:Company1 ;
rdf:rest :r1 .
:r1 a rdf:List ;
rdf:first cmp:Compan2 ;
rdf:rest :r2 .
:r2 a rdf:List ;
rdf:first cmp:Company3 ;
rdf:rest rdf:nil .
```

```
ex:Tournament ex:hasSponsors (cmp:Company1 cmp:Company2 cmp:Company3) .
```

```
ex:Tournament ex:hasSponsors ex:Sponsors .
```

N-Triples

- is based on the same syntax for comments, resources and literal values as in Turtle, but imposes some simplifying restrictions, as:
  - missing @prefix directive,
  - missing shorthand notion with semicolon or coma,
  - necessity of writing statements (triples) in a single line.

Managing RDF graphs

Creating

- any text editor
- graphical editor (IsaViz)
- Programmatically

Viewing

- RDF Gravity
- IsaViz
- dot
- Jambalaya
- W3C RDF Validator

Storing

- Sesame
- 3-Store
- JENA
- RDF-API for PHP

Storing RDF

- RDF graphs can be serialized as files (see example later) and stored in the file system
- RDF repositories provide
  – Query functionality
  – Access control
  – Distribution

Querying RDF

- Several query languages exist to retrieve resulting triples from RDF
  - RDQL
  - SERQL
  - SPARQL
- These languages use triple patterns as input and return matching triples as results

SPARQL

```
PREFIX
ex: <http://example.com/myOntology#>
SELECT ?capital ?country
WHERE { ?x abc:cityName ?capital.
?y ex:countryName ?country.
?x ex:isCapitalOf ?y.
?y ex:isInContinent ex:Europe. }
```

RDF tools

Tool	Commercial or Open-source	Environment
Anzo	Both	Java
ARC	Open-source	PHP
AllegroGraph	Commercial	Java, Prolog
Jena	Open-source	Java
Mulgara	Open-source	Java
Oracle RDF	Commercial	SQL / SPARQL
RDF::Query	Open-source	Perl
Redland	Open-source	C, many wrappers
Sesame	Open-source	Java
Talis Platform	Commercial	HTTP (Hosted)
Virtuoso	Both	C++