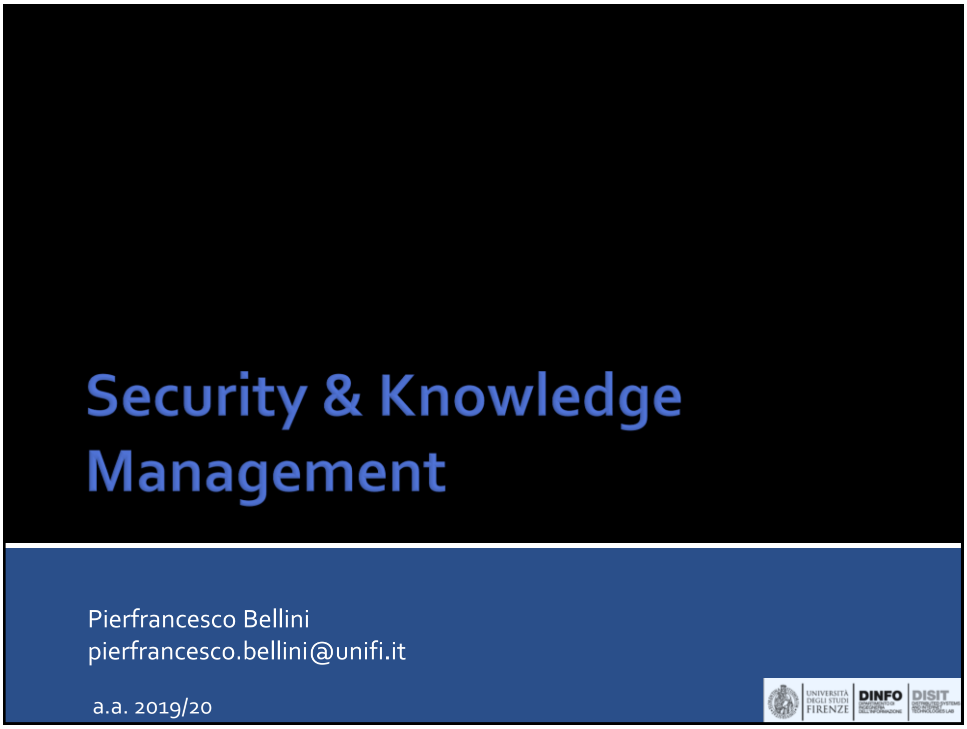
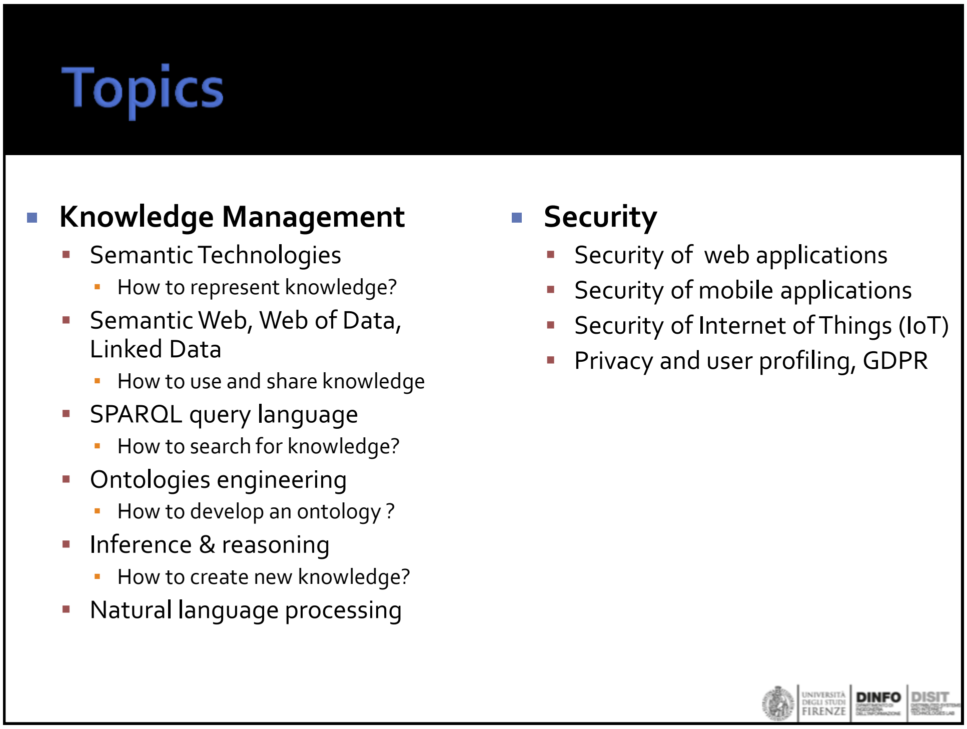


Security & Knowledge Management

Pierfrancesco Bellini
pierfrancesco.bellini@unifi.it

a.a. 2019/20



Topics

- **Knowledge Management**
  - Semantic Technologies
    - How to represent knowledge?
  - Semantic Web, Web of Data, Linked Data
    - How to use and share knowledge
  - SPARQL query language
    - How to search for knowledge?
  - Ontologies engineering
    - How to develop an ontology ?
  - Inference & reasoning
    - How to create new knowledge?
  - Natural language processing
- **Security**
  - Security of web applications
  - Security of mobile applications
  - Security of Internet of Things (IoT)
  - Privacy and user profiling, GDPR

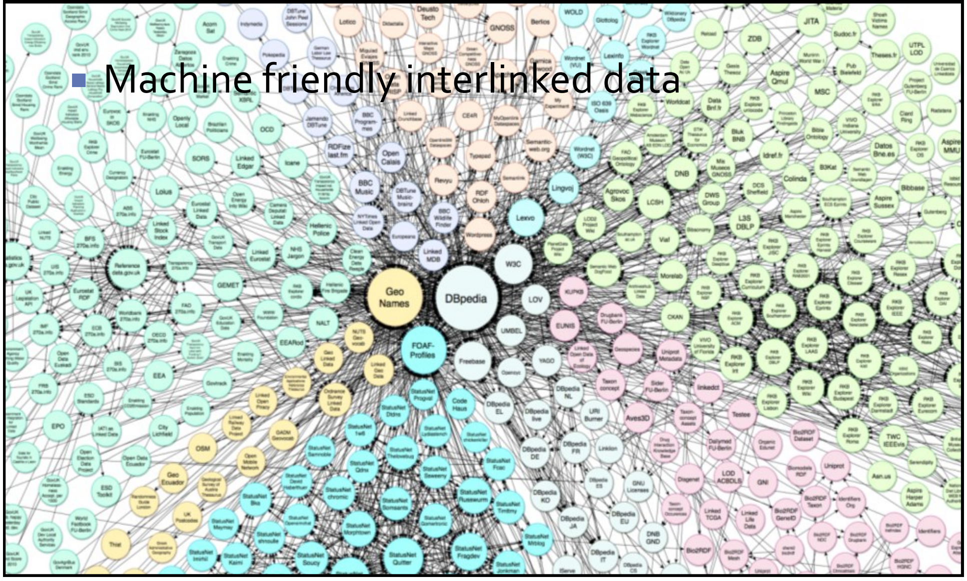
Dettagli

- **Modalità di esame**
  - Un progetto su argomenti del corso in gruppi di 1 o 2 persone
    - Es. sviluppo di una ontologia, valutazione di RDF store, …
- **Sito del corso**
  - https://www.disit.org/sekm
- **Corso**
  - mar. 16:00 – 19:00
  - ven. 16:00 – 19:00
- **Ricevimento**
  - dopo e nell'intervallo delle lezioni
  - a S. Marta su appuntamento

The Web

- Web of (HTML) documents
- A lot of **human readable** knowledge (e.g wikipedia)
- Not easily usable by machines

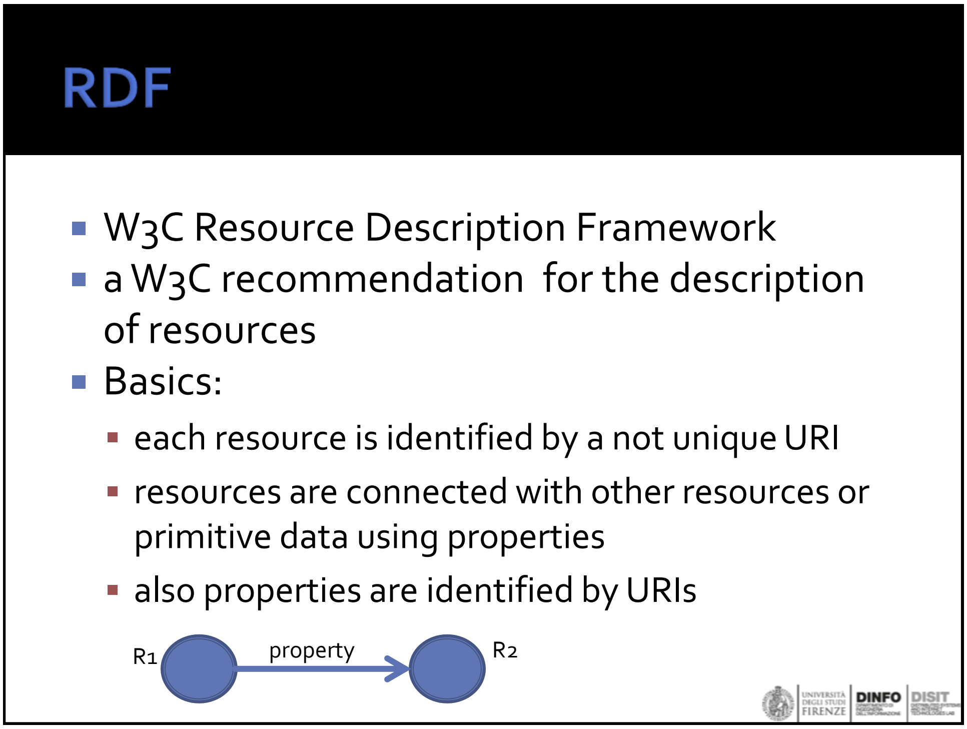
Web of Data / Semantic Web



- Machine friendly interlinked data

Web of Data / Semantic Web

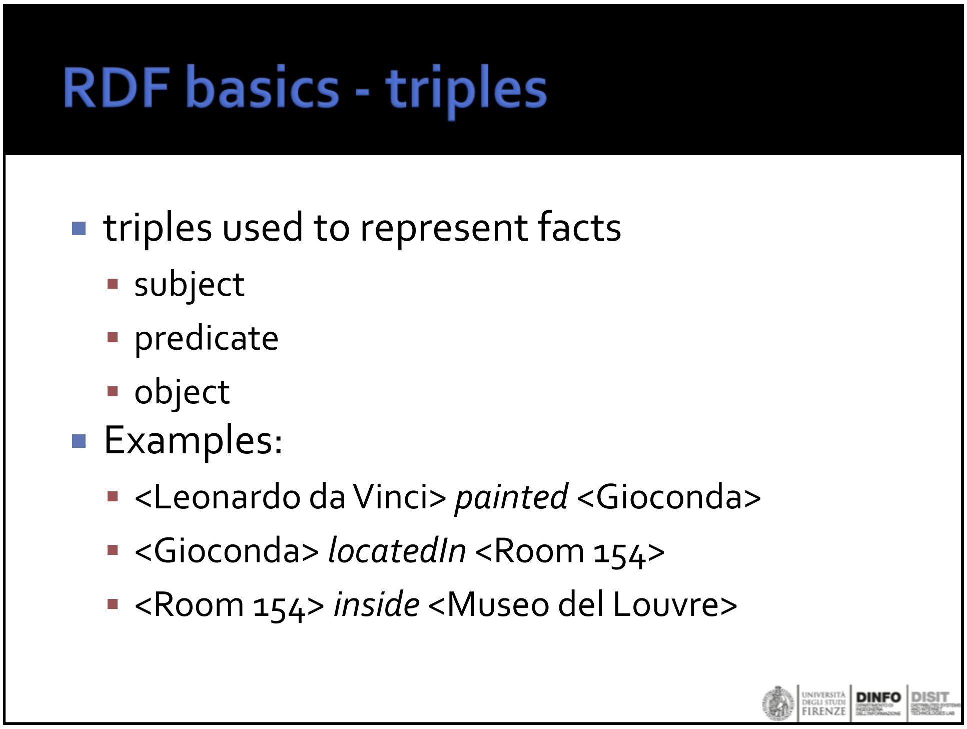
- Machine friendly
- Linked Data
  - easily accessible using the web protocols (HTTP)
  - resources identified using URLs
  - machine friendly description of the Resources
    - (W3C RDF – Resource Description Framework)
  - links among different data providers



RDF

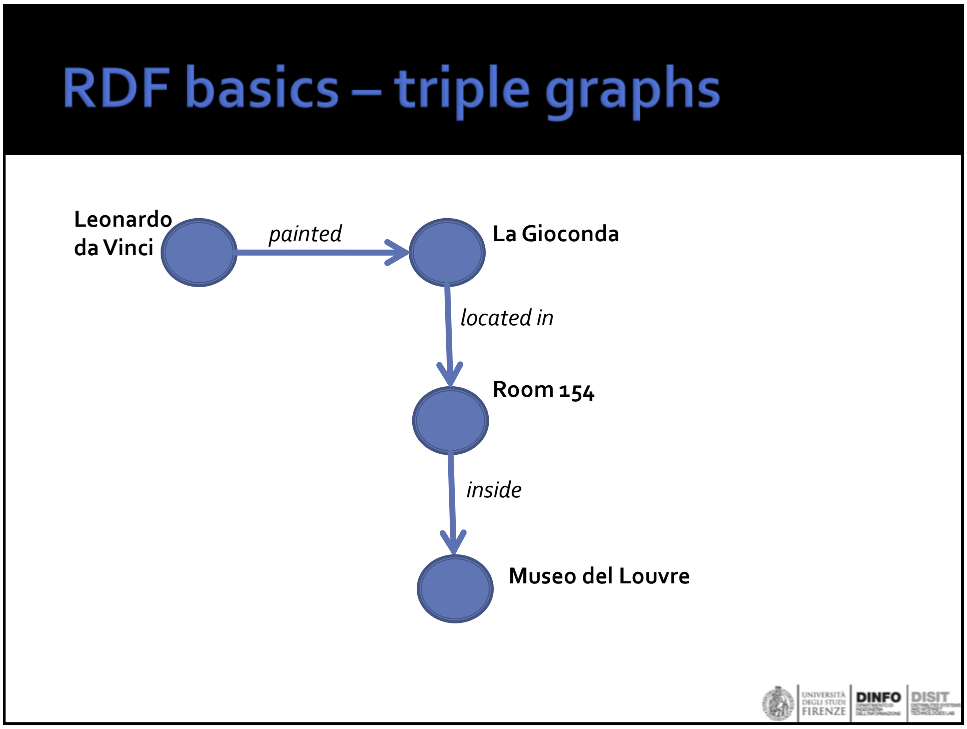
- W3C Resource Description Framework
- a W3C recommendation for the description of resources
- Basics:
  - each resource is identified by a not unique URI
  - resources are connected with other resources or primitive data using properties
  - also properties are identified by URIs

R1
property
R2



RDF basics - triples

- triples used to represent facts
  - subject
  - predicate
  - object
- Examples:
  - <Leonardo da Vinci> *painted* <Gioconda>
  - <Gioconda> *locatedIn* <Room 154>
  - <Room 154> *inside* <Museo del Louvre>



RDF basics – triple graphs

Leonardo da Vinci

painted

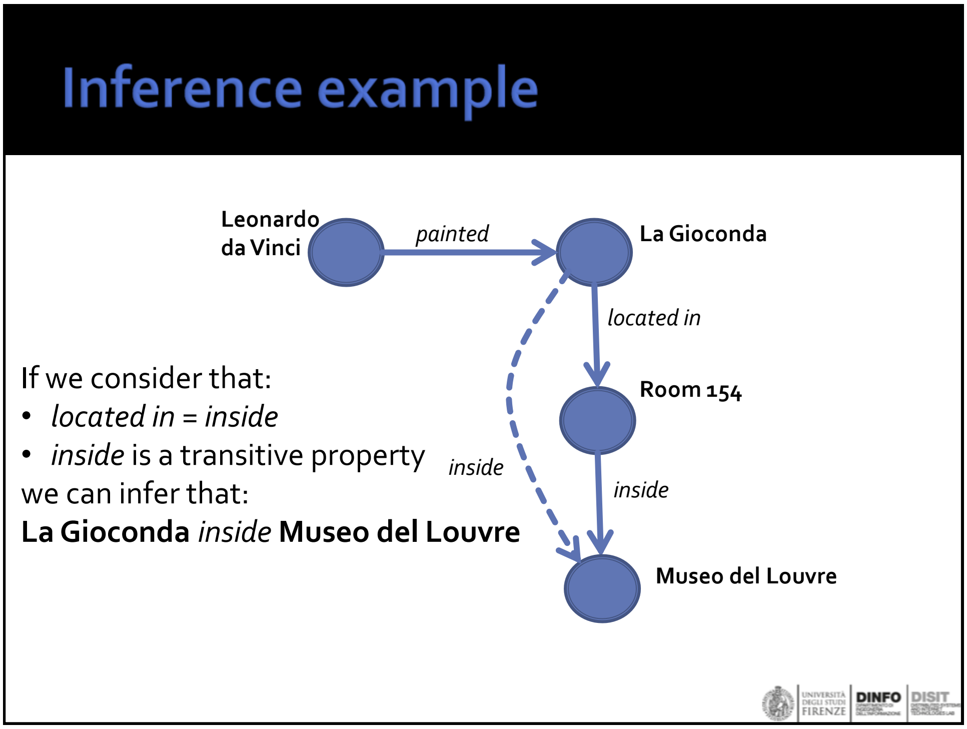
La Gioconda

located in

Room 154

inside

Museo del Louvre



Inference example

Leonardo da Vinci

painted

La Gioconda

located in

Room 154

inside

inside

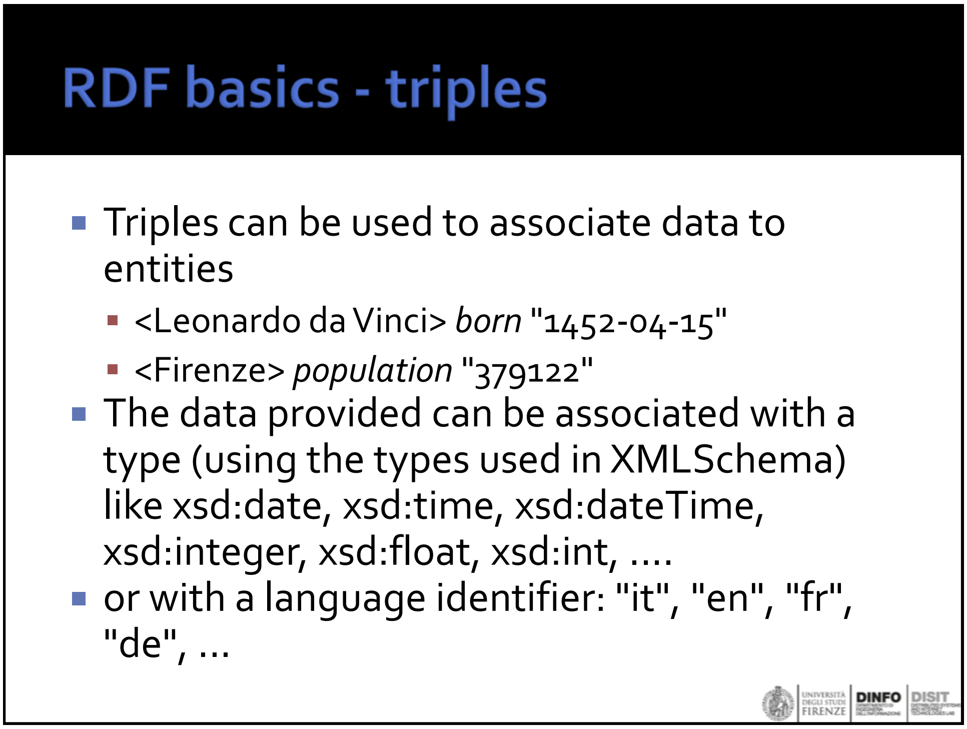
Museo del Louvre

If we consider that:

- *located in* = *inside*
- *inside* is a transitive property

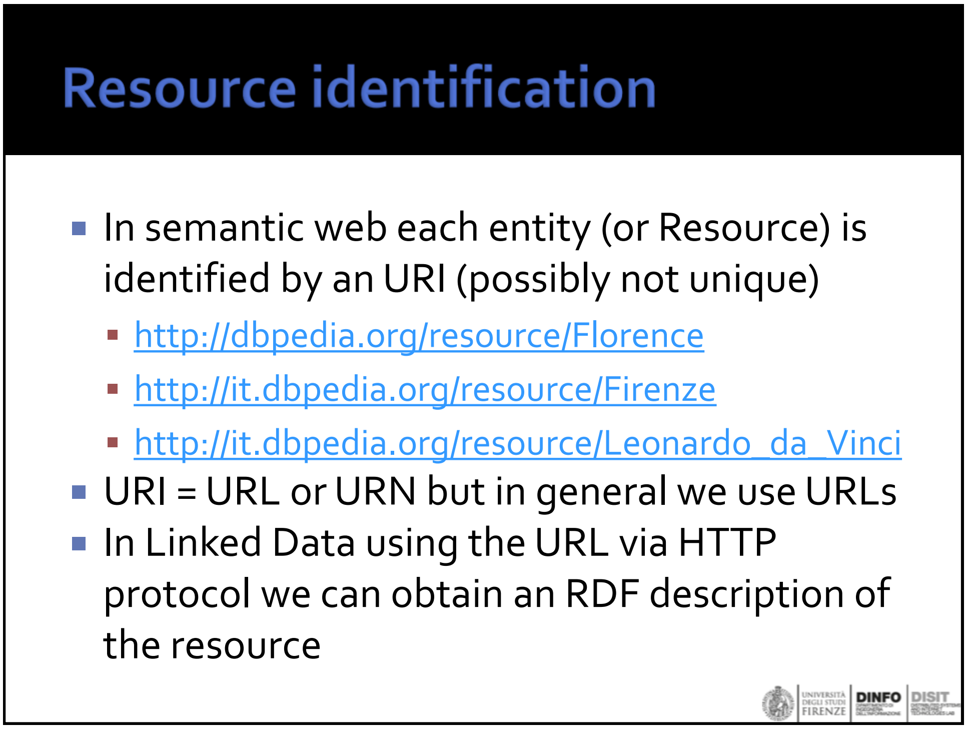
we can infer that:

La Gioconda *inside* **Museo del Louvre**



RDF basics - triples

- Triples can be used to associate data to entities
  - <Leonardo da Vinci> *born* "1452-04-15"
  - <Firenze> *population* "379122"
- The data provided can be associated with a type (using the types used in XMLSchema) like xsd:date, xsd:time, xsd:dateTime, xsd:integer, xsd:float, xsd:int, ...
- or with a language identifier: "it", "en", "fr", "de", …

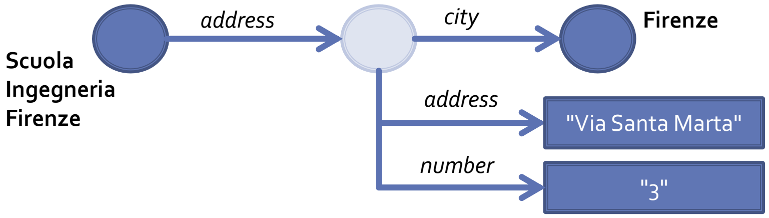


Resource identification

- In semantic web each entity (or Resource) is identified by an URI (possibly not unique)
  - http://dbpedia.org/resource/Florence
  - http://it.dbpedia.org/resource/Firenze
  - http://it.dbpedia.org/resource/Leonardo_da_Vinci
- URI = URL or URN but in general we use URLs
- In Linked Data using the URL via HTTP protocol we can obtain an RDF description of the resource

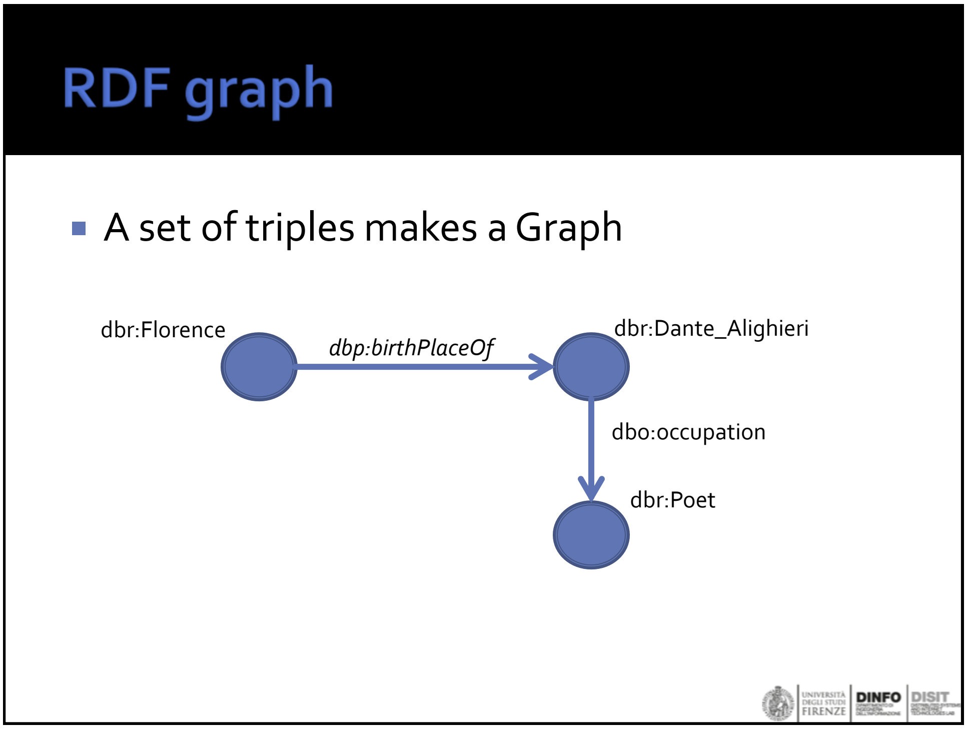
blank nodes

- Refer to an "unnamed" resource, are used to aggregate and structure entities, the name used for the definition is only local and in general cannot be used any more to refer to the part



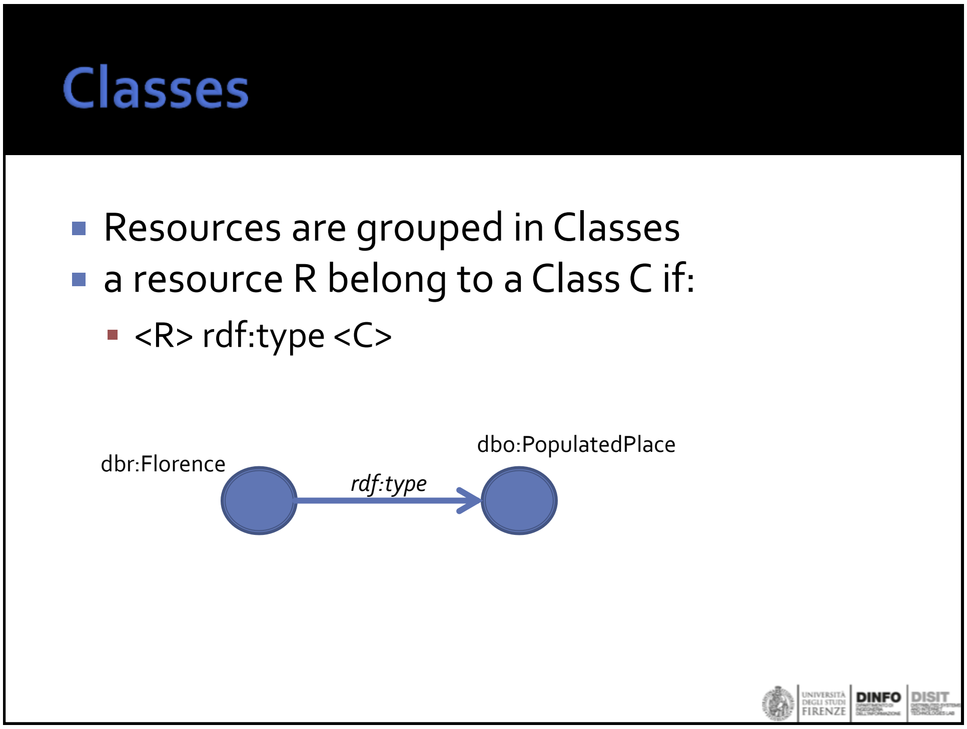
PREFIXes

- PREFIXES are used to shorten URIs
  - PREFIX dbr: <http://dbpedia.org/resource/>
  - **dbr:Florence** translated to http://dbpedia.org/resource/Florence
- Are similar to namespaces
- There are some standard prefixes:
  - rdf: < **http://www.w3.org/1999/02/22-rdf-syntax-ns#**>
  - rdfs: <**http://www.w3.org/2000/01/rdf-schema#>**
  - owl: < **http://www.w3.org/2002/07/owl#>**
  - skos: <**http://www.w3.org/2004/02/skos/core#**>
  - ...
  - use http://prefix.cc to search for common prefix definitions



RDF graph

- A set of triples makes a Graph

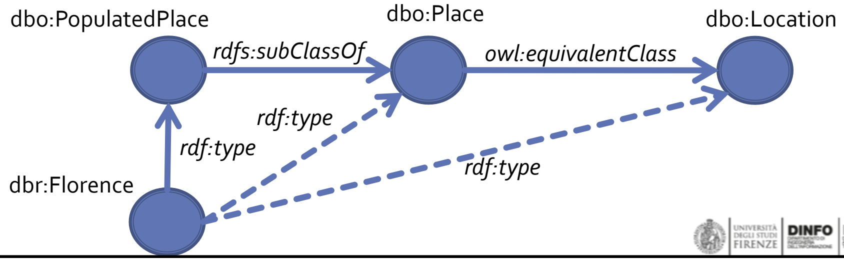


Classes

- Resources are grouped in Classes
- a resource R belong to a Class C if:
  - <R> rdf:type <C>

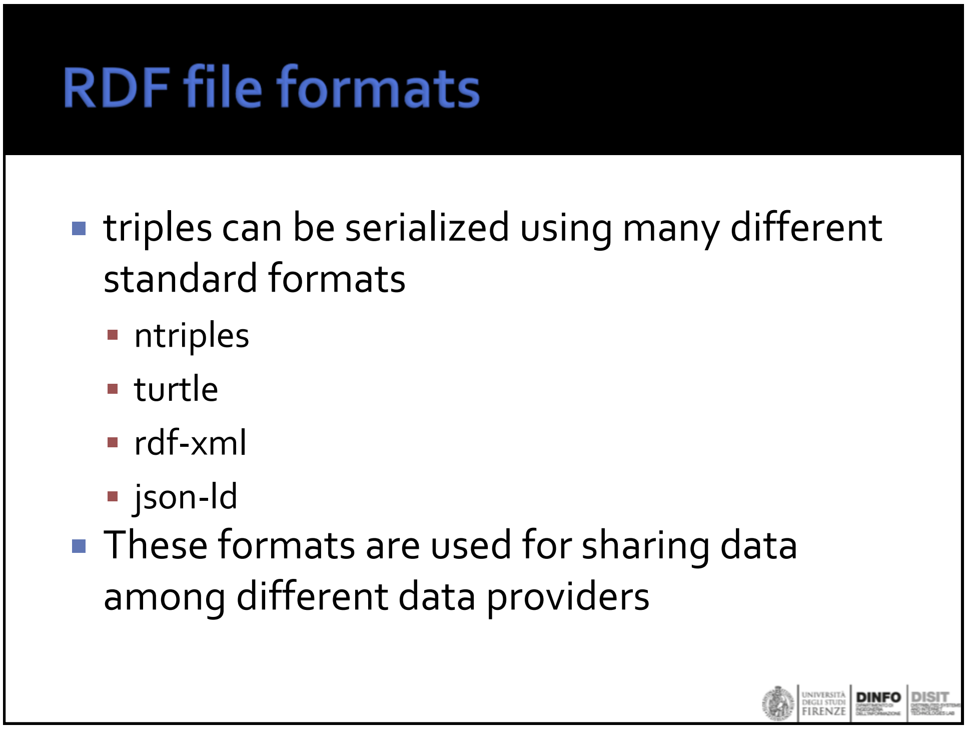
RDF description of classes

- Also classes are described using RDF
  - <C1> rdfs:subClassOf <C2>
    - states that C1 is a subclass of C2
  - <X> owl:equivalentClass <Y>
    - states that class X and Y are equivalent



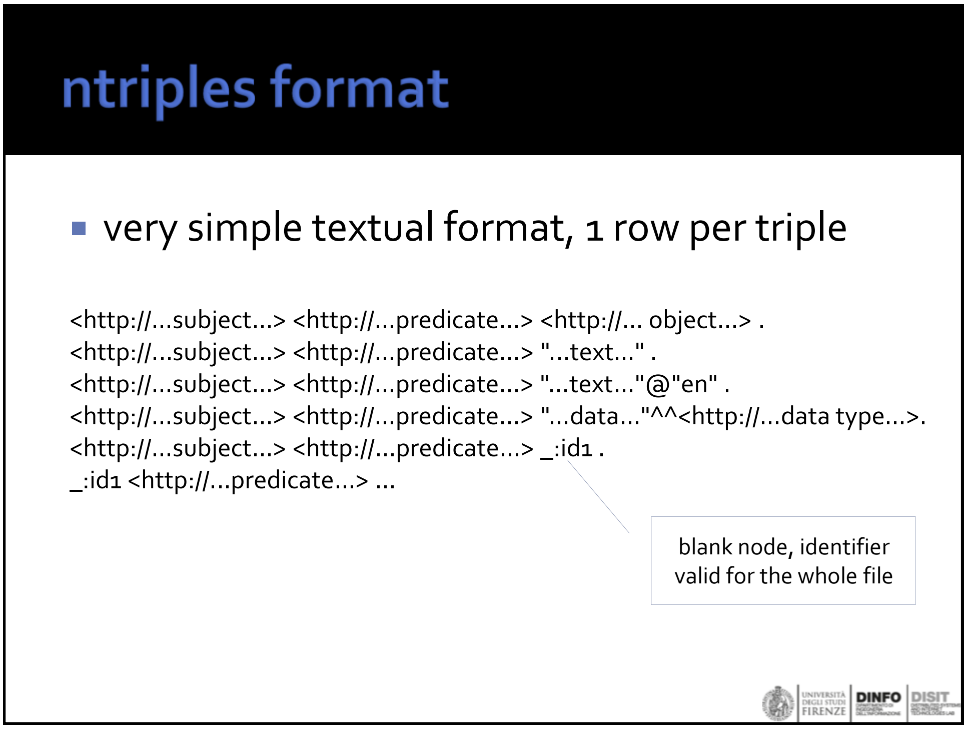
Ontology

- An Ontology (aka Vocabulary) is a description of a domain defining:
  - classes
  - properties
  - and their relations and properties
- Different "languages" are available:
  - The first attempt RDFS (RDF Schema)
  - after it was introduced Ontology Web Language 1.0
  - and then OWL 2.0



RDF file formats

- triples can be serialized using many different standard formats
  - ntriples
  - turtle
  - rdf-xml
  - json-ld
- These formats are used for sharing data among different data providers

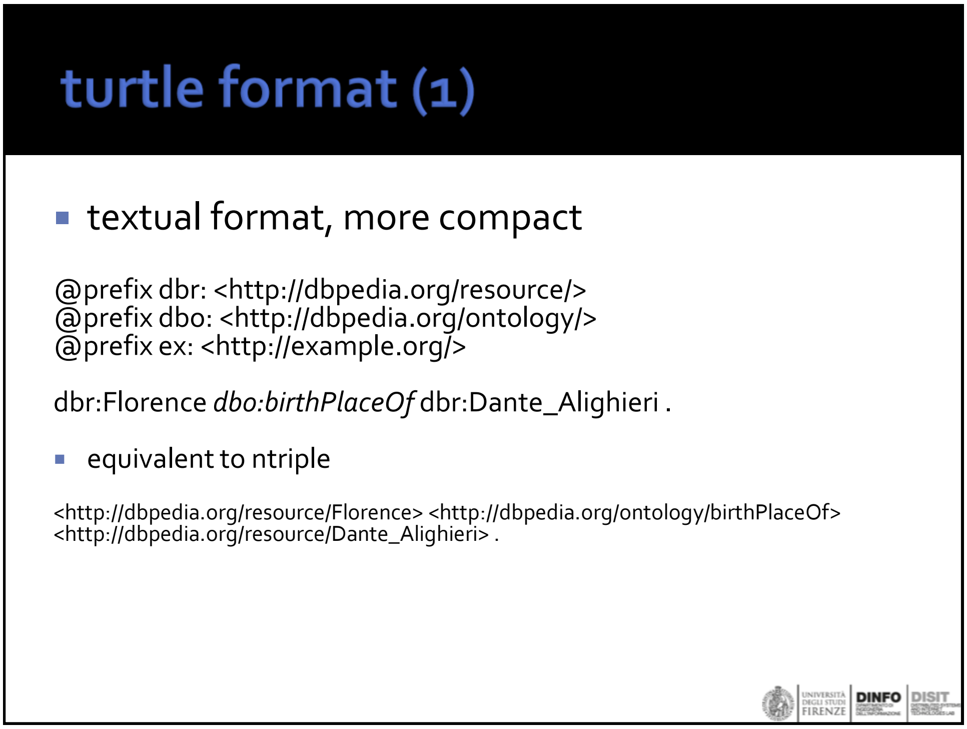


ntriples format

- very simple textual format, 1 row per triple

```
<http://...subject...> <http://...predicate...> <http://... object...> .
<http://...subject...> <http://...predicate...> "...text..." .
<http://...subject...> <http://...predicate...> "...text..."@"en" .
<http://...subject...> <http://...predicate...> "...data..."^^<http://...data type...>.
<http://...subject...> <http://...predicate...> _:id1 .
_:id1 <http://...predicate...> ...
```

blank node, identifier
valid for the whole file



turtle format (1)

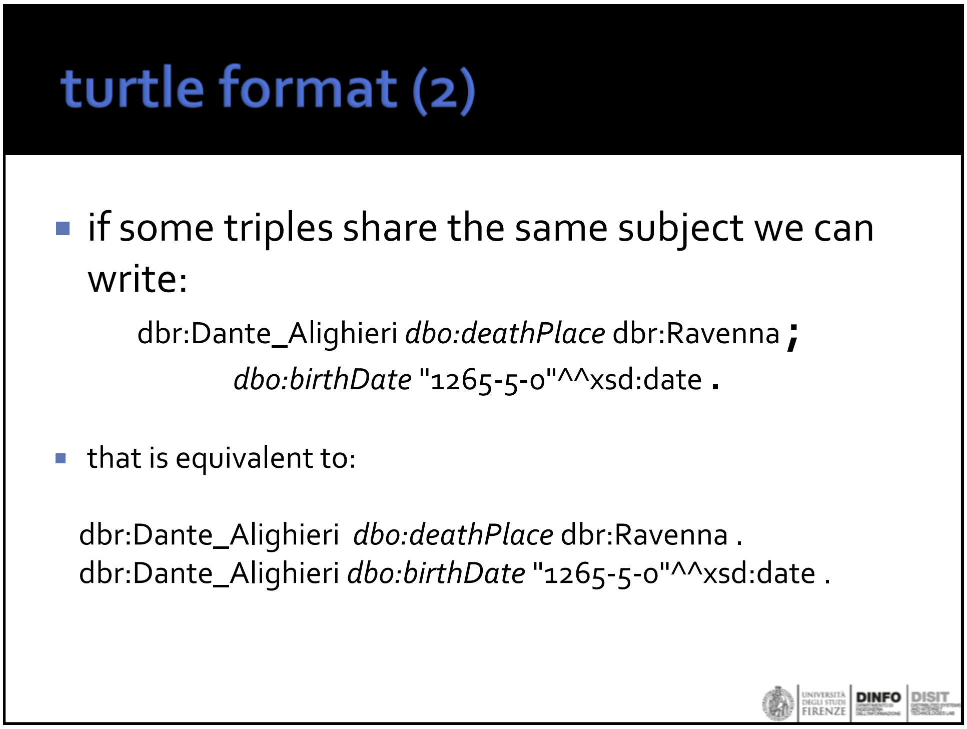
- textual format, more compact

@prefix dbr: <http://dbpedia.org/resource/>
@prefix dbo: <http://dbpedia.org/ontology/>
@prefix ex: <http://example.org/>

dbr:Florence *dbo:birthPlaceOf* dbr:Dante_Alighieri .

- equivalent to ntriple

<http://dbpedia.org/resource/Florence> <http://dbpedia.org/ontology/birthPlaceOf> <http://dbpedia.org/resource/Dante_Alighieri> .



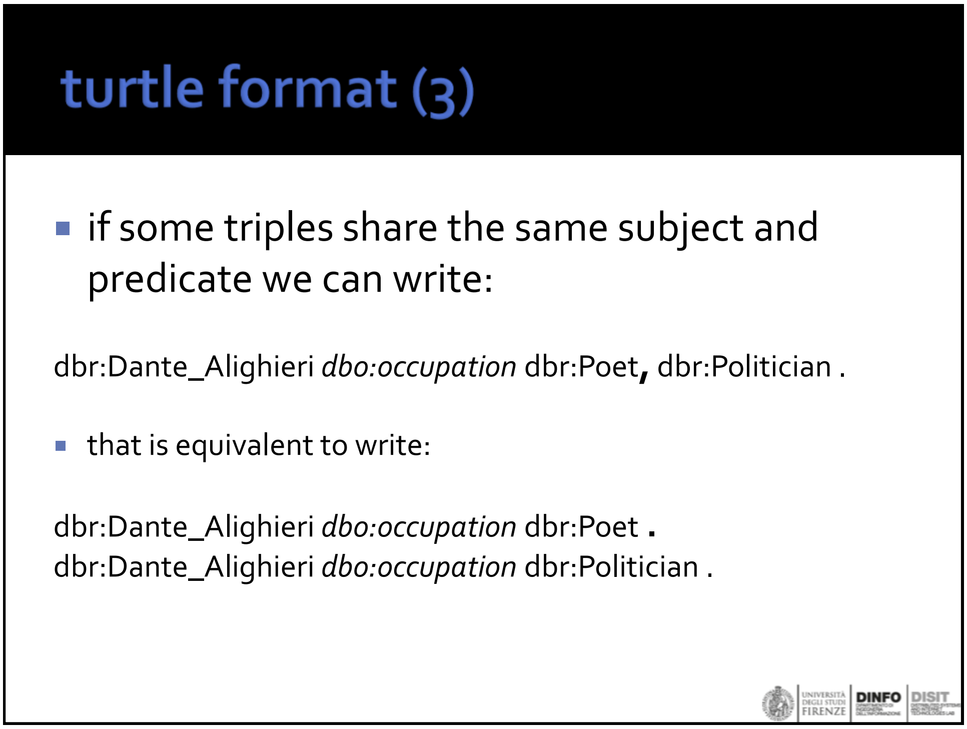
turtle format (2)

- if some triples share the same subject we can write:

dbr:Dante_Alighieri *dbo:deathPlace* dbr:Ravenna **;**
dbo:birthDate "1265-5-0"^^xsd:date **.**

- that is equivalent to:

dbr:Dante_Alighieri *dbo:deathPlace* dbr:Ravenna .
dbr:Dante_Alighieri *dbo:birthDate* "1265-5-0"^^xsd:date .



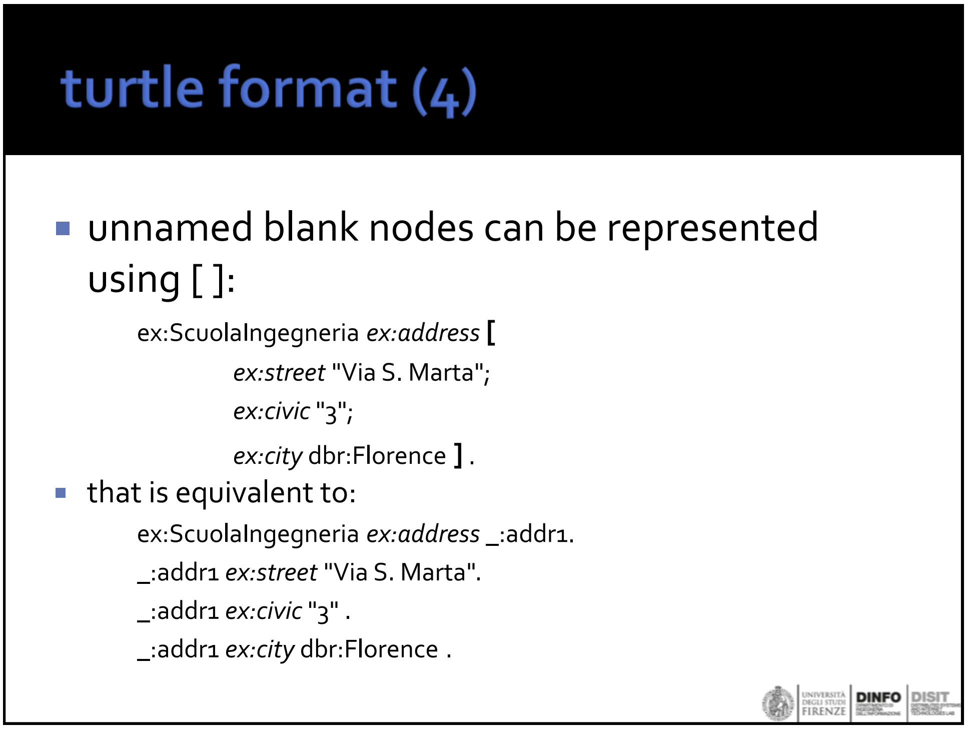
turtle format (3)

- if some triples share the same subject and predicate we can write:

```
dbr:Dante_Alighieri dbo:occupation dbr:Poet, dbr:Politician .
```

- that is equivalent to write:

```
dbr:Dante_Alighieri dbo:occupation dbr:Poet .
dbr:Dante_Alighieri dbo:occupation dbr:Politician .
```



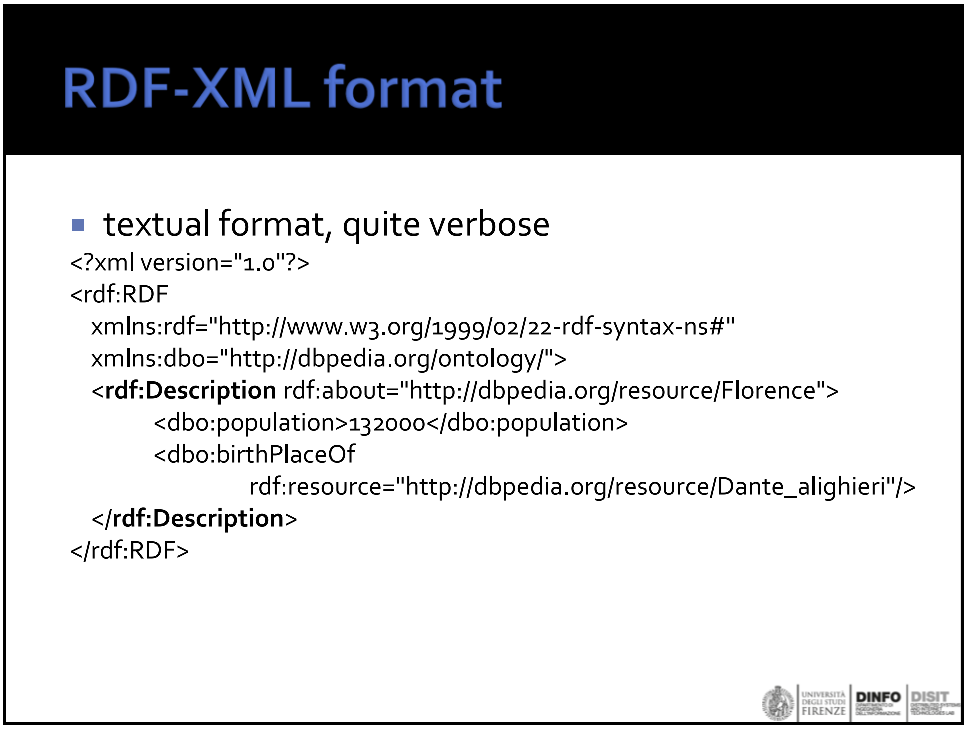
turtle format (4)

- unnamed blank nodes can be represented using [ ]:

```
ex:ScuolaIngegneria ex:address [
        ex:street "Via S. Marta";
        ex:civic "3";
        ex:city dbr:Florence ] .
```

- that is equivalent to:

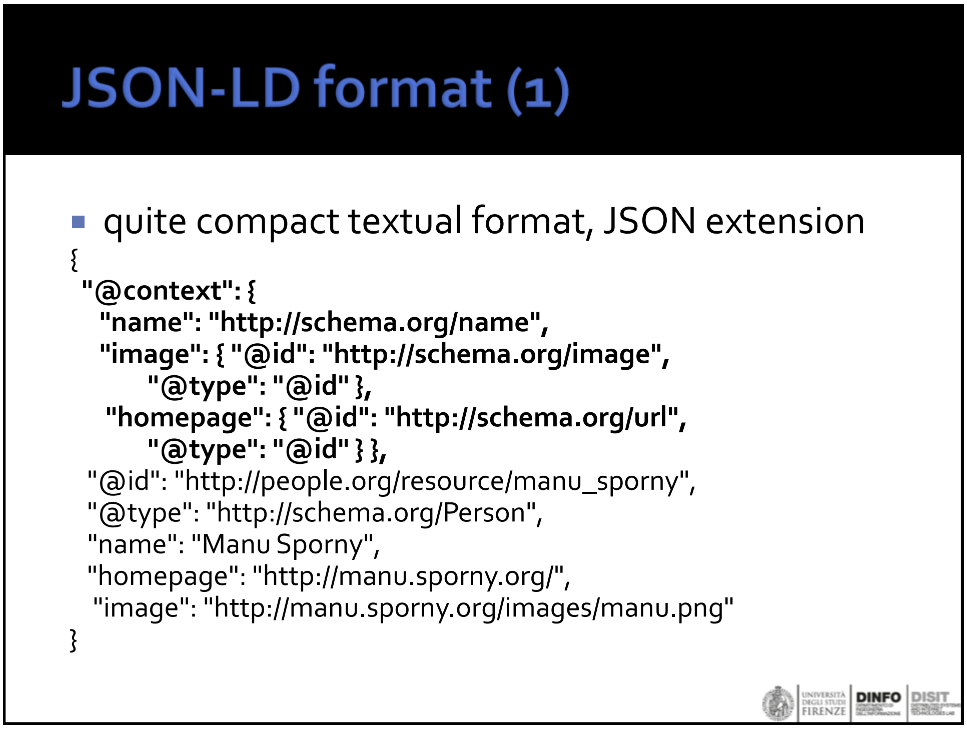
```
ex:ScuolaIngegneria ex:address _:addr1.
_:addr1 ex:street "Via S. Marta".
_:addr1 ex:civic "3" .
_:addr1 ex:city dbr:Florence .
```



RDF-XML format

- textual format, quite verbose

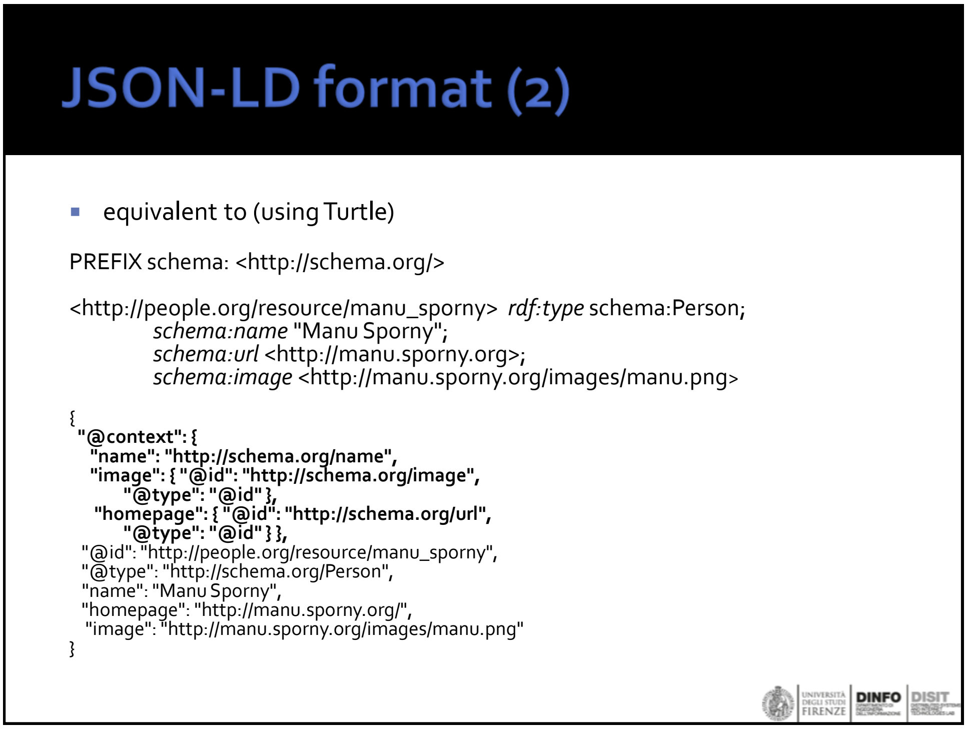
```
<?xml version="1.0"?>
<rdf:RDF
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:dbo="http://dbpedia.org/ontology/">
  <rdf:Description rdf:about="http://dbpedia.org/resource/Florence">
        <dbo:population>132000</dbo:population>
        <dbo:birthPlaceOf
                 rdf:resource="http://dbpedia.org/resource/Dante_alighieri"/>
  </rdf:Description>
</rdf:RDF>
```



JSON-LD format (1)

- quite compact textual format, JSON extension

```
{
 "@context": {
   "name": "http://schema.org/name",
   "image": { "@id": "http://schema.org/image",
        "@type": "@id" },
    "homepage": { "@id": "http://schema.org/url",
        "@type": "@id" } },
  "@id": "http://people.org/resource/manu_sporny",
  "@type": "http://schema.org/Person",
  "name": "Manu Sporny",
  "homepage": "http://manu.sporny.org/",
   "image": "http://manu.sporny.org/images/manu.png"
}
```



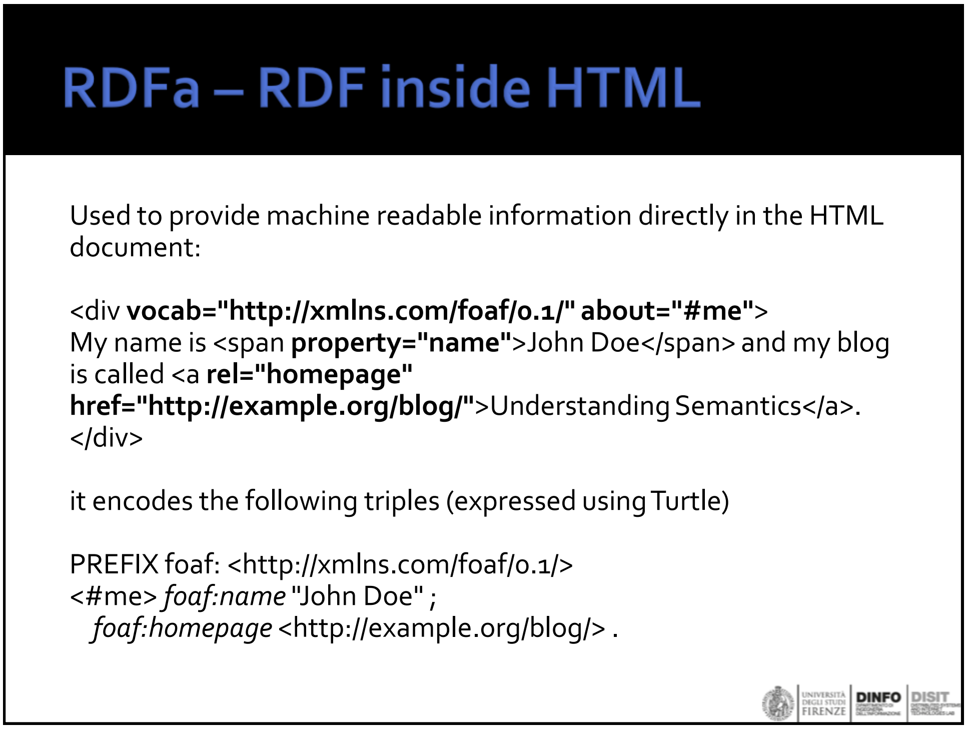
JSON-LD format (2)

- equivalent to (using Turtle)

```
PREFIX schema: <http://schema.org/>

<http://people.org/resource/manu_sporny>  rdf:type schema:Person;
            schema:name "Manu Sporny";
            schema:url <http://manu.sporny.org>;
            schema:image <http://manu.sporny.org/images/manu.png>
```

```
{
 "@context": {
   "name": "http://schema.org/name",
   "image": { "@id": "http://schema.org/image",
        "@type": "@id" },
    "homepage": { "@id": "http://schema.org/url",
        "@type": "@id" } },
  "@id": "http://people.org/resource/manu_sporny",
  "@type": "http://schema.org/Person",
  "name": "Manu Sporny",
  "homepage": "http://manu.sporny.org/",
   "image": "http://manu.sporny.org/images/manu.png"
}
```



RDFa – RDF inside HTML

Used to provide machine readable information directly in the HTML document:

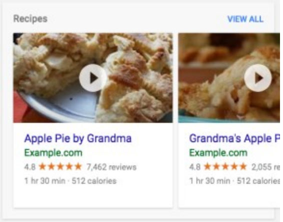
<div **vocab="http://xmlns.com/foaf/0.1/" about="#me"**>
My name is <span **property="name"**>John Doe</span> and my blog is called <a **rel="homepage" href="http://example.org/blog/"**>Understanding Semantics</a>.
</div>

it encodes the following triples (expressed using Turtle)

PREFIX foaf: <http://xmlns.com/foaf/0.1/>
<#me> *foaf:name* "John Doe" ;
  foaf:homepage <http://example.org/blog/> .

Google – Structured Data

- Use JSON-LD or RDFa in HTML
- Using properties and classes defined by http://schema.org
- To provide rich results on google search results



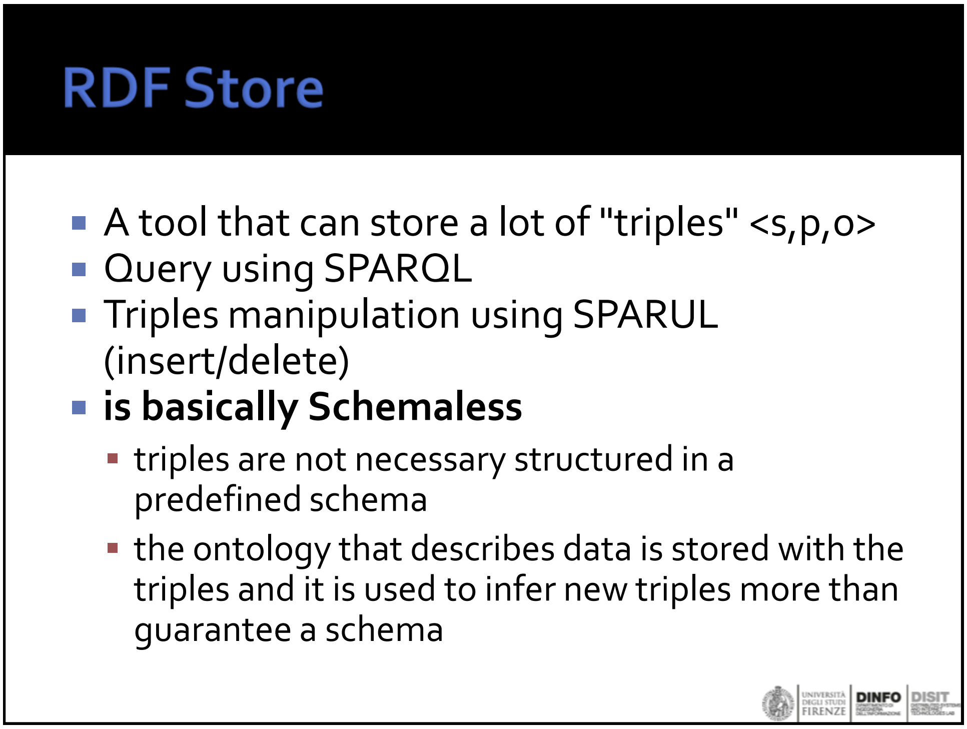
Apple Pie by Grandma
Example.com
4.8 ★★★★★ 7,462 reviews
1 hr 30 min · 512 calories

Grandma's Apple P
Example.com
4.8 ★★★★★ 2,055 re
1 hr 30 min · 512 calories

```
<script type="application/ld+json">
{
  "@context": "http://schema.org/",
  "@type": "Recipe",
  "name": "Grandma's Holiday Apple Pie",
  "author": "Elaine Smith",
  "image": "http://images.org/receipe1.jpg",
  "description": "A classic apple pie.",
  "aggregateRating": {
    "@type": "AggregateRating",
    "ratingValue": "4",
    "reviewCount": "276",
    "bestRating": "5",
    "worstRating": "1"
  },
…
```

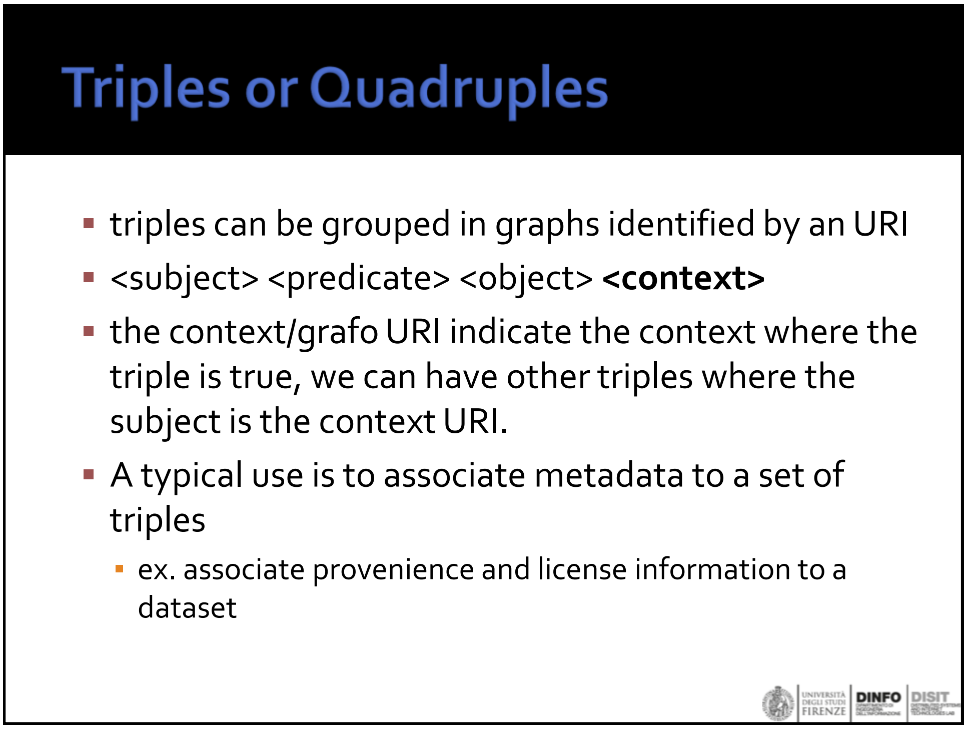
RDF Store

- A particular kind of NoSQL database
  - the only one with a **standard data model** (RDF)
  - the only one with **standard interchange formats** (RDFXML, Turtle, Ntriple, JSON-LD)
  - the only one with a **standard query language SPARQL** and with a standard communication protocol
  - the only one with a **standard manipulation language SPARUL**
- We can have many diferent standard implementations that guarantee:
  - interoperability
  - avoids vendor lock-in



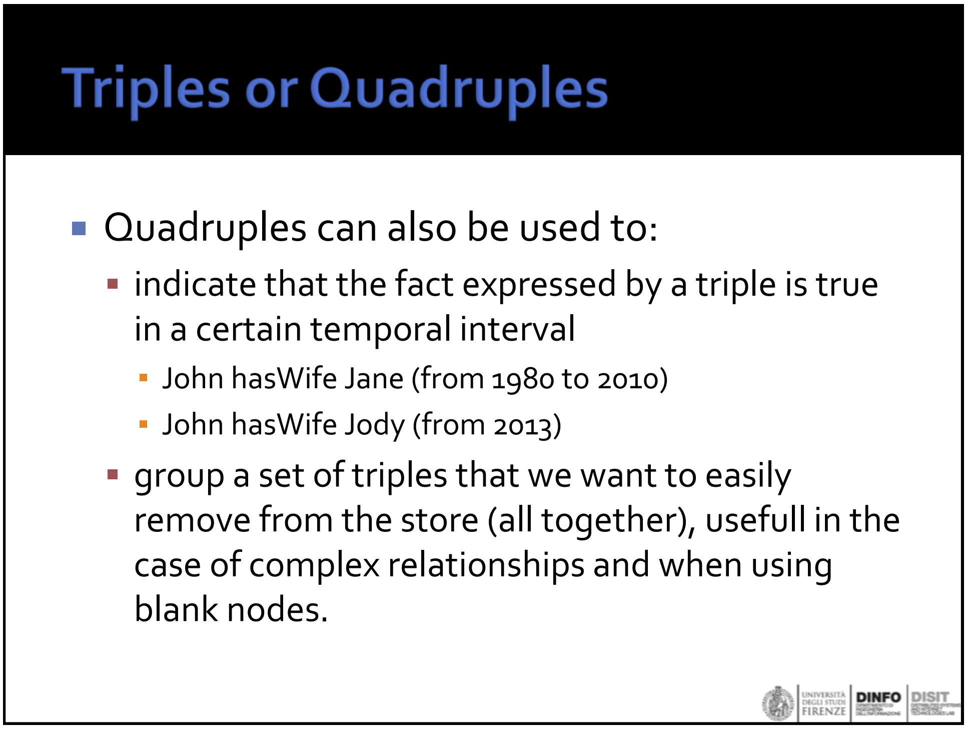
RDF Store

- A tool that can store a lot of "triples" <s,p,o>
- Query using SPARQL
- Triples manipulation using SPARUL (insert/delete)
- **is basically Schemaless**
  - triples are not necessary structured in a predefined schema
  - the ontology that describes data is stored with the triples and it is used to infer new triples more than guarantee a schema



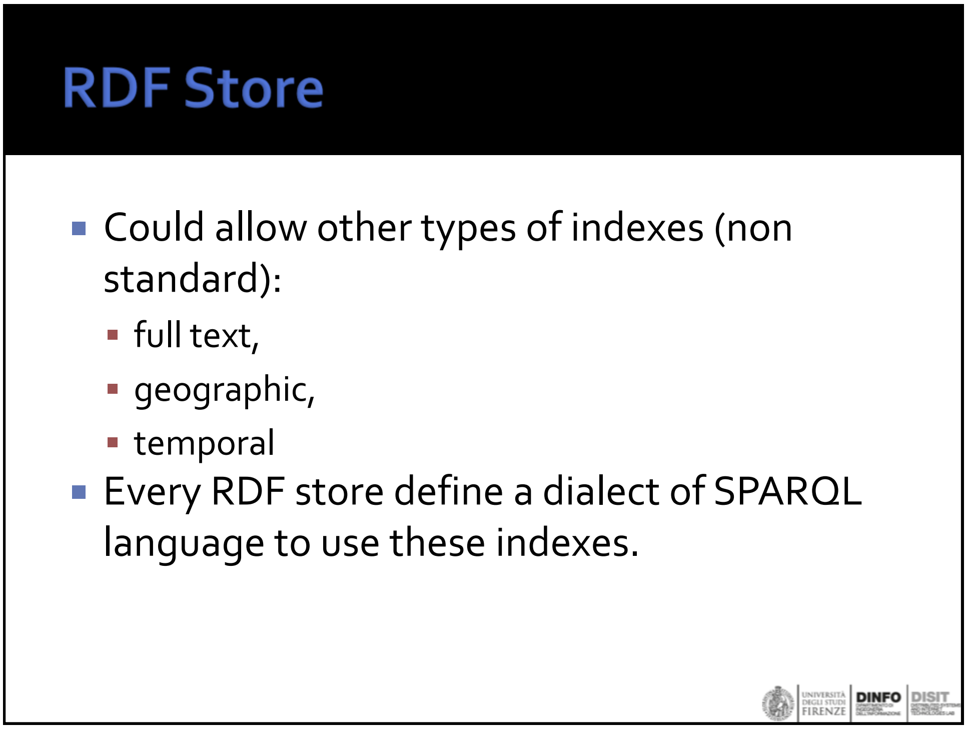
Triples or Quadruples

- triples can be grouped in graphs identified by an URI
- <subject> <predicate> <object> **<context>**
- the context/grafo URI indicate the context where the triple is true, we can have other triples where the subject is the context URI.
- A typical use is to associate metadata to a set of triples
  - ex. associate provenience and license information to a dataset



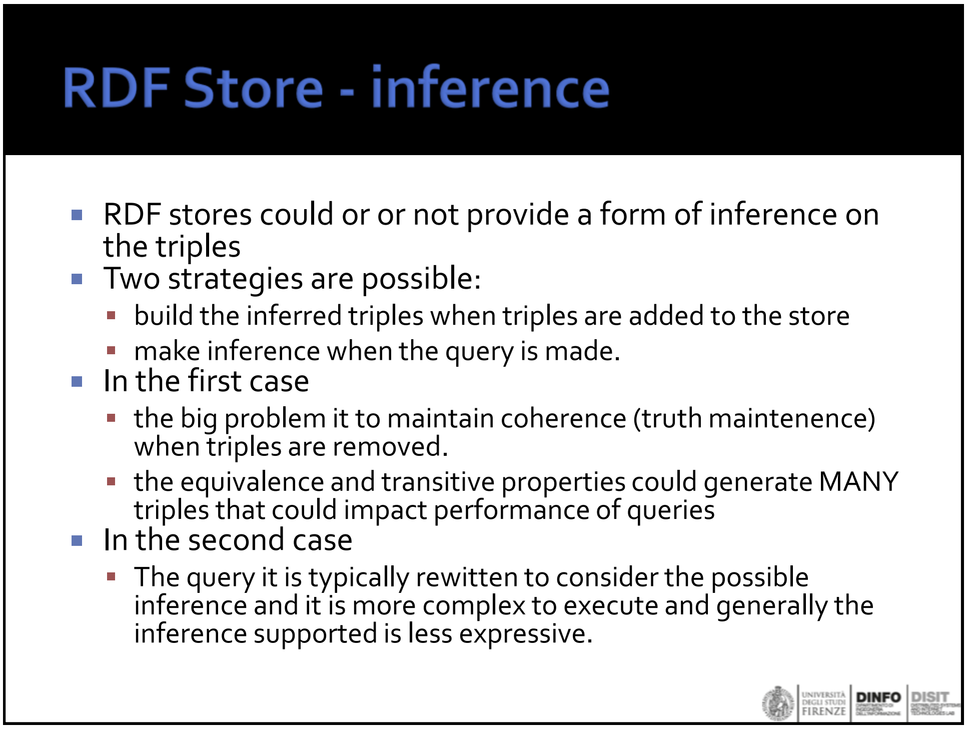
Triples or Quadruples

- Quadruples can also be used to:
  - indicate that the fact expressed by a triple is true in a certain temporal interval
    - John hasWife Jane (from 1980 to 2010)
    - John hasWife Jody (from 2013)
  - group a set of triples that we want to easily remove from the store (all together), usefull in the case of complex relationships and when using blank nodes.



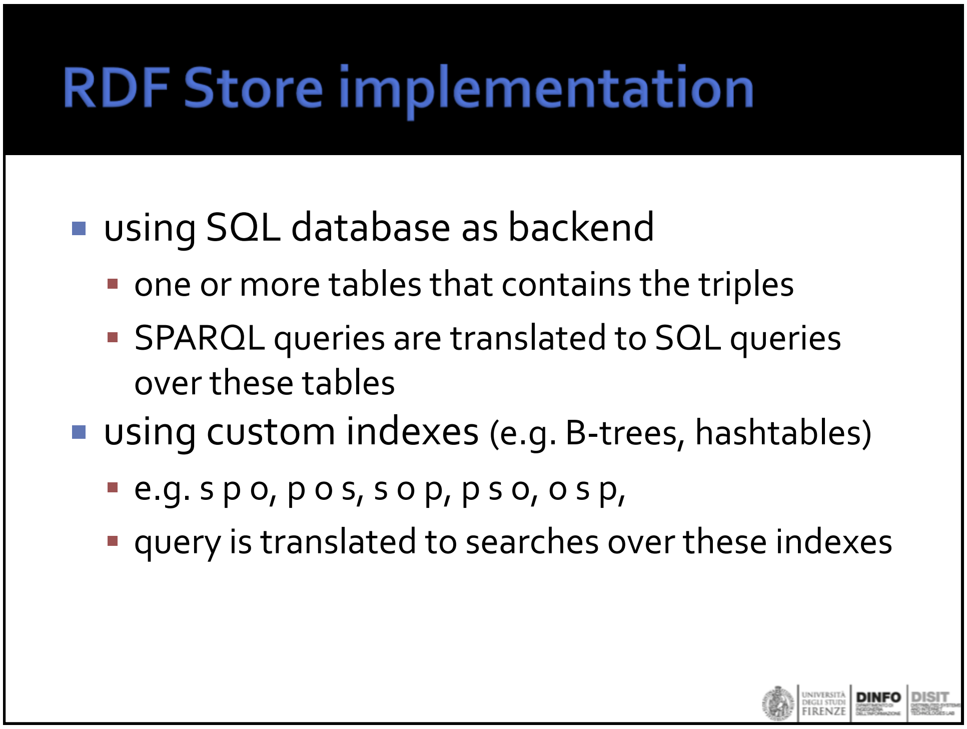
RDF Store

- Could allow other types of indexes (non standard):
  - full text,
  - geographic,
  - temporal
- Every RDF store define a dialect of SPARQL language to use these indexes.



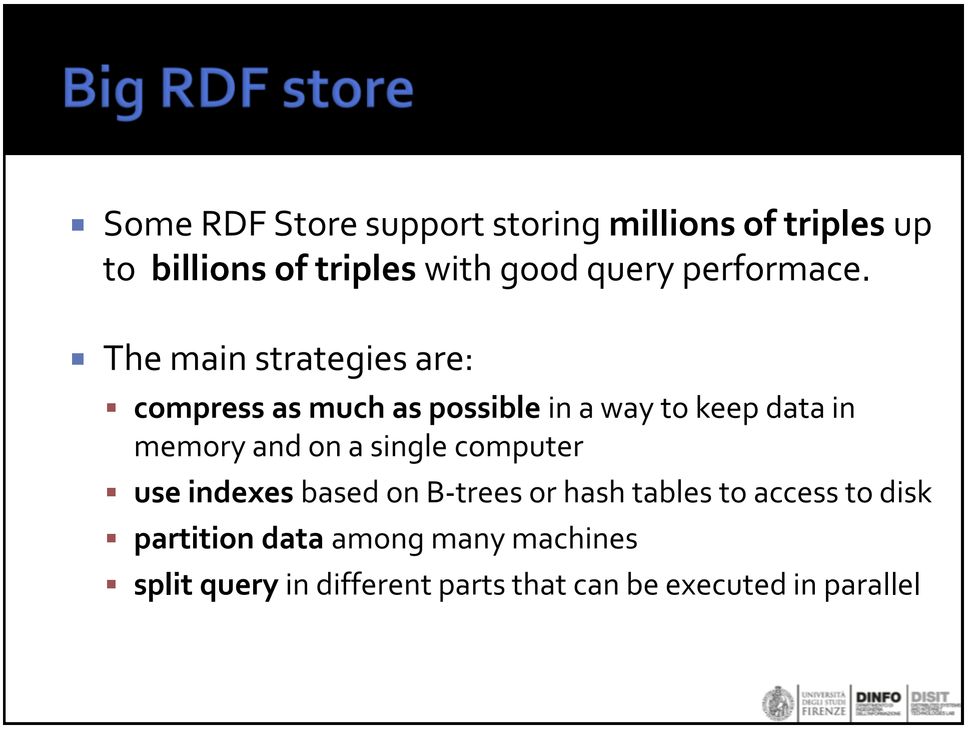
RDF Store - inference

- RDF stores could or or not provide a form of inference on the triples
- Two strategies are possible:
  - build the inferred triples when triples are added to the store
  - make inference when the query is made.
- In the first case
  - the big problem it to maintain coherence (truth maintenence) when triples are removed.
  - the equivalence and transitive properties could generate MANY triples that could impact performance of queries
- In the second case
  - The query it is typically rewitten to consider the possible inference and it is more complex to execute and generally the inference supported is less expressive.



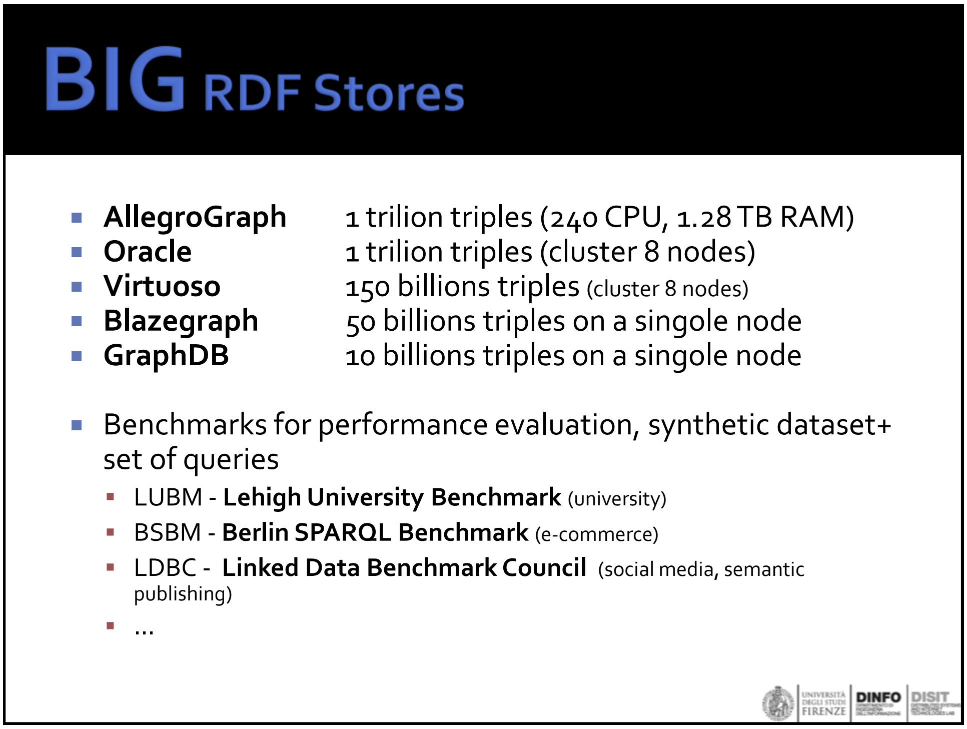
RDF Store implementation

- using SQL database as backend
  - one or more tables that contains the triples
  - SPARQL queries are translated to SQL queries over these tables
- using custom indexes (e.g. B-trees, hashtables)
  - e.g. s p o, p o s, s o p, p s o, o s p,
  - query is translated to searches over these indexes



Big RDF store

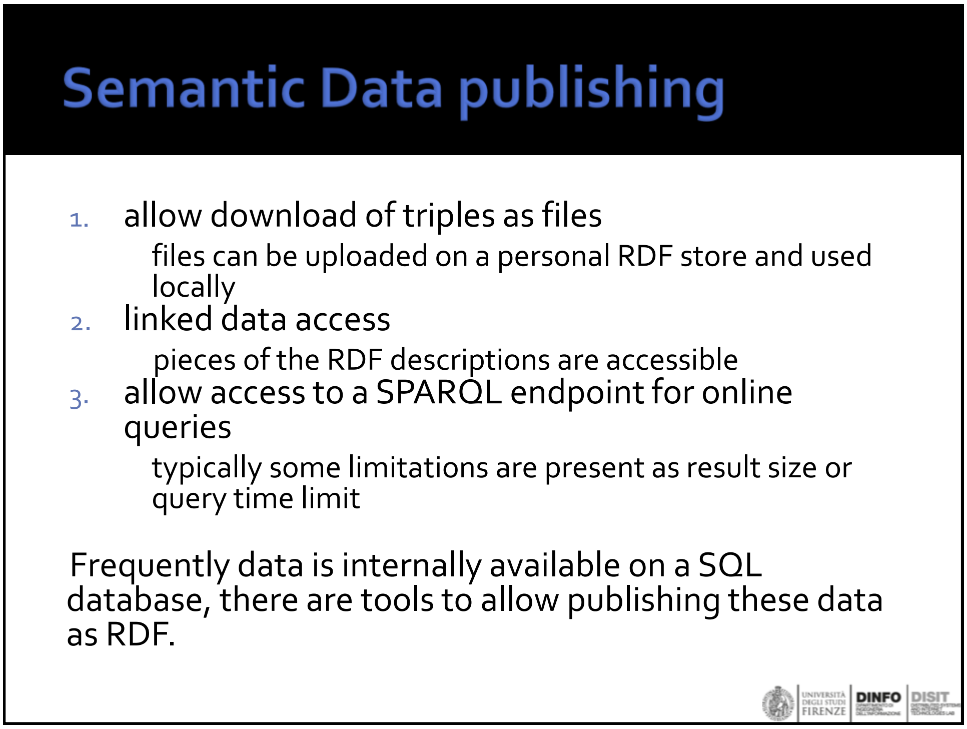
- Some RDF Store support storing **millions of triples** up to **billions of triples** with good query performace.
- The main strategies are:
  - **compress as much as possible** in a way to keep data in memory and on a single computer
  - **use indexes** based on B-trees or hash tables to access to disk
  - **partition data** among many machines
  - **split query** in different parts that can be executed in parallel



BIG RDF Stores

AllegroGraph	1 trilion triples (240 CPU, 1.28 TB RAM)
Oracle	1 trilion triples (cluster 8 nodes)
Virtuoso	150 billions triples (cluster 8 nodes)
Blazegraph	50 billions triples on a singole node
GraphDB	10 billions triples on a singole node

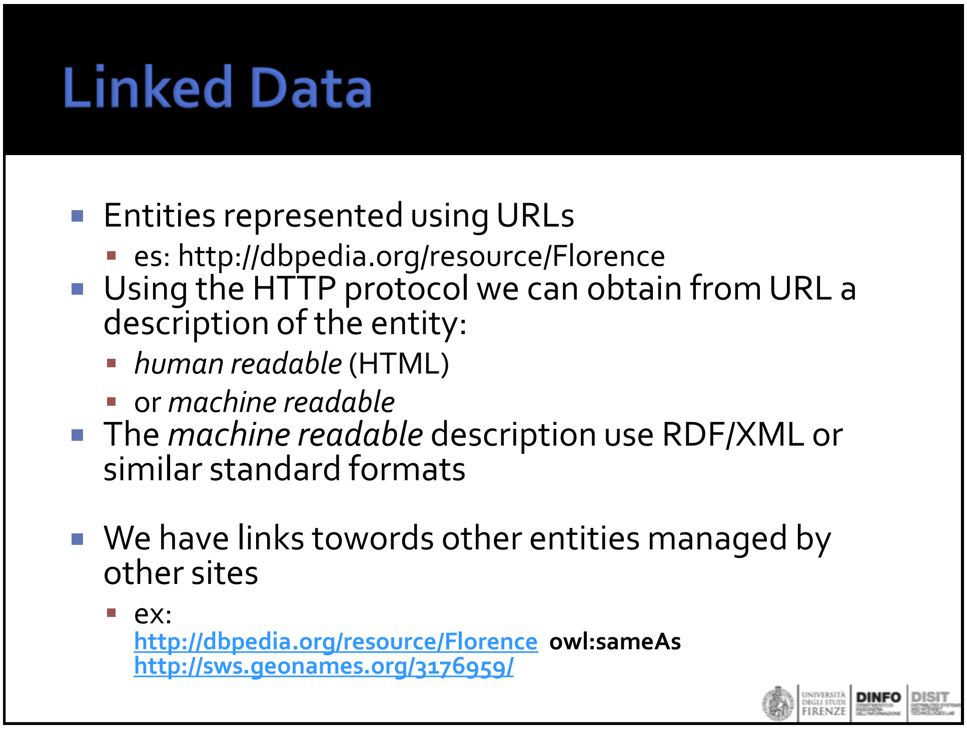
- Benchmarks for performance evaluation, synthetic dataset+ set of queries
  - LUBM - **Lehigh University Benchmark** (university)
  - BSBM - **Berlin SPARQL Benchmark** (e-commerce)
  - LDBC - **Linked Data Benchmark Council** (social media, semantic publishing)
  - …



Semantic Data publishing

1. allow download of triples as files
   files can be uploaded on a personal RDF store and used locally
2. linked data access
   pieces of the RDF descriptions are accessible
3. allow access to a SPARQL endpoint for online queries
   typically some limitations are present as result size or query time limit

Frequently data is internally available on a SQL database, there are tools to allow publishing these data as RDF.

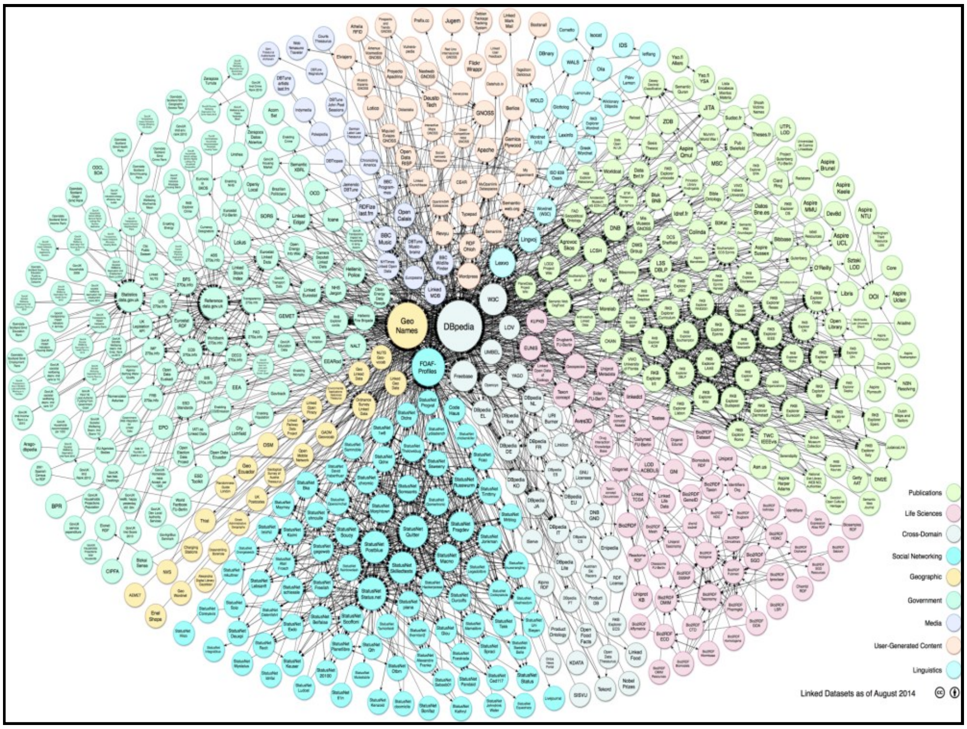


Linked Data

- Entities represented using URLs
  - es: http://dbpedia.org/resource/Florence
- Using the HTTP protocol we can obtain from URL a description of the entity:
  - *human readable* (HTML)
  - or *machine readable*
- The *machine readable* description use RDF/XML or similar standard formats
- We have links towords other entities managed by other sites
  - ex:
    http://dbpedia.org/resource/Florence **owl:sameAs** http://sws.geonames.org/3176959/

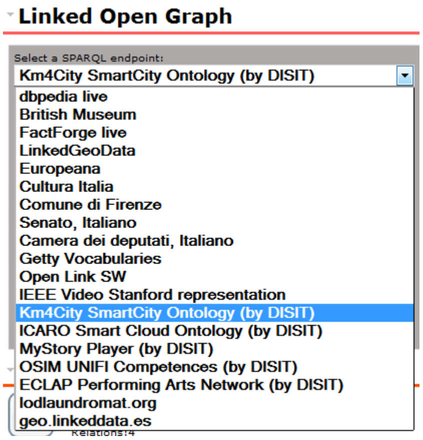
Linked Open Data (LOD)

- LOD = Linked Data + Open Data
- data is available with an open license (ex. creative common attribution)
- LOD Cloud
  - interconnected data
  - see next slide



Linked Datasets as of August 2014

Linked Open Graph
http://log.disit.org
A bus stop info….

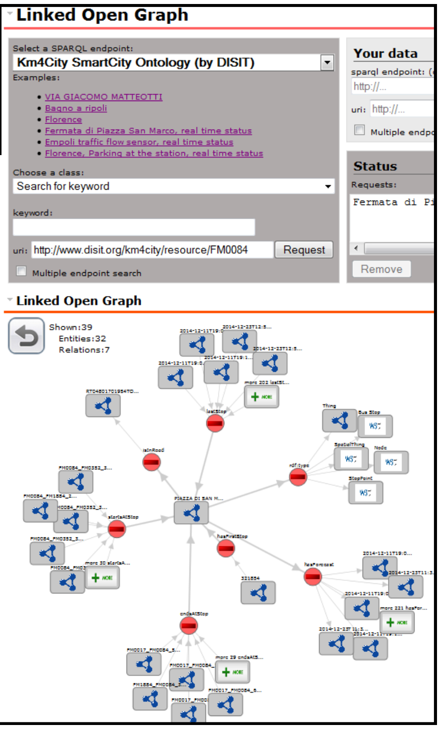


Linked Open Graph

Select a SPARQL endpoint:

Km4City SmartCity Ontology (by DISIT)

- dbpedia live
- British Museum
- FactForge live
- LinkedGeoData
- Europeana
- Cultura Italia
- Comune di Firenze
- Senato, Italiano
- Camera dei deputati, Italiano
- Getty Vocabularies
- Open Link SW
- IEEE Video Stanford representation
- Km4City SmartCity Ontology (by DISIT)
- ICARO Smart Cloud Ontology (by DISIT)
- MyStory Player (by DISIT)
- OSIM UNIFI Competences (by DISIT)
- ECLAP Performing Arts Network (by DISIT)
- lodlaundromat.org
- geo.linkeddata.es



Linked Open Graph

Select a SPARQL endpoint: Km4City SmartCity Ontology (by DISIT)

Examples:

- VIA GIACOMO MATTEOTTI
- Bagno a ripoli
- Florence
- Fermata di Piazza San Marco, real time status
- Empoli traffic flow sensor, real time status
- Florence, Parking at the station, real time status

Choose a class: Search for keyword

keyword:

uri: http://www.disit.org/km4city/resource/FM0084 Request

☐ Multiple endpoint search

Your data

sparql endpoint:

http://...

uri: http://...

☐ Multiple endp

Status

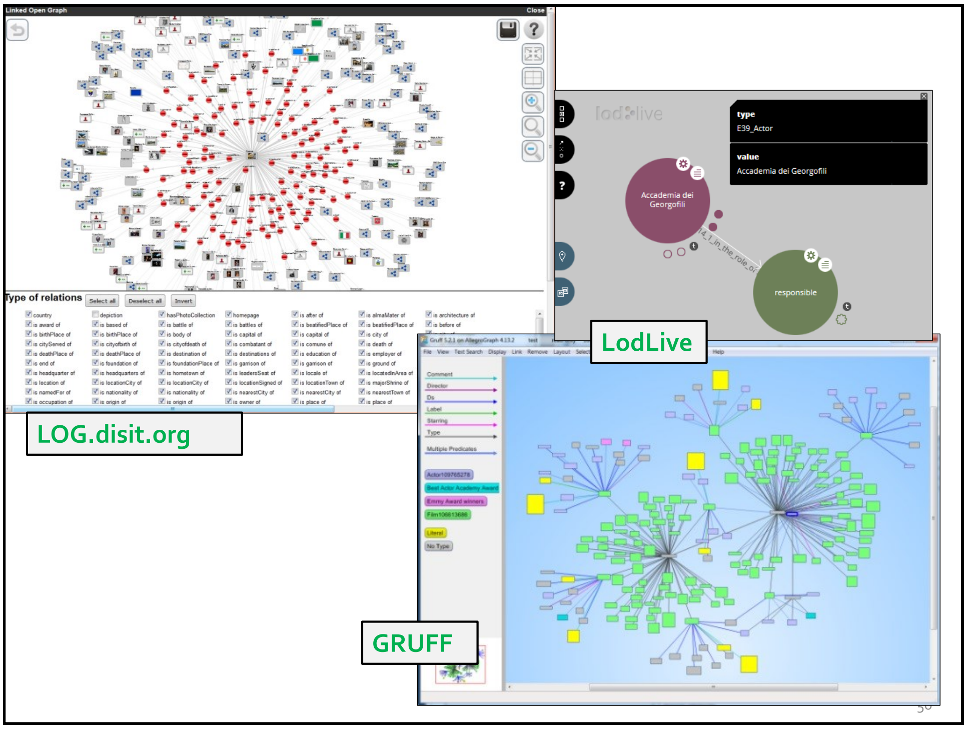
Requests:

Fermata di Pi

Remove

Linked Open Graph

Shown:39
Entities:32
Relations:7



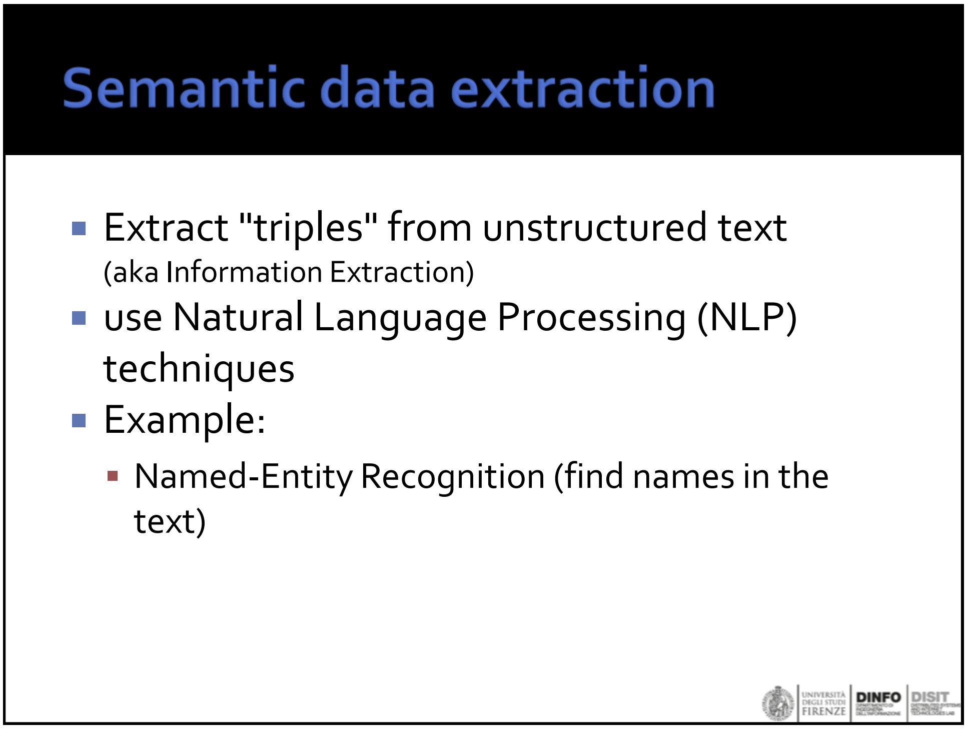
Type of relations Select all Deselect all Invert

☑ country
☑ is award of
☑ is birthPlace of
☑ is cityServed of
☑ is deathPlace of
☑ is end of
☑ is headquarter of
☑ is location of
☑ is namedFor of
☑ is occupation of
☐ depiction
☑ is based of
☑ is birthPlace of
☑ is cityofbirth of
☑ is deathPlace of
☑ is foundation of
☑ is headquarters of
☑ is locationCity of
☑ is nationality of
☑ is origin of
☑ hasPhotoCollection
☑ is battle of
☑ is body of
☑ is cityofdeath of
☑ is destination of
☑ is foundationPlace of
☑ is hometown of
☑ is locationCity of
☑ is nationality of
☑ is origin of
☑ homepage
☑ is battles of
☑ is capital of
☑ is combatant of
☑ is destinations of
☑ is garrison of
☑ is leadersSeat of
☑ is locationSigned of
☑ is nearestCity of
☑ is owner of
☑ is after of
☑ is beatifiedPlace of
☑ is capital of
☑ is comune of
☑ is education of
☑ is garrison of
☑ is locale of
☑ is locationTown of
☑ is nearestCity of
☑ is place of
☑ is almaMater of
☑ is beatifiedPlace of
☑ is city of
☑ is death of
☑ is employer of
☑ is ground of
☑ is locatedInArea of
☑ is majorShrine of
☑ is nearestTown of
☑ is place of
☑ is architecture of
☑ is before of

LOG.disit.org

LodLive

GRUFF



Semantic data extraction

- Extract "triples" from unstructured text (aka Information Extraction)
- use Natural Language Processing (NLP) techniques
- Example:
  - Named-Entity Recognition (find names in the text)