Basi Di Dati e di conoscenza

Evoluzione dei linguaggi, dei modelli e dei sistemi per basi di dati

XML-Xpath-XQuery

Arricchimento semantico delle basi di dati

- Presente fin dagli inizi della evoluzione del settore (Raymond Abrial: modello semantico, 1974; Peter Chen, modello ER, 1975).
- Spinto dalla parallela evoluzione dei linguaggi a oggetti e dei linguaggi logici, verso la fine degli anni ottanta, e del «semantic web» (Tim Berners Lee, 1999).
- Molto successo dalla diffusione degli «open data».
- Linguaggi semantici per basi di dati usati in tre contesti:
  - Durante la progettazione concettuale
  - Per rappresentare la conoscenza ad un livello astratto, che poi viene tradotto in un linguaggio logico (tipicamente relazionale)
  - Direttamente in query languages che includono la semantica e hanno capacità inferenziale.

Modello RDF

- Proposto e standardizzato dal W3C
- Informazioni rappresentate sotto forma di triple, che possono essere interpretate come soggetto – verbo – oggetto:

  <verdi ha_composto traviata>
- Istanza RDF: insieme di triple, rappresentate come un grafo, in cui i nodi corrispondono ai soggetti e agli oggetti, gli archi ai verbi.
- E' possibile ma non necessario definire uno schema
- E' possibile utilizzare uno o più vocabolari per i termini usati nell'istanza RDF, introdotti dalla istruzione «prefix» che li associa ad un prefisso da usare nelle definizioni e query:

  @prefix o: <http://www.polimi.it/ceri/opera>.

Dati RDF

- I dati rappresentabili in RDF sono di tre categorie.
  - *IRI* (Internationalized Resource Identifier): stringhe che identificano univocamente tutte le risorse disponibili sul Web
  - *Letterali:* descrivono le risorse presenti nell'istanza RDF. E` pos-sibile assegnare ai letterali un tipo; se inclusi tra apici sono implicitamente di tipo stringa, mentre numeri senza apici, con segno, sono implicitamente di tipo intero.
  - *Blank Nodes:* identificatori usati per rappresentare risorse anonime
- Nelle triple RDF il primo termine (soggetto) puo` essere un IRI o un blank node, il secondo termine (predicato) e` un IRI, e il terzo termine (oggetto) puo` essere un IRI, un letterale o un blank node.

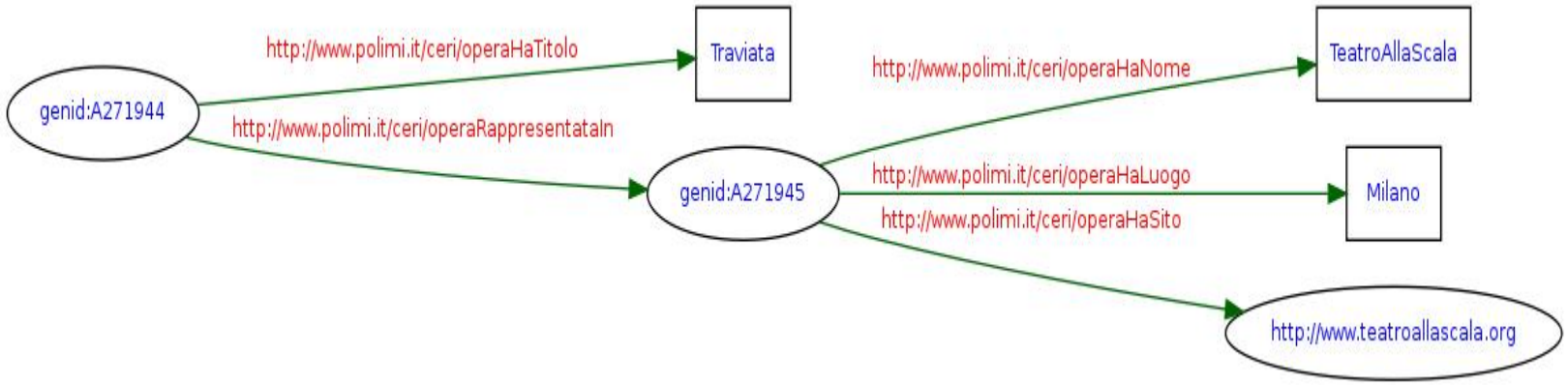
Esempio di istanza RDF

```
_:o1 o:HaTitolo 'Traviata'  ;
        o:RappresentataIn \_:p1  .
_:p1 o:HaNome 'TeatroAllaScala' ;
        o:HaLuogo 'Milano' ;
        o:HaSito http://www.teatroallascala.org .
```

- Viene adottata la notazione **N3**, che utilizza la punteggiatura per sintetizzare le parti comuni delle triple. Quando due triple condividono il soggetto si usa il separatore ';' quando condividono sia soggetto sia predicato, si usa il separatore ','.
- Oltre alla notazione N3, e` possibile utilizzare il formato XML per la definizione di una istanza RDF; tale notazione prende il nome di **RDF/XML** ed e` stata standardizzata dal W3C.

Rappresentazione a rete

```
_:o1 o:HaTitolo ’Traviata’  ;
        o:RappresentataIn \_:p1  .
_:p1 o:HaNome ’TeatroAllaScala’ ;
        o:HaLuogo ’Milano’ ;
        o:HaSito http://www.teatroallascala.org .
```



Rappresentazione XML

```
<?xml version="1.0"?>
<rdf:RDF xmlns:o="http://www.polimi.it/ceri/opera"
         xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:ceri="http://www.polimi.it/ceri/">
  <rdf:Description>
    <ceri:operaHaTitolo>Traviata</ceri:operaHaTitolo>
      <ceri:operaRappresentataIn>
        <rdf:Description>
          <ceri:operaHaNome>TeatroAllaScala</ceri:operaHaNome>
          <ceri:operaHaLuogo>Milano</ceri:operaHaLuogo>
          <ceri:operaHaSito rdf:resource="http://www.teatroallascala.org"/>
        </rdf:Description>
      </ceri:operaRappresentataIn>
  </rdf:Description>
</rdf:RDF>
```

Altro esempio di istanza RDF (1)

```
@prefix o: <http://www.polimi.it/ceri/opera>

_:Ring  o:compostoDa 'Wagner'  .
_:Ring  o:prodottoDa 'ScalaDiMilano'  .
_:Ring  o:condottoDa 'DanielBaremboim' .
_:Ring  o:messoInScenaDa 'GuyCassiers' .
_:Ring o:haTitolo  'Ring'.

_:Ring  o:comprende _:OrodelReno,
                    _:Valchiria,
                    _:Sigfrid,
                    _:Crepuscolo.

_:OroDelReno o:haTitolo  'Oro Del   Reno' ;
             o:dura 150  ;
             o:haPersonaggio 'Wotan'  .
```

Altro esempio di istanza RDF (2)

```
_:Valchiria o:haTitolo 'Valchiria' ;
            o:dura 310  ;
            o:haAtti 3  ;
            o:haPersonaggio 'Brunilde' ;
            o:haPersonaggio 'Wotan'  ;
            o:haPersonaggio 'Sieglinde' .

_:Sigfrido o:haTitolo 'Sigfrido' ;
           o:dura 320  ;
           o:haAtti 3  ;
           o:haPersonaggio 'Brunilde' ;
           o:haPersonagio 'Sigfrido' .
```

```
_:Crepuscolo o:haTitolo  'Crepuscolo degli
Dei' ;
             o:dura 360  ;
             o:haAtti 3  ;
             o:haPrologo 'si' ;
             o:haPersonaggio 'Brunilde' ;
             o:haPersonaggio 'Sigrfido';
             o:haPersonaggio 'Waltraute' .
```

RDF Schema

- Estensione di RDF come raccomandazione del W3C dal 2004;
- Aggiunge ad RDF costrutti per descrivere metadati, ovvero struttura e proprietà di istanze RDF.
- I principali costrutti messi a disposizione da RDFS sono le classi e le proprietà.
  - Una classe RDFS consente di definire insiemi di risorse di un'istanza RDF che hanno caratteristiche comuni.
  - Una proprietà RDFS consente invece di definire il soggetto e l'oggetto dei predicati in un'istanza RDF.

RDF Schema

- Esempio di classe:

```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix o: <http://www.polimi.it/ceri/opera> .
o:Compositore rdf:type rdfs:Class.
_:Wagner rdf:type o:Compositore.
```

- Esempio di proprietà:

```
o:nome a rdfs:Property;
        rdfs:domain o:Compositore;
        rdfs:range rdfs:Literal.
_:Ring a o:Opera.
o:compostoDa a rdfs:Property;
        rdfs:domain o:Opera;
        rdfs:range o:Compositore.
```

Altre estensioni semantiche

- È possibile definire in RDFS relazioni di generalizzazione tra classi.

  o:Artista rdf:type rdfs:Class.

  o:Compositore rdfs:subClassOf o:Artista.

  - La semantica di RDFS definisce l' ereditarietà: se i compositori sono artisti e gli artisti sono persone, si deduce che i compositori sono persone; se le persone possiedono come proprietà una città e una data di nascita, anche i direttori d'orchestra hanno queste proprietà.
- Una maggior ricchezza semantica rispetto ad RDFS è offerta dal Web Ontology Language (OWL), nelle sue versioni OWL Lite, OWL DL e OWL Full.
  - OWL Lite pone delle restrizioni sui costrutti OWL utilizzabili che in garantiscono un calcolo efficiente delle inferenze.
  - OWL DL estende OWL Lite garantendo solo che tutte le inferenze possono essere calcolate in un tempo finito
  - OWL Full fornisce il massimo dell'espressività senza garantire la computabilità completa.

SPARQL

- SPARQL (Simple Protocol and RDF Query Language) è un linguaggio di interrogazione simile ad SQL proposto dal W3C per interrogare dati descritti in RDF, ricevendo il risultato in un formato XML.
- Quattro tipi di query:
  - SELECT per interrogazioni classiche
  - DESCRIBE, per ottenere una descrizione delle risorse presenti presso un database RDF interrogabile in SPARQL (detto «endpoint»)
  - ASK, per sapere se specifici termini sono disponibili nell'endpoint
  - COSTRUCT, per costruire un nuovo grafo RDF a partire da una interrogazione.
- Due versioni del linguaggio
  - 1.0 (raccomandazione W3C del 2008);
  - 1.1 (nuova versione più potente del 2013).

Sintassi

Una query SPARQL 1.0 è composta da cinque parti:

- La clausola opzionale PREFIX per introdurre vocabolari.
- Il risultato prodotto dalla query (una delle quattro clausole SELECT, DESCRIBE, ASK e CONSTRUCT)
- Gli endpoint consultati, tramite le clausole FROM e FROM NAMED.
- La parte centrale della query, introdotta dalla clausola {\tt WHERE}
- Modificatori opzionali, che includono i costrutti ORDER BY, LIMIT e OFFSET

Triple e graph patterns

- Un **triple pattern** è una tripla nelle cui posizioni è possibile far comparire, in aggiunta a IRI, letterali e blank nodes, anche variabili, introdotte dal simbolo '?'; durante la valutazione delle query le variabili vengono «legate» (binding)
- Ad esempio:

  < ?o1 o:HaPersonaggio 'Sigfrido' >

  valutato sulla istanza RDF, la variabile o1? è legata ai blank node _:Sigfrido e _:Crepuscolo.
- Vari triple pattern possono essere combinati assieme, formando un **graph pattern**.

Query Sparql: 1

Estrarre i nomi delle opere che hanno Wotan come personaggio

```
PREFIX  o: <http://www.polimi.it/ceri/opera>
SELECT ?t
FROM  <http://www.polimi.it/ceri/ring.rdf>
WHERE {
        ?o  o:haPersonaggio 'Wotan'  .
        ?o  o:haTitolo ?t    .
        }
```

?t
'Oro del Reno'
'Valchiria'

Query Sparql: 2

Estrarre i titoli delle opere e i nomi dei personaggi che hanno gli stessi personaggi che compaiono in Valchiria

```
PREFIX o: <http://www.polimi.it/ceri/opera>
SELECT ?t, ?p
FROM <http://www.polimi.it/ceri/ring.rdf>
WHERE {
        ?o1 o:haPersonaggio ?p .
        ?o2 o:haPersonaggio ?p .
        ?o1 != ?o2 .
        ?o1 o:haTitolo ?t .
        ?o2 o:haTitolo 'Valchiria' .
        }
```

?t	?p
‘Oro del Reno’	‘Wotan’
‘Sigfrido’	‘Brunilde’
‘Crepuscolo degli Dei’	‘Brunilde’

Query Sparql: 3

Estrarre le coppie di opere che hanno un personaggio in comune

```
PREFIX o: <http://www.polimi.it/ceri/opera>
SELECT ?t1, ?t2, ?p
FROM <http://www.polimi.it/ceri/ring.rdf>
WHERE {
        ?o1 o:haPersonaggio ?p .
        ?o2 o:haPersonaggio ?p .
        ?o1 != o2 .
        ?o1 o:haTitolo ?t1 .
        ?o2 o:haTitolo ?t2 .
        }
```

Query Sparql: 4

Estrarre i titoli delle opere e i nomi dei personaggi che hanno gli stessi personaggi che compaiono in Valchiria e filtrare le opere la cui durata è superiore a 300 minuti

```
PREFIX o: <http://www.polimi.it/ceri/opera>
SELECT ?t1, ?t2, ?p
FROM <http://www.polimi.it/ceri/ring.rdf>
WHERE {
        ?o1 o:haPersonaggio ?p .
        ?o2 o:haPersonaggio ?p .
        ?o1 != o2 .
        ?o1 o:haTitolo ?t1 .
        ?o2 o:haTitolo ?t2  .
        ?o2 o:haDurata ?d .
        FILTER (?d > 300).
        }
```

Query Sparql: 5

Estrarre le e opere che hanno Wotan come personaggio oppure durano più di 350 minuti.

```
PREFIX o: <http://www.polimi.it/ceri/opera>
SELECT ?t
FROM <http://www.polimi.it/ceri/ring.rdf>
WHERE {
        { ?o o:haTitolo ?t .
        ?o o:haPersonaggio ?p .
                FILTER (?p = 'Wotan') .}
UNION
        { ?o o:haTitolo ?t .
        ?o o:dura ?m .
        FILTER (?m > 350). }
        }
```

Query Sparql: 6

Estrarre estrae titolo, durata, numero di atti e presenza di prologo di tutte le opere.

```
PREFIX o: <http://www.polimi.it/ceri/opera>
SELECT ?t, ?d, ?a, ?p
FROM <http://www.polimi.it/ceri/ring.rdf>
WHERE {
        ?o o:haTitolo ?t .
        ?o o:haDurata ?d .
        OPTIONAL {
                ?o o:haAtti ?a .
                ?o o:haPrologo ?p. }
}
```

Query Sparql: 7

Estrarre il titolo di tutte le opere che hanno qualche personaggio diverso da Sigfrido

```
PREFIX o: <http://www.polimi.it/ceri/opera>
SELECT ?t
FROM <http://www.polimi.it/ceri/ring.rdf>
WHERE {
        ?o o:haTitolo ?t
        ?o o:haPersonaggio ?p
        FILTER {
                ?p != 'Sigfrido'}
        }
```

Query Sparql: 8

Estrarre le opere in cui non è presente il personaggio Sigfrido

```
PREFIX o: <http://www.polimi.it/ceri/opera>
SELECT ?t
FROM <http://www.polimi.it/ceri/ring.rdf>
WHERE {
        ?o o:haTitolo ?t
        OPTIONAL {
                ?o haPersonaggio ?p .
                FILTER (?p = 'Sigfrido')}
        FILTER (! BOUND(?p))
        }
```

Query Sparql: Costruct

Costruire nuove tuple RDF a partire dall'istanza RDF e costruendo legami per le variabili ?t e ?p, in particolare le coppie il cui il primo elemento è il nome Ring e il cui secondo elemento è un personaggio di una delle quattro opere che lo compongono.

```
PREFIX o: <http://www.polimi.it/ceri/opera>
CONSTRUCT { ?t haPersonaggio ?p . }
FROM <http://www.polimi.it/ceri/ring.rdf>
WHERE {
        ?r o:haTitolo ?t .
        ?r o:comprende ?o .
        ?o o:haPersonaggio ?p .
        }
```

Query Sparql: Ask

valuta se una query ha un risultato non nullo, cioè se alle variabili della query viene associata almeno una tupla di binding. La query ASK ha valore **false**.

```
PREFIX o: <http://www.polimi.it/ceri/opera>
ASK
FROM <http://www.polimi.it/ceri/ring.rdf>
WHERE {
        ?o o:haTitolo 'Walkiria' .
        ?o o:haPrologo ?p .
        }
```

Query Sparql: Describe

Estrae tutta l'informazione conosciuta relativamente alle risorse che soddisfano una query, in una forma definita dall'endpoint SPARQL.

```
PREFIX o: <http://www.polimi.it/ceri/opera>
DESCRIBE ?o
FROM <http://www.polimi.it/ceri/ring.rdf>
WHERE {
        ?o o:haTitolo 'Walkiria' .
        }
```

SPARQL 1.1

- Introduce le clausole GROUP BY e HAVING
- Introduce l'operatore binario MINUS, che completa la copertura dell'algebra relazionale, e la clausola NOT EXISTS, che ricorda le sottoquery di SQL.
- Introduce anche alcuni aspetti che estendono in modo significativo e caratteristico il potere espressivo del linguaggio, tra cui le nozioni di **entailment**, di **property path e** l'operatore SERVICE, che consente forme sofisticate di interoperabilità.

Aggregazione in SPARQL 1.1

Calcolare la durata totale delle quattro opere che formano il Ring

```
PREFIX o: <http://www.polimi.it/ceri/opera>
SELECT ((SUM ?d) AS ?DurataDelRing)
FROM <http://www.polimi.it/ceri/ring.rdf>
WHERE {
        ?r o:haTitolo 'Ring' .
        ?r o:comprende ?o .
        ?o o:haDurata ?d .
         }
GROUP BY ?r
```

Negazione in SPARQL 1.1

Estrarre il titolo di tutte le opere che hanno qualche personaggio diverso da Sigfrido (si danno due soluzioni possibili in Sparql 1.1.; vedi anche query Q8).

```
PREFIX o: <http://www.polimi.it/ceri/opera>
SELECT ?t
FROM <http://www.polimi.it/ceri/ring.rdf>
WHERE {
        ?o o:haPersonaggio ?p .
        ?o o:haTitolo ?t .}
MINUS {
        ?o haPersonaggio 'Sigfrido' .
        ?o haTitolo ?t . } }
```

```
PREFIX o: <http://www.polimi.it/ceri/opera>
SELECT ?t
FROM <http://www.polimi.it/ceri/ring.rdf>
WHERE {
        ?o o:haTitolo ?t .
        FILTER (NOT EXISTS ?o haPersonaggio 'Sigfrido')}
```

Sottoquery in SPARQL 1.1

Calcolare quanti sono i personaggi che compaiono pi\`u di due volte nell'istanza RDF di Figura \ref{ring}, si noti la nidificando due query aggregate.

```
PREFIX o: <http://www.polimi.it/ceri/opera>
SELECT ((sum ?p) AS ?n)
FROM <http://www.polimi.it/ceri/ring.rdf>
WHERE
        { SELECT ?p
        WHERE { ?r o:haTitolo 'Ring' .
                ?r o:comprende ?o .
                ?o o:haPersonaggio ?p . }
        GROUP BY ?p
        HAVING ( COUNT(?o) > 2 ) }
```

Entailment regimes in SPARQL 1.1

Estendono la valutazione delle query includendo nel risultato tutte le
triple che sono implicate dalla istanza in base agli entailments RDFS.

Se è definito:

_:Wagner a o:Compositore.

o:Compositore rdfs:subClassOf o:Artista.

o:Artista rdfs:subClassOf o:Persona.

La query che segue restituisce «Richard Wagner»:

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX o: <http://www.polimi.it/ceri/opera>
SELECT ?n
FROM <http://www.polimi.it/ceri/ring.rdf>
WHERE
      { ?p o:nome ?n .
      ?p rdf:type o:Persona . }
```

Property Path in SPARQL 1.1

Estende SPARQL per la gestione della ricorsione,. Supponendo che lo schema RDF comprenda un predicato **amico**, che collega fra loro blank nodes corrispondenti a persone, e supponendo che tali blank nodes abbiano una relazione **haNome**, l'interrogazione percorre transitivamente la relazione a partire dalla persona di nome Giorgio.

```
SELECT ?n
WHERE
      { ?x haNome ?n .
        ?x (amico)* ?y .
        ?y haNome 'Giorgio' . }
```

Sparql e interoperabilità

Tramite istruzioni PREFIX è possibile importare definizioni di standard, vocabolari e ontologie

PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX dbcat: <http://dbpedia.org/resource/Category:>
PREFIX dbpprop: <http://dbpedia.org/property/>
PREFIX skos: <http://www.w3.org/2004/02/skos/core#>
PREFIX geo: <http://www.w3.org/2003/01/geo/wgs84_pos#>
PREFIX yago: <http://dbpedia.org/class/yago/>

Sparql e interoperabilità, 2

- È possibile importare i dati da istanze RDF, tramite la clausola FROM NAMED
- La clausola GRAPH consente di valutare la query su una particolare istanza RDF, oppure di determinare le istanze RDF che producono ciascun binding.

```
SELECT ?g, ?n
FROM NAMED <http://www.deib.polimi.it/reports>
FROM NAMED <http://www.dia.uniroma3.it/reports>
FROM NAMED <http://www.ingegneria.unibg.it/reports >
WHERE {
        GRAPH ?g
        { ?r scrittoDa ?p .
          ?p haNome ?n .
          ?r haTitolo 'Genomic Data Management' . } }
```

Sparql e interoperabilità, 3

La keyword SERVICE indica che una sotto-query va rivolta ad uno SPARQL endpoint remoto

```
SELECT ?n
FROM NAMED <http://www.deib.polimi.it/reports>
WHERE {
        ?r scrittoDa ?p .
        ?p haNome ?n .
        ?r haTitolo 'Genomic Data Management' .
        SERVICE <http://www.dia.uniroma3.it/reports>
                { ?p1 haNome ?n .
                  ?r scrittoDa ?p1 .
                  ?r haTitolo 'Data Mapping' . }
        }
```

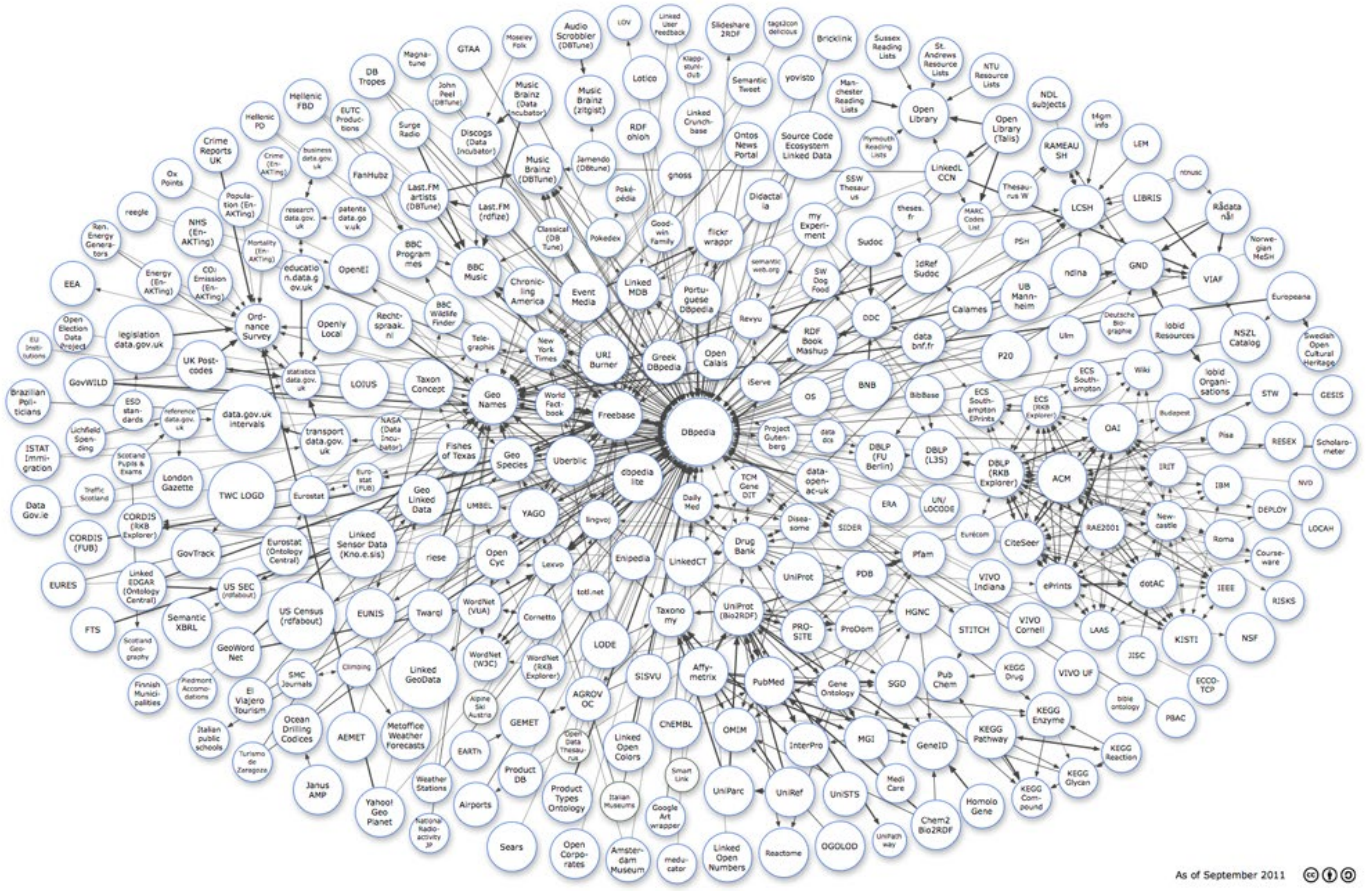
Linked e Open Data

- I Linked Data sono risorse disponibili sul Web, descritte tramite triple RDF e collegate fra loro tramite riferimenti (link).

- Pubblicare risorse sotto forma di Linked Data vuol dire aderire ad una buona pratica di pubblicazione, usando le IRI per identificare le risorse, lo standard HTTP per trasferire dati tra diversi nodi del Web, e gli standard RDF, RDFS e OWL per descrivere le triple.

- I linked data sono elencati sul sito www.linkeddata.org (la rappresentazione nella prossima slide è del 2011)

Linked e Open Data



Alcune risorse dei linked data

- Risorse interdominio: DBPedia (un dataset periodicamente estratto in modo automatico dai nodi di WikiPedia) Freebase, Yago e OpenCyc.
- Dati di tipo geografico: Geonames (vari milioni di nomi di luoghi) e LinkedGeoData (derivato dal progetto OpenStreetMap).
- Dati relativi ai media: BBC, New York Times e la Reuters.
- Dati relativi all'istruzione e alla ricerca: American Library of Congress, Open Library, DBLP
- Dati relativi alle scienze della vita: Gene Ontology (che descrive i geni), UniProt (che descrive le proteine), Kegg (la enciclopedia di Kyoto dei geni e dei genomi) e PubMed (che descrive le pubblicazioni scientifiche relative alla medicina).
- Dati relativii al commercio: GoodRelations (che descrive i vari aspetti del commercio elettronico)

Open data

- Sono dati amministrativi resi di dominio pubblico per aumentare la trasparenza delle amministrazioni nei riguardi dei cittadini.
- Tra i primi a pubblicare open data, gli Stati Uniti (www.data.gov) nel 2009, sotto la spinta dell'amministrazione Obama, e la Gran Bretagna (www.data.gov.uk).
- Il fenomeno della pubblicazione dei dati aperti, talvolta non connessi alla rete dei linked data, è in continua evoluzione, e citiamo ad esempio i siti dei comuni di Milano (www.dati.comune.milano.it/) e di Roma (www.dati.comune.roma.it/).