RDF-based Knowledge Graphs

Ernesto Jiménez-Ruiz

Lecturer in Artificial Intelligence

Useful information

- Recorded lectures and slides will be added to moodle.
- (Online) drop-in hours: **Wednesday from 1pm to 3pm**
  - Better to arrange meeting via e-mail.
- MSc project definition.

Graph Data Models

Data Graphs

- **Foundation** of any Knowledge Graph
- **Intuitive abstractions** to capture entities and their relationships.
- **Do not require schema** to start adding data.
- **Flexibility** to assemble data from multiple sources.

Data Graphs

– **Foundation** of any Knowledge Graph

– **Intuitive abstractions** to capture entities and their relationships.

– **Do not require schema** to start adding data.

– **Flexibility** to assemble data from multiple sources.

– Enable **queries over paths** of arbitrary length.

– Allow the use of **graph analytics** techniques.

– Correspondence with **logical** unary and binary **predicates**.

Types of graph-structured data models (i)

– Directed edge-labelled (multi)graphs (multi-relational graphs)

– Heterogeneous graphs (heterogeneous information network)

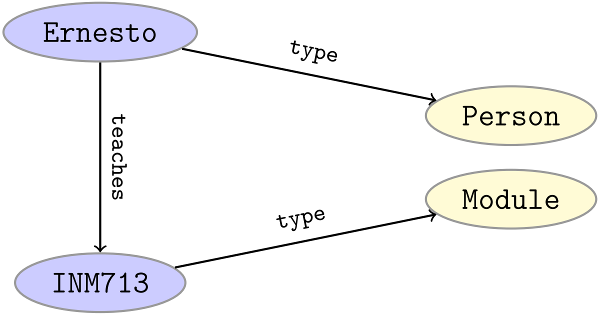
– Property graphs

Types of graph-structured data models (i)

- Directed edge-labelled (multi)graphs (multi-relational graphs)
- Heterogeneous graphs (heterogeneous information network)
- Property graphs
- Hypergraphs (edges connecting set of nodes)
- Hypernodes (nested graphs in a node)

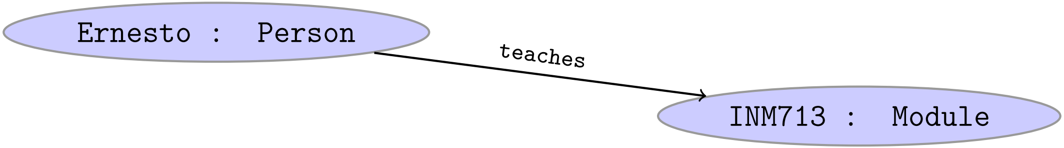
Types of graph-structured data models (ii)

Directed edge-labelled (multi)graphs



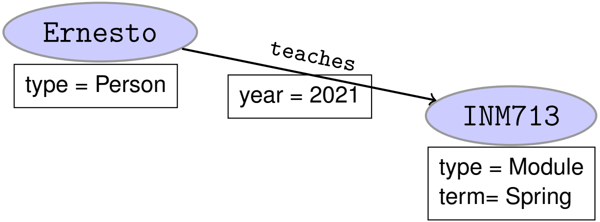
Types of graph-structured data models (iii)

Heterogeneous graphs



Types of graph-structured data models (iv)

Property graphs



How to store Graph models?

- Relational model
  - Single table with 3 columns (source, edge, target)
  - Property tables (one table per type of entity, *e.g.*, Person, Module)
  - Binary tables (one table per relationship, *e.g.*, source, target)

M. Wylot and others. RDF Data Storage and Query Processing Schemes. ACM Computing Surveys 2018

How to store Graph models?

- Relational model
  - Single table with 3 columns (source, edge, target)
  - Property tables (one table per type of entity, *e.g.*, Person, Module)
  - Binary tables (one table per relationship, *e.g.*, source, target)

- **Graph-based databases** (NoSQL)

M. Wylot and others. RDF Data Storage and Query Processing Schemes. ACM Computing Surveys 2018

NoSQL (Not only SQL) databases (i)

- **No strict definition** of NoSQL.
- They typically have a **denormalised** data model.
  - Data has some duplication.
  - Query performance is increased.
  - Writing and updates are less efficient.
- **No integrity constraints.**
- Consistency is checked afterwards ("eventually consistent" models).

NoSQL (Not only SQL) databases (ii)

Types of NoSQL databases:

- Key-Value pair-based databases (*e.g.*, ArangoDB)
- Column-oriented databases (*e.g.*, MonetDB)
- Document-oriented databases (*e.g.*, MongoDB, ArangoDB)

(*) Object-oriented databases can also be seen as a type of NoSQL but they include data integrity.

NoSQL (Not only SQL) databases (ii)

Types of NoSQL databases:

- Key-Value pair-based databases (*e.g.*, ArangoDB)
- Column-oriented databases (*e.g.*, MonetDB)
- Document-oriented databases (*e.g.*, MongoDB, ArangoDB)
- **Graph databases** (*e.g.*, Neo4j, Virtuoso, Amazon Neptune, Oracle)
  - **RDF Triplestores** (*e.g.*, RDFox, Apache Jena, RDFlib)
  - Most Graph databases are also suitable as RDF triplestores.

(*) Object-oriented databases can also be seen as a type of NoSQL but they include data integrity.

RDF: Resource Description Framework

RDF in a nutshell (i)

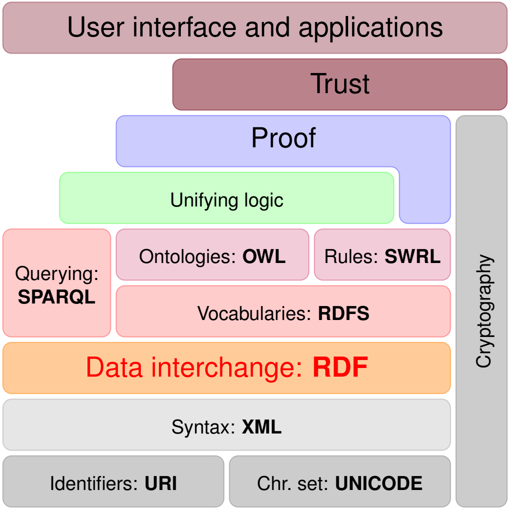
- Resource Description Framework (RDF)
- A standardised data model based on the **directed edge-labelled graph** model.
  - **Nodes**: Internationalised Resource Identifiers (IRIs), literals, and blank nodes (nodes without identifier).
  - **Edges**: IRIs
- **W3C recommendation**.
- Conceptual **modelling of resources**.

RDF in a nutshell (ii)

- RDF intended for annotation of Web-accessible resources (1999).
- Evolved into a general purpose language for describing structured information—**on the web or elsewhere**.
- The goal of RDF is to enable applications to **exchange data in a meaning-preserving way**.
- It is considered the **basic representation format** underlying the Semantic Web.
- In this module we will study **RDF-based Knowledge Graphs**

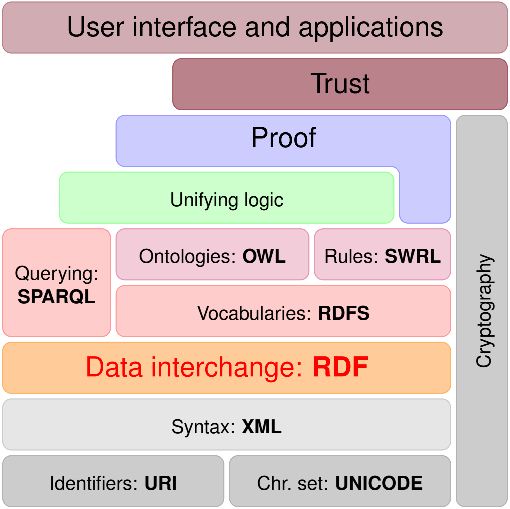
Semantic Web Stack

– Central block in the SW stack.



Semantic Web Stack

- Central block in the SW stack.
- In this module we will explore:
  - RDF
  - SPARQL
  - RDFS/OWL
  - A bit of logic and semantics
  - Applications



RDF data model

RDF Triples

– All information in RDF is expressed using a *triple* pattern.

RDF Triples

- All information in RDF is expressed using a *triple* pattern.
- A triple consists of a subject, a predicate, and an object.

RDF Triples

- All information in RDF is expressed using a *triple* pattern.
- A triple consists of a subject, a predicate, and an object.

RDF Triples

– All information in RDF is expressed using a *triple* pattern.

– A triple consists of a subject, a predicate, and an object.
Examples:

subject	predicate	object
England	has capital	London

RDF Triples

- All information in RDF is expressed using a *triple* pattern.
- A triple consists of a subject, a predicate, and an object.
  Examples:

subject	predicate	object
England	has capital	London
England	has queen	Elizabeth II

RDF Triples

- All information in RDF is expressed using a *triple* pattern.
- A triple consists of a subject, a predicate, and an object.
  Examples:

subject	predicate	object
England	has capital	London
England	has queen	Elizabeth II
Elizabeth II	born year	1926

RDF Triples

- All information in RDF is expressed using a *triple* pattern.
- A triple consists of a subject, a predicate, and an object.
  Examples:

subject	predicate	object
England	has capital	London
England	has queen	Elizabeth II
Elizabeth II	born year	1926

- Another word for an RDF triple is a *statement* or *fact*.

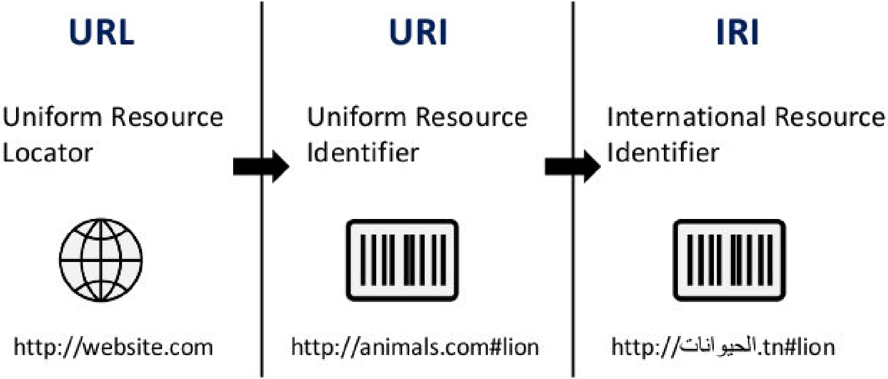
RDF Triples

- All information in RDF is expressed using a *triple* pattern.
- A triple consists of a subject, a predicate, and an object.
  Examples:

subject	predicate	object
England	has capital	London
England	has queen	Elizabeth II
Elizabeth II	born year	1926

- Another word for an RDF triple is a *statement* or *fact*.
- The elements of an RDF triple are either: *URI references*, *literals*, *blank nodes*.

Identifying resources



Identifying resources in RDF

- RDF talks about *resources*.
- Resources are identified by IRIs/URIs.
- E.g., in `dbpedia.org` KG graph:

Identifying resources in RDF

- RDF talks about *resources*.
- Resources are identified by IRIs/URIs.
- E.g., in `dbpedia.org` KG graph:

England:	`http://dbpedia.org/resource/England`
has capital:	`http://dbpedia.org/ontology/capital`
London:	`http://dbpedia.org/resource/London`
has queen:	`http://dbpedia.org/ontology/monarch`
Elizabeth II:	`http://dbpedia.org/resource/Elizabeth_II`

Identifying resources in RDF

- RDF talks about *resources*.
- Resources are identified by IRIs/URIs.
- E.g., in `dbpedia.org` KG graph:

England:	`http://dbpedia.org/resource/England`
has capital:	`http://dbpedia.org/ontology/capital`
London:	`http://dbpedia.org/resource/London`
has queen:	`http://dbpedia.org/ontology/monarch`
Elizabeth II:	`http://dbpedia.org/resource/Elizabeth_II`

- URIs/IRIs are not necessarily dereferenceable.

Identifying resources: URI $\not\subseteq$ URL

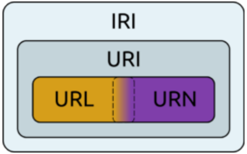
URLs are not the only URIs:

Identifying resources: URI $\not\subseteq$ URL

URLs are not the only URIs:

– ISBN:

`urn:isbn:978-1-4503-7615-0`



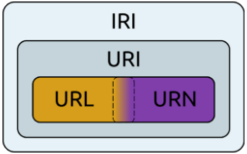
Identifying resources: URI $\not\subseteq$ URL

URLs are not the only URIs:

- ISBN:

  `urn:isbn:978-1-4503-7615-0`
- Geo:

  `geo:51.527264,-0.10247`



Identifying resources: URI $\not\subseteq$ URL

URLs are not the only URIs:

- ISBN:

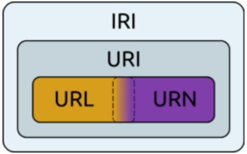
  `urn:isbn:978-1-4503-7615-0`

- Geo:

  `geo:51.527264,-0.10247`

- Mail:

  `mailto:ernesto.jimenez-ruiz@city.ac.uk`



Identifying resources: URI $\not\subseteq$ URL

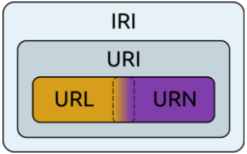
URLs are not the only URIs:

- ISBN:

  `urn:isbn:978-1-4503-7615-0`
- Geo:

  `geo:51.527264,-0.10247`
- Mail:

  `mailto:ernesto.jimenez-ruiz@city.ac.uk`
- and others …



Identifying resources: URIs and (local) CURIs

– **URIs are often long** and hard to read and write.

Identifying resources: URIs and (local) CURIs

- **URIs are often long** and hard to read and write.
- Most serialisations use an **abbreviation** mechanism.
  - Define “prefixes”, “namespaces”.
  - RDF/XML format: XML namespaces and entities.

Identifying resources: URIs and (local) CURIs

- **URIs are often long** and hard to read and write.
- Most serialisations use an **abbreviation** mechanism.
  - Define “prefixes”, “namespaces”.
  - RDF/XML format: XML namespaces and entities.
- E.g., in Turtle serialisation:

```
@prefix dbp:  <http://dbpedia.org/resource/> .
@prefix dbpo:  <http://dbpedia.org/ontology/> .
```

- A *CURI* (Compact URI) or *QName* like `dbp:London` stands for `http://dbpedia.org/resource/London`

URIs, QNames and data value

– We can then state that England's capital is London as:

```
<http://dbpedia.org/resource/England
<http://dbpedia.org/ontology/capital>
<http://dbpedia.org/resource/London> .
```

URIs, QNames and data value

– We can then state that England's capital is London as:

```
<http://dbpedia.org/resource/England
<http://dbpedia.org/ontology/capital>
<http://dbpedia.org/resource/London> .
```

– Or use prefixes:

```
dbp:England dbpo:capital dbp:London .
```

URIs, QNames and data value

– We can then state that England's capital is London as:

```
<http://dbpedia.org/resource/England
<http://dbpedia.org/ontology/capital>
<http://dbpedia.org/resource/London> .
```

– Or use prefixes:

```
dbp:England  dbpo:capital  dbp:London  .
```

– But what if we want to state that London's population is **9,304,000**?

– We cannot have one URI for every integer, decimal number, string, etc.

Literals in RDF (i)

- Literals are used to represent **data values**.
- A literal in a RDF consist of these elements:
  - A lexical form: “London”, “9,304,000”, “Londres”
  - A (supported) datatype IRI (from OWL, RDF or XML Schema datatypes)
  - A language tag (*e.g.*, “en”, “es”)

Literals in RDF (i)

- Literals are used to represent **data values**.
- A literal in a RDF consist of these elements:
  - A lexical form: “London”, “9,304,000”, “Londres”
  - A (supported) datatype IRI (from OWL, RDF or XML Schema datatypes)
  - A language tag (*e.g.*, “en”, “es”)
- Examples:

```
dbp:London dbpo:population "9,304,000"^^xsd:integer .
dbp:London rdfs:label "London"^^xsd:string .
dbp:London rdfs:label "Londres"@es .
```

Literals in RDF (ii)

– If the **literal is not typed** it is assumed to be a string:

```
dbp:London dbpo:officialName "London" .
```

Literals in RDF (ii)

- If the **literal is not typed** it is assumed to be a string:

```
dbp:London dbpo:officialName "London" .
```

- The **language defines metadata** about the string value, but **not correctness** of the language.

Literals in RDF (ii)

- If the **literal is not typed** it is assumed to be a string:

  `dbp:London dbpo:officialName "London" .`

- The **language defines metadata** about the string value, but **not correctness** of the language.

- **Ill-typed** literals:
  - Syntactically correct but semantically inconsistent.
  - The graph should still be constructed but with a warning.
  - *e.g.*, `dbp:London dbpo:areaUrbanKm2 "1737.9"^^xsd:integer .`

Literals in RDF (iii)

- Equality:
  - "1572.0"^^xsd:float and "1572"^^xsd:float
  - "01"^^xsd:integer and "1"^^xsd:integer
  - "Argentina"@es and "Argentina"@en
  - "1"^^xsd:integer and "1.0"^^xsd:float
  - The values are (semantically) the same, but the RDF literal is not (syntactically) the same.

RDF Graphs

– An *RDF graph* is a set of triples. E.g.,

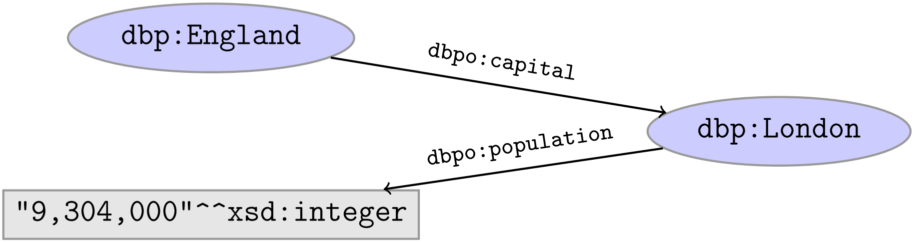
```
dbp:England  dbpo:capital     dbp:London .
dbp:London    dbpo:population  "9,304,000"^^xsd:integer .
```

RDF Graphs

– An *RDF graph* is a set of triples. E.g.,

```
dbp:England  dbpo:capital      dbp:London .
dbp:London    dbpo:population  "9,304,000"^^xsd:integer .
```

– RDF graphs are often represented as a directed labelled graph:

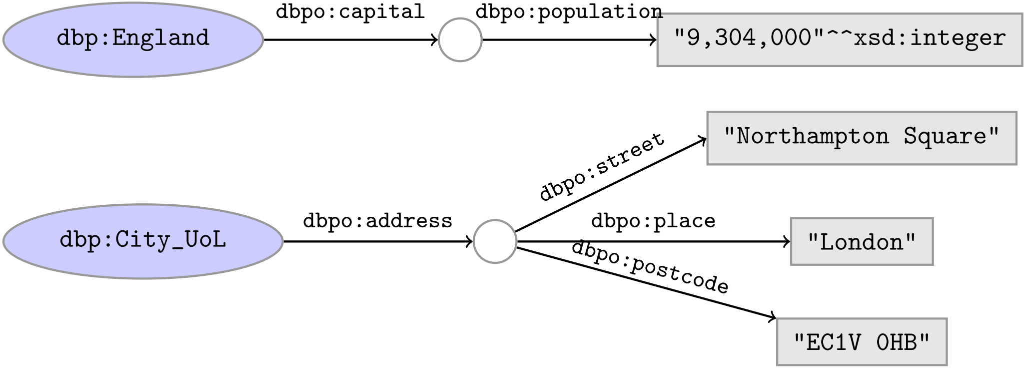


RDF Named Graphs

- We can **refer to a set of triples** with a name (*i.e.*, URI)
- Why?
  - **Independent** sources: 1 file, 1 graph.
  - Add specific **context** to a set of triples (*e.g.*, provenance).
  - Allowing to **query** a specific set of triples.
- Triples in a named graph sometimes referred as **quads**.

Blank nodes

- Blank nodes are like resources without a URI.
- Use when resource is unknown, or has no (natural) identifier. *e.g.*,:



RDF Triple Grammar

– Literals and blank nodes may not appear everywhere in triples:

RDF Triple Grammar

– Literals and blank nodes may not appear everywhere in triples:

	s	p	o
• URI references may occur in all positions	✔	✔	✔

RDF Triple Grammar

– Literals and blank nodes may not appear everywhere in triples:

	s	p	o
• URI references may occur in all positions	✔	✔	✔
• Literals may only occur in object position	✘	✘	✔

RDF Triple Grammar

– Literals and blank nodes may not appear everywhere in triples:

	s	p	o
• URI references may occur in all positions	✔	✔	✔
• Literals may only occur in object position	✘	✘	✔
• Blank nodes can not occur in predicate position	✔	✘	✔

RDF Triple Grammar

- Literals and blank nodes may not appear everywhere in triples:

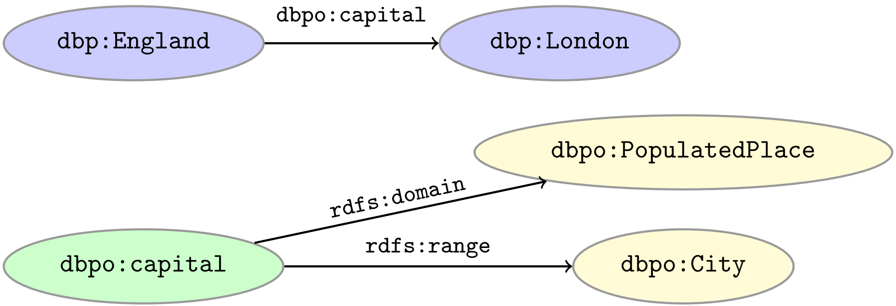
	s	p	o
• URI references may occur in all positions	✔	✔	✔
• Literals may only occur in object position	✘	✘	✔
• Blank nodes can not occur in predicate position	✔	✘	✔

- Why?
  - Literals are just values, no relationships from literals allowed.
  - Blank nodes in predicate position deemed “too meaningless” and confusing.

Is a RDF graph a graph?

Yes, although a resource can appear as both an edge and a node:

dbp:England	dbp:capital	dbp:London	.
dbp:capital	rdfs:domain	dbpo:PopulatedPlace	.
dbp:capital	rdfs:range	dbp:City	.



RDF Data Model Summary

Why URIs?

- URIs naturally have a **“global” scope**, unique throughout the web.
  - Contrasts to, e.g., keys in rel. DB which are unique within a table.
  - Helps to avoid name clashes.
  - Example: merging two product catalogues.

    `http://www.abc-company.com/category/item/123`

    `http://www.xyz-company.com/product/123`

Why URIs?

- URIs naturally have a **"global" scope**, unique throughout the web.
  - Contrasts to, e.g., keys in rel. DB which are unique within a table.
  - Helps to avoid name clashes.
  - Example: merging two product catalogues.

    `http://www.abc-company.com/category/item/123`

    `http://www.xyz-company.com/product/123`

- URLs are also **addresses**.
  - Exploit the well-functioning machinery of web browsing.
  - Find data by following data identifiers, i.e., URIs.
  - *"A web of data."*

Why triples?

- Simple unit of information
- Any information format can be transformed into triples.
  - Examples:

Tabular (spreadsheets, DBs):	row	column	cell
Trees (XML):	parent	path	child

Why triples?

– Simple unit of information

– Any information format can be transformed into triples.

  – Examples:

Tabular (spreadsheets, DBs):	row	column	cell
Trees (XML):	parent	path	child

– Relationships are made explicit and are first-class citizens:

  – The predicate is an element in the triple.
  – Can be described in RDF (*i.e.*, triples describing an predicate).

  dbp:capital rdfs:domain dbpo:PopulatedPlace .

Why graphs?

- A single, but highly **versatile**, format.
  - Everything is on the same format: triples!

Why graphs?

- A single, but highly **versatile**, format.
  - Everything is on the same format: triples!

- **Graph analytics** can be performed over the RDF graph.

Why graphs?

- A single, but highly **versatile**, format.
  - Everything is on the same format: triples!

- **Graph analytics** can be performed over the RDF graph.

- **Flexibility to merge** RDF graphs: Just take their union
  - With tabular data, table dimensions must match.
  - With trees, a node can only have one parent.

Why graphs?

- A single, but highly **versatile**, format.
  - Everything is on the same format: triples!

- **Graph analytics** can be performed over the RDF graph.

- **Flexibility to merge** RDF graphs: Just take their union
  - With tabular data, table dimensions must match.
  - With trees, a node can only have one parent.

- **Flexibility to extended** the RDF graph? Just add more triples!
  - Need not redefine the database table, or
  - to restructure the XML schema.

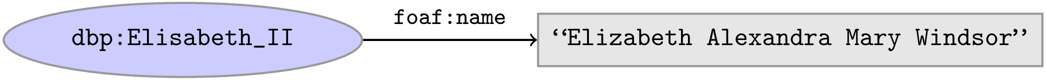
RDF serialisations

RDF Serialisations

There are many serialisations for the RDF data model:

- RDF/XML the W3C standard
- **Turtle**
- JSON-LD
- N3
- TriG

RDF/XML Serialisation



Machine readable:

```
<?xml version="1.0"?>
<rdf:RDF xmlns:dbp="http://dbpedia.org/resource/"
         xmlns:foaf="http://xmlns.com/foaf/0.1/"
         xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
  <rdf:Description rdf:about="http://dbpedia.org/resource/Elisabeth_II">
    <foaf:name>Elizabeth Alexandra Mary Windsor</foaf:name>
  </rdf:Description>
</rdf:RDF>
```

Turtle Serialisation

Human readable/writable

```
@prefix dbp: <http://dbpedia.org/resource/> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .

dbp:Elisabeth_II foaf:name "Elizabeth Alexandra Mary Windsor" .
```

Turtle: URI references and triples

Full URIs are surrounded by < and >:

```
<http://dbpedia.org/resource/London>
```

Turtle: URI references and triples

Full URIs are surrounded by < and >:

```
<http://dbpedia.org/resource/London>
```

Statements are triples terminated by a period '.':

```
<http://dbpedia.org/resource/London>
   <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
      <http://dbpedia.org/ontology/City> .
```

Turtle: URI references and triples

Full URIs are surrounded by < and >:

```
<http://dbpedia.org/resource/London>
```

Statements are triples terminated by a period '.':

```
<http://dbpedia.org/resource/London>
   <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
      <http://dbpedia.org/ontology/City> .
```

Use '`a`' to abbreviate `rdf:type`:

Turtle: Namespaces

QNames/CURIs are written without any special characters.

Turtle: Namespaces

QNames/CURIs are written without any special characters.

Namespace prefixes are declared with `@prefix`:

```
@prefix dbp: <http://dbpedia.org/resource/> .

dbp:London a <http://dbpedia.org/ontology/City> .
```

Turtle: Namespaces

QNames/CURIs are written without any special characters.

Namespace prefixes are declared with `@prefix`:

```
@prefix dbp: <http://dbpedia.org/resource/> .

dbp:London a <http://dbpedia.org/ontology/City> .
```

A default namespace may be declared:

```
@prefix dbp: <http://dbpedia.org/resource/> .
@prefix : <http://dbpedia.org/ontology/> .

dbp:London a :City .
```

Turtle: Literals

Literal values are enclosed in double quotes:

```
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix dbp: <http://dbpedia.org/resource/> .
@prefix : <http://dbpedia.org/ontology/> .

dbp:London rdfs:label "City of London" .
```

Turtle: Literals

Literal values are enclosed in double quotes:

```
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix dbp: <http://dbpedia.org/resource/> .
@prefix : <http://dbpedia.org/ontology/> .

dbp:London rdfs:label "City of London" .
```

Possibly with type or language information:

```
dbp:London rdfs:label "Londres"@es .
dbp:London :population "9,304,000"^^xsd:integer .
```

Turtle: Literals

Literal values are enclosed in double quotes:

```
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix dbp: <http://dbpedia.org/resource/> .
@prefix : <http://dbpedia.org/ontology/> .

dbp:London rdfs:label "City of London" .
```

Possibly with type or language information:

```
dbp:London rdfs:label "Londres"@es .
dbp:London :population "9,304,000"^^xsd:integer .
```

Numbers and booleans may be written without quotes:

```
dbp:London :population 9,304,000 .
dbp:London :isCapital true .
```

Turtle: Statements sharing elements (i)

Instead of:

```
dbp:London rdf:type dbo:City .
dbp:London rdfs:label "Oslo" .
dbp:London :population 9,304,000 .
```

Turtle: Statements sharing elements (i)

Instead of:

```
dbp:London rdf:type dbo:City .
dbp:London rdfs:label "Oslo" .
dbp:London :population 9,304,000 .
```

. . . statements may share a subject with '`;`':

```
dbp:London rdf:type dbo:City ;
         rdfs:label "Oslo" ;
         :population 9,304,000 .
```

Turtle: Statements sharing elements (ii)

Instead of:

```
dbp:London rdfs:label "London"@en .
dbp:London rdfs:label "Londres"@es .
dbp:London rdfs:label "Londra"@it .
```

Turtle: Statements sharing elements (ii)

Instead of:

```
dbp:London rdfs:label "London"@en .
dbp:London rdfs:label "Londres"@es .
dbp:London rdfs:label "Londra"@it .
```

. . . statements may share subject and predicate with ‘,’:

```
dbp:London rdfs:label "London"@en ,
                      "Londres"@es ,
                      "Londra"@it .
```

Turtle: Blank nodes

Blank nodes are designated with underscores or `[...]`.

England has a capital with population 9,304,000:

```
dbp:England :capital _:someplace .
_:someplace :population 9,304,000 .
```

Turtle: Blank nodes

Blank nodes are designated with underscores or `[...]`.

England has a capital with population 9,304,000:

```
dbp:England :capital _:someplace .
_:someplace :population 9,304,000 .
```

There is a city with official name London:

```
[] a :City ;
   :officialName "London" .
```

Turtle: Blank nodes

Blank nodes are designated with underscores or `[...]`.

England has a capital with population 9,304,000:

```
dbp:England :capital _:someplace .
_:someplace :population 9,304,000 .
```

There is a city with official name London:

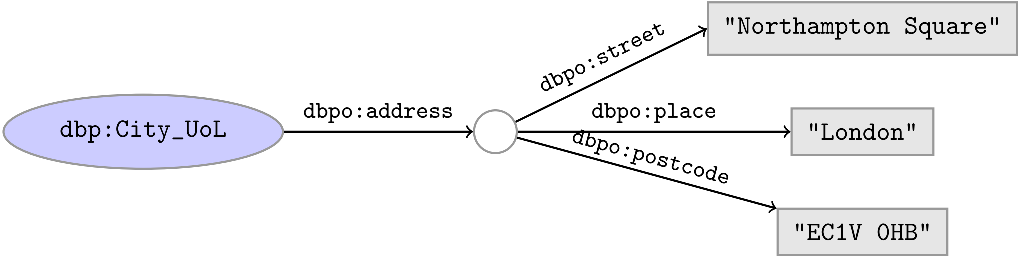
```
[] a :City ;
   :officialName "London" .
```

City has address Northampton Square, EC1V 0HB:

```
:City_UoL :address [ :street "Northampton Square" ;
                  :postcode "EC1V 0HB" ] .
```

Turtle: Blank nodes (Question)

The blank node here:

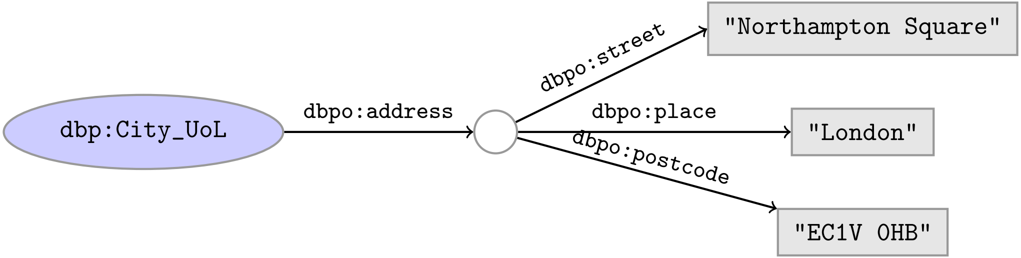


has no ‘name.’

Why does Turtle use ‘blank node identifiers’ like `_:someplace`?

Turtle: Blank nodes (Question)

The blank node here:



has no ‘name.’

Why does Turtle use ‘blank node identifiers’ like `_:someplace`?

Answer: makes it possible to use same node in several triples.

Turtle: Merging RDF files (i)

Merging two RDF files containing named blank nodes

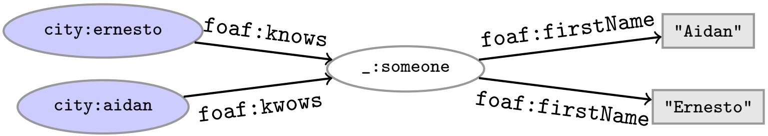
File 1

```
city:ernesto foaf:knows _:someone .
_:someone foaf:firstName  "Aidan" .
```

File 2

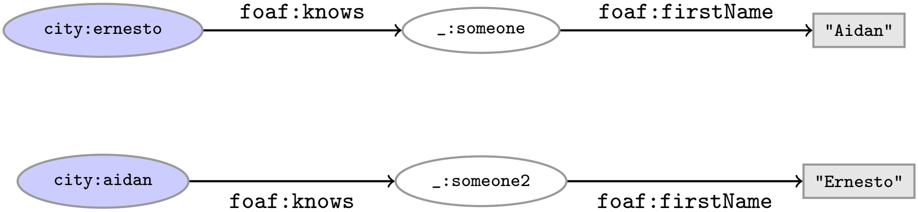
```
city:aidan foaf:knows _:someone .
_:someone foaf:firstName  "Ernesto" .
```

gives the RDF graph:



Turtle: Merging RDF files (i)

Solution: Renaming `_:someone` to `_:someone2` in File 2.



Turtle: complex statements

We can use triples to form complex statements, e.g.:

Turtle: complex statements

We can use triples to form complex statements, e.g.:

Data structures

```
:INM373 :hasLecturers
            [ rdf:first :epriego ;
              rdf:rest [ rdf:first :ernesto ;
                         rdf:rest [ rdf:first :simone ;
                                    rdf:rest [ rdf:first :andrey ;
                                               rdf:rest [ rdf:first :alex ;
                                                          rdf:rest rdf:nil ]
                                    ] ] ] ] .
```

Turtle: complex statements

We can use triples to form complex statements, e.g.:

Data structures

```
:INM373 :hasLecturers
            [ rdf:first :epriego ;
              rdf:rest [ rdf:first :ernesto ;
                         rdf:rest [ rdf:first :simone ;
                                    rdf:rest [ rdf:first :andrey ;
                                               rdf:rest [ rdf:first :alex ;
                                                          rdf:rest rdf:nil ]
                                             ] ] ] ] .
```

Turtle shorthand for lists

```
:INM373 :hasLecturers (:epriego :ernesto :simone :andrey :alex ) .
```

Turtle: TriG to support named graphs

– TriG RDF syntax extends Turtle with support for named graphs.

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

:G1 {
    :INM373 rdf:type :Module .
    :INM373 :hasLecturers ( :epriego :ernesto :simone :andrey :alex ) .
    :INM373 :module_type "Core" }
:G2 {
   :INM713 rdf:type :Module .
   :INM713 :hasLecturers ( :ernesto ) .
   :INM713 :module_type "Elective" }
```

Turtle: Other things

Use '`#`' to comment:

```
# This is a comment.
dbp:London a dbpo:City . # This is another comment.
```

Turtle: Other things

Use '#' to comment:

```
# This is a comment.
dbp:London a dbpo:City . # This is another comment.
```

Use '\' to escape special characters:

```
:someGuy :foaf:name "James \"Mr. Man\" Olson" .
```

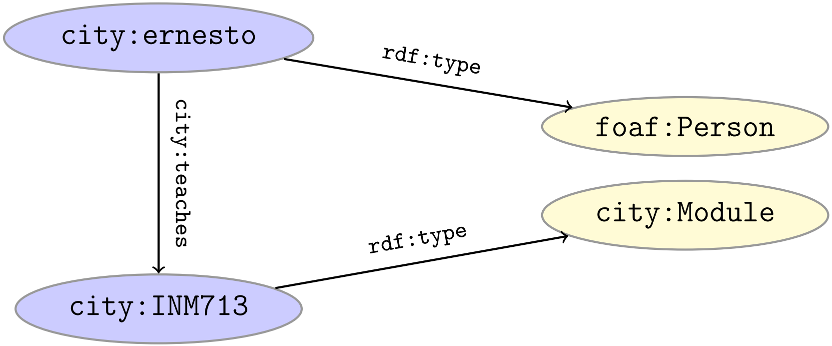
Turtle specification: `http://www.w3.org/TR/turtle/`.

RDF and Property Graphs

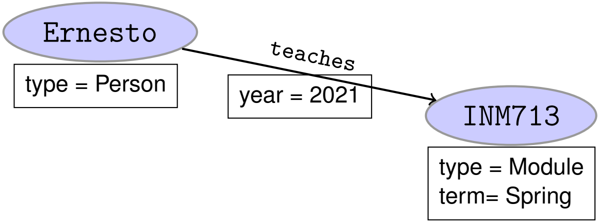
Annotating statements/triples in RDF (i)

How to encode that **"Ernesto teaches INM713 in 2021"**?

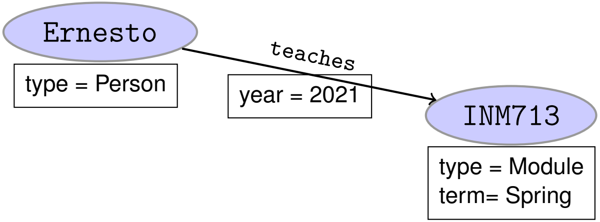
(*i.e.*, **Metadata of a triple**: provenance, beliefs, uncertainty, creator, time, etc.)



Using property Graphs



Using property Graphs



But can we do it in RDF?

Annotating statements/triples in RDF (ii)

RDF Solution 1: **Named Graphs**

```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix dbpo: <http://dbpedia.org/ontology/>
@prefix : <http://www.example.org/university/london/city#> .

:AcademicYear2021 {
   :ernesto :teaches :INM713 .
   :INM713 rdf:type :Module .
   :INM713 :module_type "Elective"
}
:AcademicYear2021 dbpo:year "2021"^^xsd:gYear .
```

Annotating statements/triples in RDF (iii)

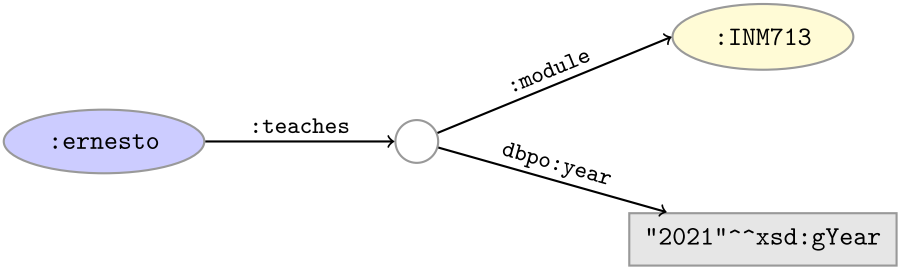
RDF Solution 2: **Singleton Property**

```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix dbpo: <http://dbpedia.org/ontology/>
@prefix : <http://www.example.org/university/london/city#> .

:teaches#2021 rdf:singletonPropertyOf :teaches .
:teaches#2021 dbpo:year "2021"^^xsd:gYear .
:ernesto :teaches#2021 :INM713 .
```

Annotating statements/triples in RDF (iv)

RDF Solution 3: **N-Ary relationships**



Annotating statements/triples in RDF (v)

RDF Solution 4: **Reification**

```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix dbpo: <http://dbpedia.org/ontology/>
@prefix : <http://www.example.org/university/london/city#> .

:ernesto :teaches :INM713 .
_:s rdf:type rdf:Statement
    rdf:subject :ernesto;
    rdf:predicate :teaches ;
    rdf:object :INM713 ;
    dbpo:year "2021"^^xsd:gYear .
:aidan :likes _:s .
```

Annotating statements/triples in RDF (vi)

RDF Solution 5: **RDF*** (Not yet an W3C recommendation)

- Uses embedding triple operator « »
- Compact solution easier to read than reification
- Closer to Property Graphs

```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix dbpo: <http://dbpedia.org/ontology/>
@prefix : <http://www.example.org/university/london/city#> .

:ernesto :teaches :INM713 .
<<:ernesto :teaches :INM713>> dbpo:year "2021"^^xsd:gYear .
```

Olaf Hartig: RDF* and SPARQL*: An Alternative Approach to Annotate Statements in RDF. ISWC Poster 2017
O. Hartig and B. Thompson. Foundations of an Alternative Approach to Reification in RDF. CoRR, abs/1406.3399, 2019

RDF vocabularies

Vocabularies

- Families of related notions are grouped into *vocabularies*.
- Some important, well-known namespaces—and prefixes:

  Modelling vocabulary:

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

  `rdfs: <http://www.w3.org/2000/01/rdf-schema#>` – RDF Schema

  `owl: <http://www.w3.org/2002/07/owl#>` – OWL

  Support vocabularies:

  `foaf: <http://xmlns.com/foaf/0.1/>` – Friend of a friend

  `dcterms: <http://purl.org/dc/terms/>` – Dublin Core

  `bfo: <http://purl.obolibrary.org/obo/bfo.owl#>` – Basic Formal Ontology

Vocabularies

- Families of related notions are grouped into *vocabularies*.
- Some important, well-known namespaces—and prefixes:

  Modelling vocabulary:

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

  `rdfs:  <http://www.w3.org/2000/01/rdf-schema#>` – RDF Schema

  `owl:  <http://www.w3.org/2002/07/owl#>` – OWL

  Support vocabularies:

  `foaf:  <http://xmlns.com/foaf/0.1/>` – Friend of a friend

  `dcterms:  <http://purl.org/dc/terms/>` – Dublin Core

  `bfo:  <http://purl.obolibrary.org/obo/bfo.owl#>` – Basic Formal Ontology

- Usually, a description is published at the namespace base URI.
- Note that the prefix is not standardised.

Example vocabularies: RDF, RDFS

RDF: describing RDF graphs.

- `rdf:Statement`
- `rdf:subject`,
  `rdf:predicate`,
  `rdf:object`
- `rdf:type`

Example vocabularies: RDF, RDFS

RDF: describing RDF graphs.

- `rdf:Statement`
- `rdf:subject`,
  `rdf:predicate`,
  `rdf:object`
- `rdf:type`

RDFS: describing RDF vocabularies.

- `rdfs:Class`
- `rdfs:subClassOf`,
  `rdfs:subPropertyOf`
- `rdfs:domain`,
  `rdfs:range`
- `rdfs:label`

Example vocabularies: RDF, RDFS

RDF: describing RDF graphs.

- `rdf:Statement`
- `rdf:subject`,
  `rdf:predicate`,
  `rdf:object`
- `rdf:type`

RDFS: describing RDF vocabularies.

- `rdfs:Class`
- `rdfs:subClassOf`,
  `rdfs:subPropertyOf`
- `rdfs:domain`,
  `rdfs:range`
- `rdfs:label`

Examples:

```
dbp:London rdf:type dbpo:City .
dbp:London rdfs:label "London"@en .
dbpo:City rdfs:subClassOf dbpo:Place .
```

Example vocabularies: OWL

OWL: describing ontologies

- `owl:inverseOf`
- `owl:equivalentClass`
- `owl:disjointWith`
- `owl:sameAs`

Example vocabularies: OWL

OWL: describing ontologies

- `owl:inverseOf`
- `owl:equivalentClass`
- `owl:disjointWith`
- `owl:sameAs`

Examples:

```
dbp:London owl:sameAs ex:London .
dbpo:location owl:inverseOf dbpo:isLocatedIn .
dbpo:City owl:disjointWith dbpo:Person .
dbpo:City owl:equivalentClass ex:City .
```

Example vocabularies: FOAF, Dublin Core

FOAF: person data and relations.

- `foaf:Person`
- `foaf:knows`
- `foaf:firstName`,
  `foaf:lastName`,
  `foaf:gender`

Example vocabularies: FOAF, Dublin Core

FOAF: person data and relations.

- `foaf:Person`
- `foaf:knows`
- `foaf:firstName`,
  `foaf:lastName`,
  `foaf:gender`

Dublin Core: library metadata.

- `dcterms:creator`,
  `dcterms:contributor`
- `dcterms:format`,
  `dcterms:language`,
  `dcterms:licence`

Example vocabularies: FOAF, Dublin Core

FOAF: person data and relations.

- `foaf:Person`
- `foaf:knows`
- `foaf:firstName`, `foaf:lastName`, `foaf:gender`

Dublin Core: library metadata.

- `dcterms:creator`, `dcterms:contributor`
- `dcterms:format`, `dcterms:language`, `dcterms:licence`

Examples:

```
city:ernesto rdf:type foaf:Person .
city:ernesto foaf:knows city:aidan .
city:INM713 dcterms:creator city:ernesto .
```

Example vocabularies: BFO

- Basic Formal Ontology:
  `http://www.obofoundry.org/ontology/bfo.html`
- It is an “upper level ontology”
- Lays the foundations of many ontologies in the biological domain.
- *e.g.*, `http://bioportal.bioontology.org/`

RDF on the Web

Where is it? (i)

- Pages driven by semantic data. *e.g.*,:
  - BBC: `https://www.bbc.co.uk/ontologies`
  - gov.uk: `https://docs.publishing.service.gov.uk/manual/knowledge-graph.html`
- In some serialisation: XML/RDF, Turtle, ...
  - Vocabularies: `http://xmlns.com/foaf/spec/index.rdf`
  - Example datasets: `https://lod-cloud.net/datasets`
  - FOAF descriptions: `https://sws.ifi.uio.no/vocab/ernesto_foaf.rdf`

Where is it? (ii)

- From *SPARQL endpoints*:
  - Data kept in a *triple store*
  - Exposes data (in different formats)
    - with endpoint frontends, e.g., `http://dbpedia.org/resource/London`, or
    - by direct SPARQL query: *e.g.*, `http://dbpedia.org/sparql` `https://www.ebi.ac.uk/rdf/datasets/`

Examples of Knowledge Graphs

- **Open knowledge graphs**
  - DBpedia, Freebase, Wikidata, YAGO
  - Domains: media, government, geography, tourism, life sciences, ecotoxicology, etc.

Aidan Hogan and others. Knowledge Graphs. CoRR abs/2003.02320 (2020)

Examples of Knowledge Graphs

- **Open knowledge graphs**
  - DBpedia, Freebase, Wikidata, YAGO
  - Domains: media, government, geography, tourism, life sciences, ecotoxicology, etc.
- **Enterprise knowledge graphs**
  - Websearch (*e.g.*, Bing, Google),
  - Commerce (*e.g.*, Airbnb, Amazon, eBay, Uber),
  - Social networks (*e.g.*, Facebook, LinkedIn),
  - Finance (*e.g.*, Accenture, Banca d'Italia, Bloomberg, Capital One)

Aidan Hogan and others. Knowledge Graphs. CoRR abs/2003.02320 (2020)

Acknowledgements

- Prof. Martin Giese and others (University of Oslo)
- INF4580 – Semantic technologies
- https://www.uio.no/studier/emner/matnat/ifi/INF4580/

Laboratory Session

Laboratory Session

- Break-out room for questions regarding code.
- General questions in chat or main room.
- Creation of RDF-based Knowledge Graphs.