Consiglio Nazionale delle Ricerche

STLAB
SEMANTIC TECHNOLOGY LABORATORY

Francesco Poggi
Istituto di Scienze e Tecnologie della Cognizione (CNR-ISTC)
francesco.poggi@cnr.it

Apache Jena - un framework Java applicazioni Semantic Web e Linked Data

Apache Jena

- Un framework Java per sviluppare applicazioni del Semantic Web (SW)
- Fornisce Java API per le principali tecnologie del SW
- Sviluppato da Brian McBride per HP, ora un progetto Apache
- Usato per fare parsing, creare ed effettuare ricerche su modelli RDF, e molto altro...

Apache Jena documentation overview

- The RDF API - the core RDF API in Jena
- SPARQL - querying and updating RDF models using the SPARQL standards
- Fuseki - SPARQL server which can present RDF data and answer SPARQL queries over HTTP
- I/O - reading and writing RDF data
- RDF Connection - a SPARQL API for local datasets and remote services
- Assembler - describing recipes for constructing Jena models declaratively using RDF
- Inference - using the Jena rules engine and other inference algorithms to derive consequences from RDF models
- Ontology - support for handling OWL models in Jena
- Data and RDFS - apply RDFS to graphs in a dataset
- TDB2 - a fast persistent triple store that stores directly to disk
- TDB - Original TDB database
- SHACL - SHACL processor for Jena
- ShEx - ShEx processor for Jena
- Text Search - enhanced indexes using Lucene for more efficient searching of text literals in Jena models and datasets.
- GeoSPARQL - support for GeoSPARQL
- Permissions - a permissions wrapper around Jena RDF implementation
- Tools - various command-line tools and utilities to help developers manage RDF data and other aspects of Jena
- How-To's - various topic-specific how-to documents
- QueryBuilder - Classes to simplify the programmatic building of various query and update statements.
- Extras - various modules that provide utilities and larger packages that make Apache Jena development or usage easier but that do not fall within the standard Jena framework.
- Javadoc - JavaDoc generated from the Jena source

Vedi https://jena.apache.org/documentation/

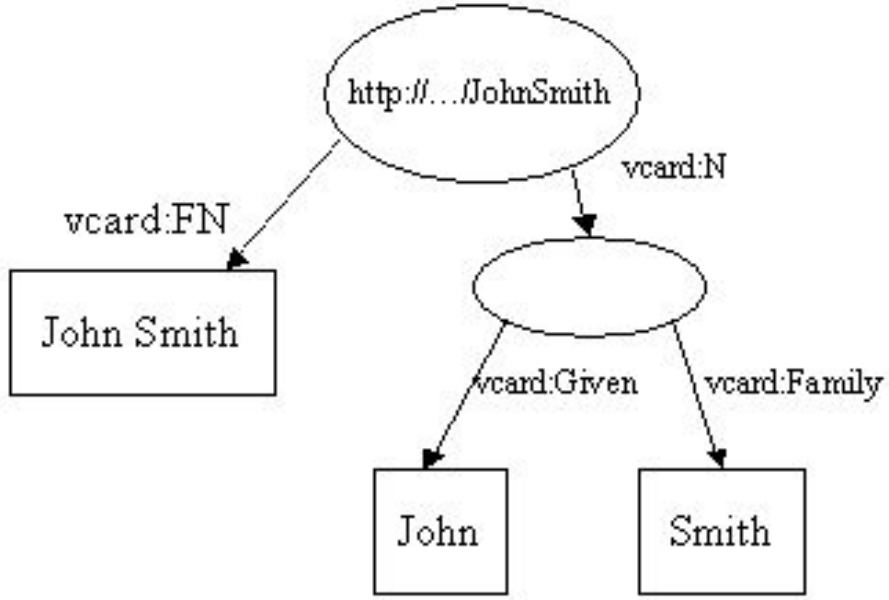
Cosa vedremo oggi

- Leggere e scrivere RDF con Apache Jena
- SPARQL - eseguire query su modelli RDF usando lo standard SPARQL
- Ontologie - supporto per gestire modelli in Jena
- Reasoners e rule engines: supporto per l'inferenza
  - Inferenza RDFS sui dati
- [TDB2 - a fast persistent triple store that stores directly to disk]
  - [TDB - Original TDB database]
- [Virtuoso]
  - https://vos.openlinksw.com/owiki/wiki/VOS/VirtJenaProvider
  - https://community.openlinksw.com/t/using-jena-framework-and-virtuoso-for-transactional-operations/3030

Jena core classes

`RDFNode` (interfaccia)

- rappresenta un nodo di un grafo RDF
- super-interfaccia di
  - `Resource` (interfaccia)
    - identificate da un URI o bNodes
  - `Literal`



Concetti del SW e relative classi Jena

ARTIFACT	SEMANTIC WEB	JENA JAVA CLASS	NOTES
Subject, predicate, object	URI	`Resource, Property`	A resource can be subject, object, or predicate
Statement	Statement	`Statement`	
Data	Ontology and instance data	`Graph` and `Model`	Graphs are a basic building block for models - e.g. common interface to low-level RDF stores. Models are the main container for RDF information presented in graph form, and are designed to have rich APIs. Several types of models exist, e.g. `Model` for RDF and `OntModel` for OWL.
Query and results	SPARQL and Semantic Web data	`Query` and `ResultSet`	
Reasoner	Reasoner	`Reasoner`	
Rules	SWRL	`Reasoner`	Rule support determined by specific reasoner

Leggere e scrivere RDF con Apache Jena

Ci sono due metodi principali per **leggere** dati RDF in Jena:

```
// 1. Using the RDFDataMgr
// Create a model and read into it from file "data.ttl" in Turtle format.
Model model = RDFDataMgr.loadModel("data.ttl", Lang.TURTLE) ;

// Read into an existing Model
RDFDataMgr.read(model, "data.ttl") ;

// 2. Using Model
Model model = ModelFactory.createDefaultModel() ;
model.read("data.ttl", "TURTLE") ;

OutputStream out = new FileOutputStream("data_new.ttl");
RDFDataMgr.write(out, model, Lang.TURTLE);

model.write(output, "TURTLE") ;
```

Leggere e scrivere RDF con Apache Jena

Ci sono due metodi principali per **scrivere** dati RDF in Jena:

```
// 1. Using the RDFDataMgr
OutputStream out = new FileOutputStream("data_new.ttl");
RDFDataMgr.write(out, model, Lang.TURTLE);

// 2. Using Model
model.write(output, "TURTLE") ;
```

`RDFFormat`s and Jena syntax names

The string name traditionally used in `model.write` is mapped to RIOT `RDFFormat` as follows:

Jena writer name	RIOT RDFFormat
`"TURTLE"`	`TURTLE`
`"TTL"`	`TURTLE`
`"Turtle"`	`TURTLE`
`"N-TRIPLES"`	`NTRIPLES`
`"N-TRIPLE"`	`NTRIPLES`
`"NT"`	`NTRIPLES`
`"JSON-LD"`	`JSONLD`
`"RDF/XML-ABBREV"`	`RDFXML`
`"RDF/XML"`	`RDFXML_PLAIN`
`"N3"`	`N3`
`"RDF/JSON"`	`RDFJSON`

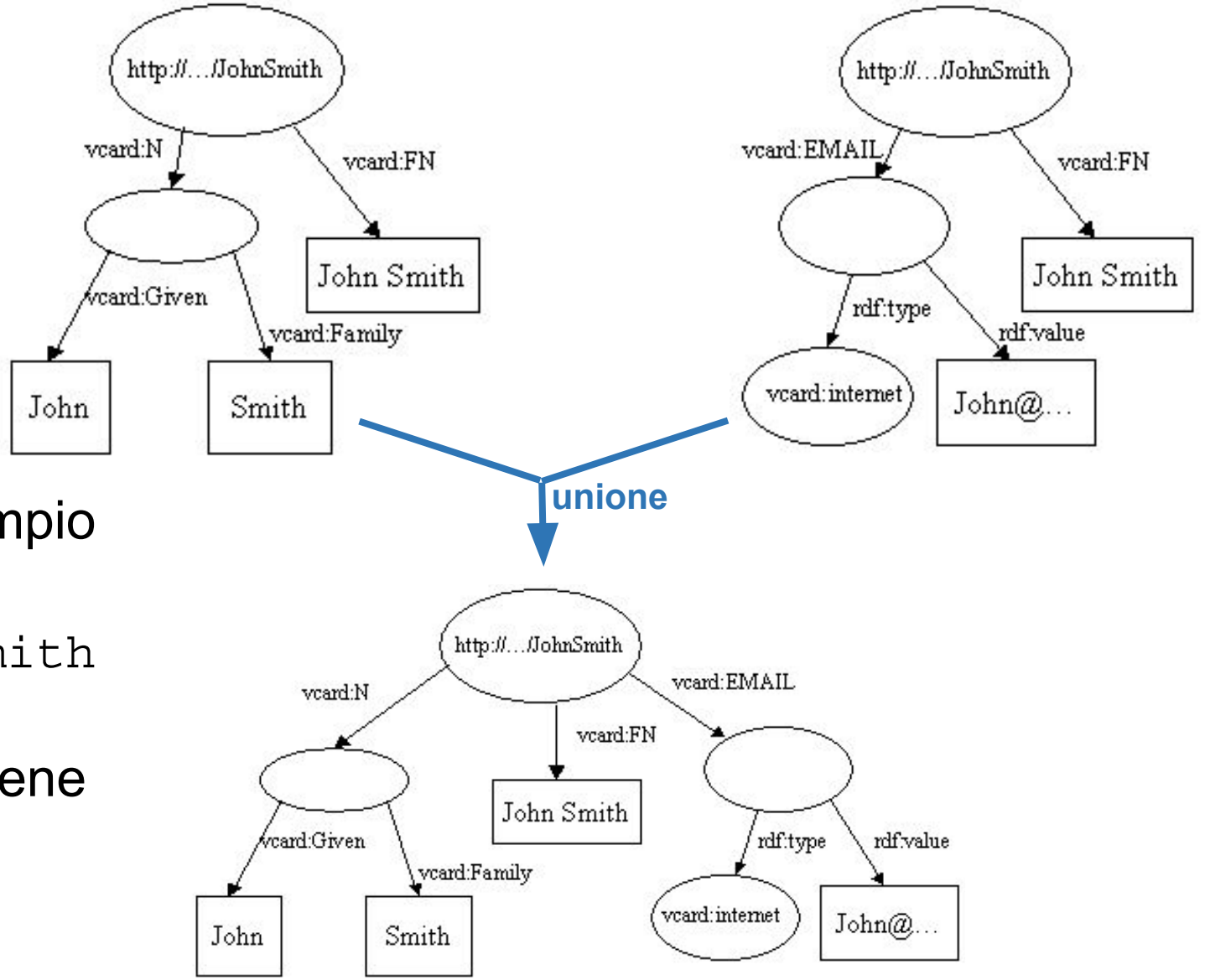
Operazioni sui modelli

Jena fornisce tre operazioni per manipolare i modelli nel loro complesso:

- unione
- intersezione
- differenza.

Esempio: l'unione di due modelli è l'unione degli insiemi di statement di ciascun Modello, e permette di unire dati provenienti da fonti diverse.

Operazioni sui modelli



Quando i due modelli di esempio vengono uniti:

- i due nodi `http://...JohnSmith` vengono uniti in uno
- l'arco `vcard:FN` duplicato viene eliminato

SPARQL in the Semantic Web stack

S **P** **A** **R** **Q** **L**

P rotocol

A nd

R DF

Q uery

L anguage

W3C Recommendation

W3C

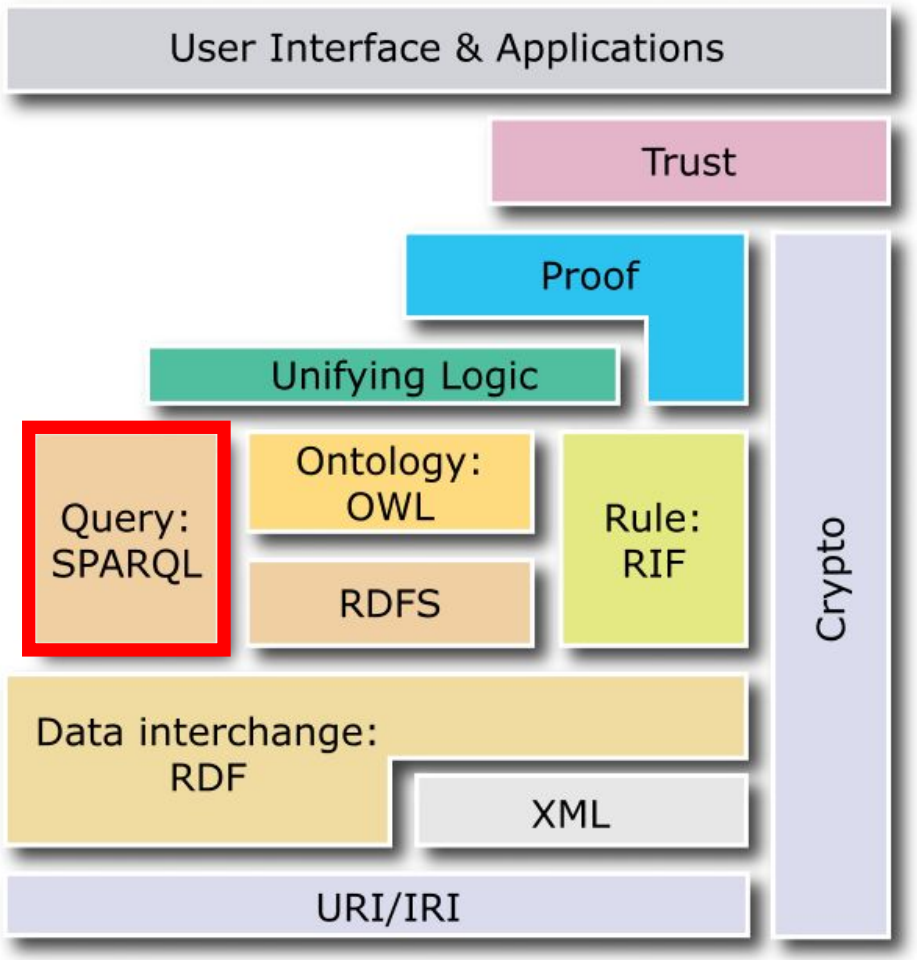
SPARQL 1.1 Query Language

W3C Recommendation 21 March 2013

This version:
http://www.w3.org/TR/2013/REC-sparql11-query-20130321/
Latest version:
http://www.w3.org/TR/sparql11-query/
Previous version:
http://www.w3.org/TR/2012/PR-sparql11-query-20121108/
Editors:
Steve Harris, Garlik, a part of Experian
Andy Seaborne, The Apache Software Foundation
Previous Editor:
Eric Prud'hommeaux, W3C

SPARQL

The standard language for querying and manipulating semantic knowledge graphs represented as RDF triples



Source: https://it.m.wikipedia.org/wiki/File:Semantic_Web_Stack.png

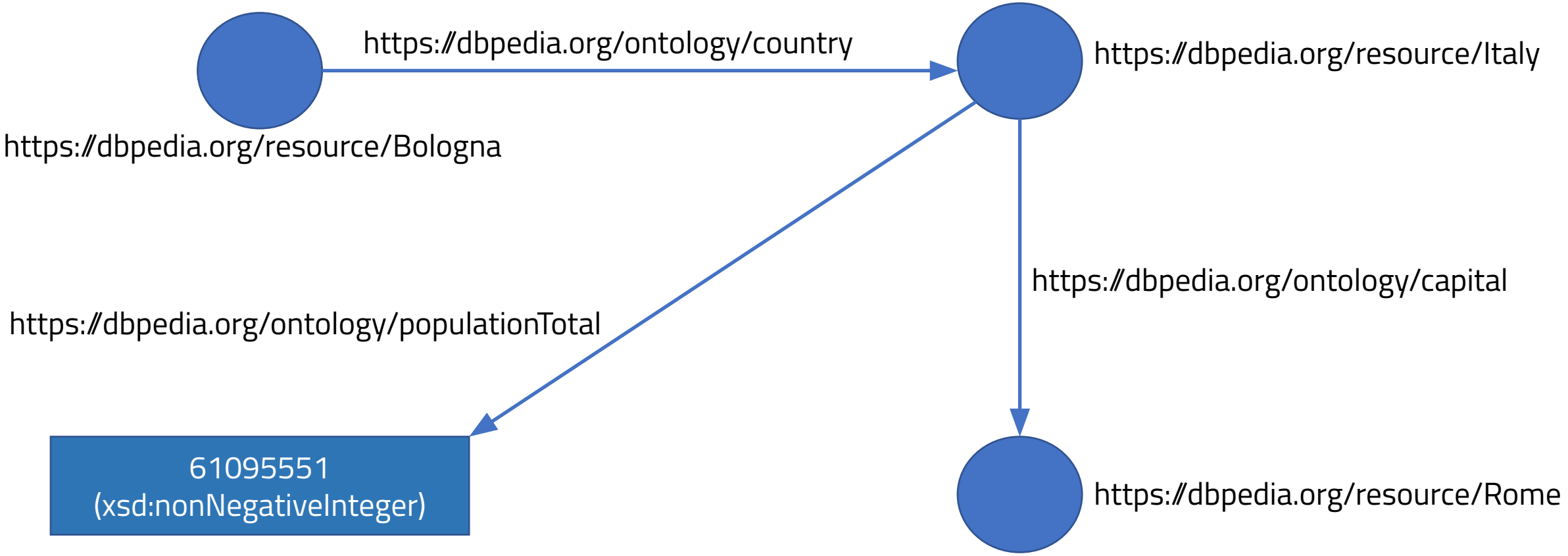
Resource Description Framework - defining triples

- Every entity in the data is univocally identified by a URI/IRI (Uniform Resource Identifier/Internationalized Resource Identifier)
- Data is expressed in the form of **triples**

`subject predicate object`

- `Subject` and `predicate` always have a URI
- `Object` can be:
  - the subject of another triple and thus identified by a URI
  - a literal value (e.g., a string, a date, a number)
- `Subject` and `Object` are nodes and `predicate`s are arcs connecting the nodes
- When an `Object` is the `Subject` of another triple (thus the triples share the same entity), triples are interconnected (interconnected graph of triples Linked Data)

Example of interconnected (knowledge) graph



THANK YOU!