UNIVERSITÀ DEGLI STUDI FIRENZE | DINFO DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE | DISIT DISTRIBUTED SYSTEMS AND INTERNET TECHNOLOGIES LAB
http://www.disit.org

UNIVERSITÀ DEGLI STUDI FIRENZE MABIDA

Big Data Tools: RDF

Pierfrancesco Bellini

University of Florence, Department of Information Engineering,

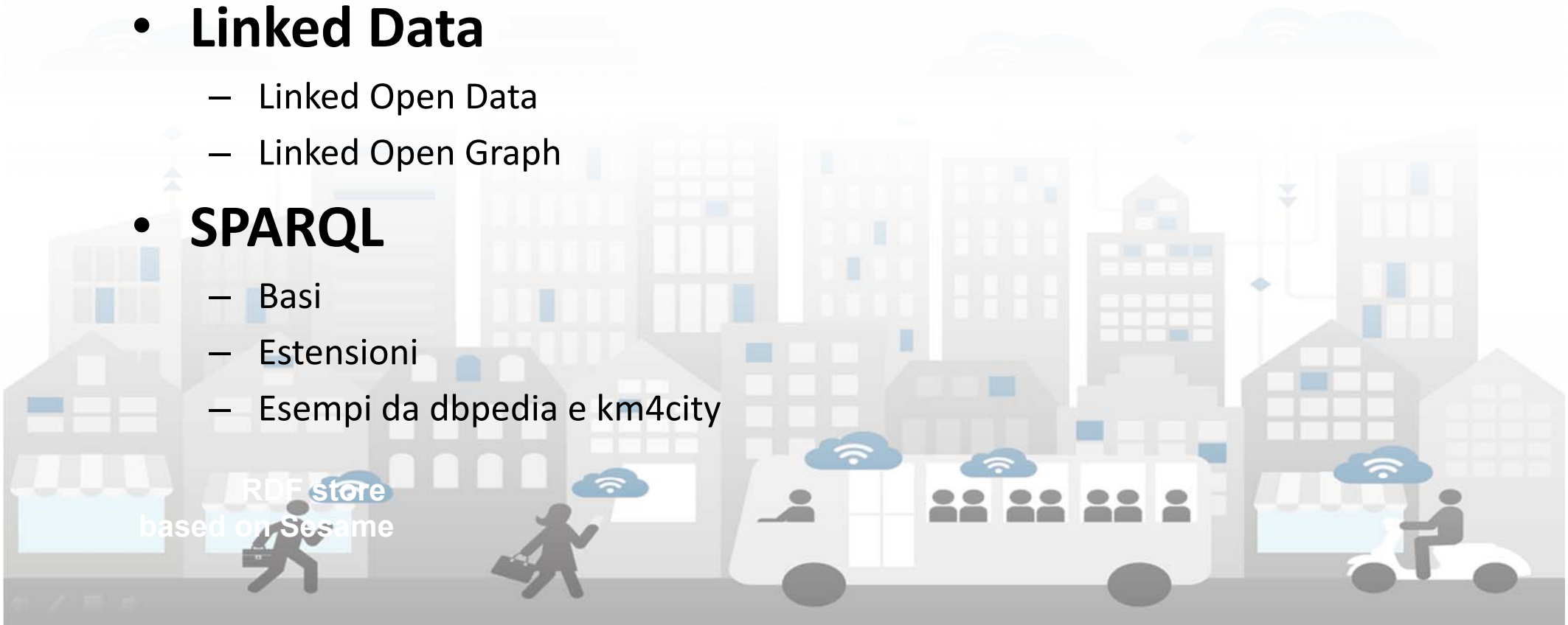
DISIT Lab, http://www.disit.org, http://www.sii-mobility.org,

pierfrancesco.bellini@unifi.it

MASTER BIG DATA, ANALYTICS AND TECHNOLOGIES FOR MANAGEMENT

Outline

- **RDF Stores**
  - Caratteristiche
  - Inferenza
  - Big RDF stores
- **Linked Data**
  - Linked Open Data
  - Linked Open Graph
- **SPARQL**
  - Basi
  - Estensioni
  - Esempi da dbpedia e km4city



NoSQL Databases

- **key-value databases**
  - *es: berkleyDB, memcacheDB, ...*
- **document databases**
  - *es: SOLR, MongoDB, ...*
- **wide-column databases**
  - *es: BigTable, Hbase, Cassandra*
- **graph databases**
  - *es: Neo4J, ...*
  - **RDF databases**
    - ***es: virtuoso, GraphDB, Blazegraph,***

RDF Database

- Un tipo particolare di NoSQL database
  - unico che ha un **modello dati standardizzato**
  - unico con **formati di interscambio standard** (RDFXML, Turtle, Ntriple)
  - Unico con un **linguaggio di interrogazione standard SPARQL** ed anche un protocollo di comunicazione standard
  - unico con anche **linguaggio di manipolazione standard SPARUL**
- Si possono avere diverse implementazioni dello standard che garantisce:
  - interoperabilità
  - evita il vendor lock-in

RDF Database

- a.k.a RDF Store
- Tool che permette la memorizzazione di una grande quantità di "triple" <s,p,o>
- Query tramite linguaggio SPARQL
- Manipolazione tramite SPARUL
- **Schemaless**
  - i dati non necessariamente sono strutturati in uno schema prefissato
  - Anche l'ontologia che descrive i dati è memorizzata con triple e viene usata per inferire nuove triple più che per garantire uno schema

Triple o Quadruple

- triple possono essere raggruppate in grafi identificati da una URI
- <subject> <predicate> <object> **<context>**
- Il context/grafo indica il contesto in cui la tripla è vera, anche il contesto è identificato da una URI e si possono avere altre triple dove il soggetto è il contesto.
- Un uso tipico è per poter associare dei metadati a un insieme di triple
  - es. associare a un dataset dati di provenienza e licenza uso

Triple o Quadruple

- Quadruple si possono usare:
  - per indicare che un certo fatto espresso da una tripla è vero in un certo contesto temporale
    - John hasWife Jane (from 1980 to 2010)
    - John hasWife Jody (from 2013)
  - Per raggruppare un insieme di triple che si vogliono facilmente eliminare dallo store tutte insieme, specialmente se i legami sono abbastanza articolati o usa blank nodes

RDF Store

- Possono permettere anche altri tipi di indicizzazione (non standard):
  - testo,
  - geografica,
  - temporale
- Ogni RDF store definisce un proprio dialetto del linguaggio SPARQL per usare questi indici.

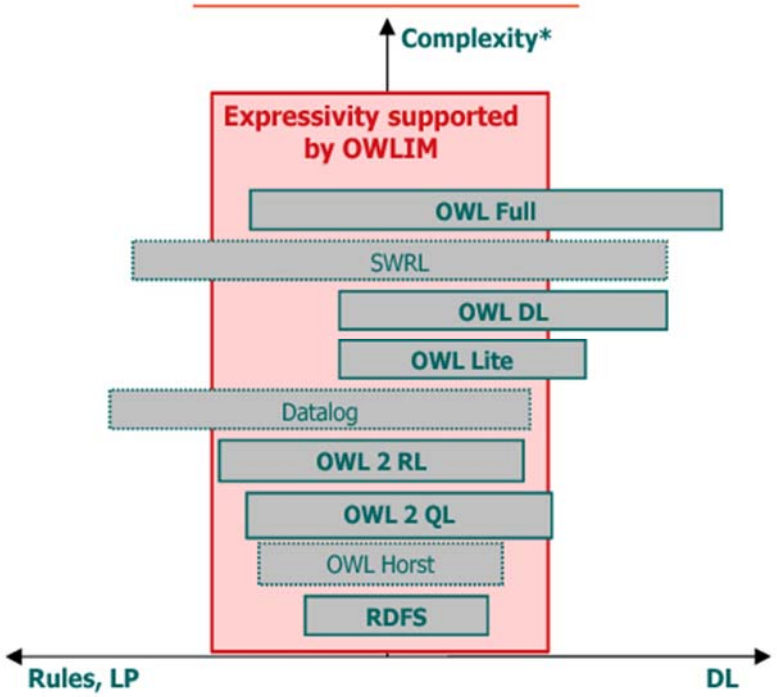
RDF Store - inferenza

- Gli RDF store possono o meno supportare una forma di inferenza sulle triple inserite
- Si hanno due strategie:
  - calcolare inferenza all'inserimento delle triple
  - eseguire inferenza al momento della query
- Nel primo caso
  - il grosso problema è mantenere la coerenza (truth maintenence) nel caso di eliminazione di triple.
  - le relazioni di equivalenza o transitive generano moltissime triple che possono appesantire tutte le query
- Nel secondo caso
  - La query è più complicata da eseguire e tipicamente meno espressiva

Esempio di Inferenza con OWLIM/GraphDB

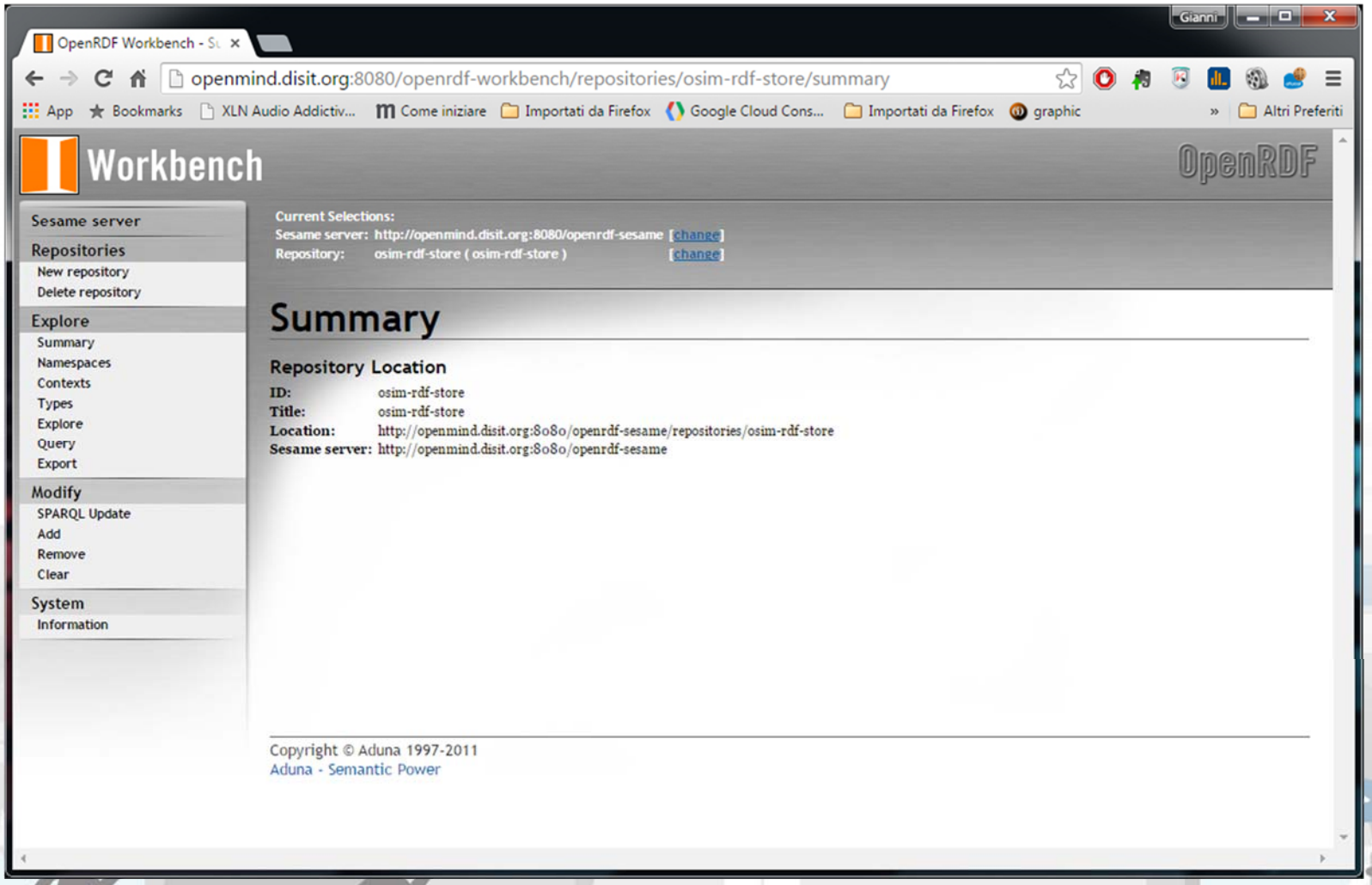
- I motori di reasoning utilizzano prevalentemente un tipo di inferenza detta Rule Based, che prevede 2 approcci:
  - **Forward-chaining**. Approccio data-driven che esegue inferenza sulla conoscenza esplicita, direttamente su tutte le triple senza gestire eventuali inconsistenze.
  - **Backward-chaining**. Approccio goal-driven che esegue inferenza con l'obiettivo di provare dei fatti (risultanti da una query di input).

- Il linguaggio OWL / OWL2 prevede vari livelli di espressività
- GraphDB utilizza prevalentemente l'approccio Forward-chaining, eseguendo l'inferenza durante la fase di ***indicizzazione***.
- 4 Diversi set di regole:
  - ***owl-max*** (livello più espressivo)
  - ***owl-horst***
  - ***rdfs*** – semantica standard RDF(S)
  - ***empty*** – no inferenza;



Esempio di Inferenza con Owlim/GraphDB

- Esempio di Inferenza su RDF Store Sesame (*osim-rdf-store*) + OWLIM



Esempio di Inferenza con Owlim/GraphDB

- Esempio di inferenza: *subClassOf*

Explore (<urn:u-gov:unifi:AC_AB0:8cf8e70205520a44e90211a34e6b7a9e>)

The results shown maybe truncated.

Subject	Predicate	Object	Context
<urn:u-gov:unifi:AC_AB0:8cf8e70205520a44e90211a34e6b7a9e>	rdf:type	i.o:Person	<file://C:/fakepath/OntoAnagraficaUnifi.nt>
<urn:u-gov:unifi:AC_AB0:8cf8e70205520a44e90211a34e6b7a9e>	rdf:type	i.o:Person	<file://C:/fakepath/CinecaRegistered.nt>
<urn:u-gov:unifi:AC_AB0:8cf8e70205520a44e90211a34e6b7a9e>	rdf:type	i.o:Person	<file://C:/fakepath/Collaborations.nt>
<urn:u-gov:unifi:AC_AB0:8cf8e70205520a44e90211a34e6b7a9e>	rdf:type	i.o:Person	<file://C:/fakepath/dumpo1.xml>
<urn:u-gov:unifi:AC_AB0:8cf8e70205520a44e90211a34e6b7a9e>	rdf:type	owl:NamedIndividual	<file://C:/fakepath/unifiCF.owl>
<urn:u-gov:unifi:AC_AB0:8cf8e70205520a44e90211a34e6b7a9e>	rdf:type	CMSAteneoCompetence:Professor	<file://C:/fakepath/unifiCF.owl>
<urn:u-gov:unifi:AC_AB0:8cf8e70205520a44e90211a34e6b7a9e>	rdf:type	CMSAteneoCompetence:Professor	<file://C:/fakepath/dumpo1.xml>
<urn:u-gov:unifi:AC_AB0:8cf8e70205520a44e90211a34e6b7a9e>	rdf:type	CMSAteneoCompetence:Teacher	
<urn:u-gov:unifi:AC_AB0:8cf8e70205520a44e90211a34e6b7a9e>	rdfs:label	"Paolo Nesi"	<file://C:/fakepath/OntoAnagraficaUnifi.nt>
<urn:u-gov:unifi:AC_AB0:8cf8e70205520a44e90211a34e6b7a9e>	rdfs:label	"paolo nesi"^^xsd:string	<file://C:/fakepath/unifiCF.owl>
<urn:u-gov:unifi:AC_AB0:8cf8e70205520a44e90211a34e6b7a9e>	i.o:name	"Paolo Nesi"	<file://C:/fakepath/OntoAnagraficaUnifi.nt>
<urn:u-gov:unifi:AC_AB0:8cf8e70205520a44e90211a34e6b7a9e>	i.o:name	"Paolo Nesi"	<file://C:/fakepath/dumpo1.xml>
<urn:u-gov:unifi:AC_AB0:8cf8e70205520a44e90211a34e6b7a9e>	CMSAteneoCompetence:haSSD	CMSAteneoCompetence:ING-INF_05	
<urn:u-gov:unifi:AC_AB0:8cf8e70205520a44e90211a34e6b7a9e>	CMSAteneoCompetence:haSSD	CMSAteneoCompetence:ING-INF_05	<file://C:/fakepath/OntoAnagraficaUnifi.nt>
<urn:u-gov:unifi:AC_AB0:8cf8e70205520a44e90211a34e6b7a9e>	CMSAteneoCompetence:haSSD	CMSAteneoCompetence:K05A	<file://C:/fakepath/OntoAnagraficaUnifi.nt>
<urn:u-gov:unifi:AC_AB0:8cf8e70205520a44e90211a34e6b7a9e>	CMSAteneoCompetence:isAffiliatedOf	<http://www.unifi.it/cercachi/show.php?f=s&codice=051400&fonte=dipartimento%20di%20sistemi%20e%20informatica>	<file://C:/fakepath/unifiCF.owl>
<urn:u-gov:unifi:AC_AB0:8cf8e70205520a44e90211a34e6b7a9e>	CMSAteneoCompetence:isAffiliatedOf	<http://www.disit.dsi.unifi.it/>	
<urn:u-gov:unifi:AC_AB0:8cf8e70205520a44e90211a34e6b7a9e>	CMSAteneoCompetence:isWorkingFor	<http://www.disit.dsi.unifi.it/>	<file://C:/fakepath/unifiCF.owl>

```
PREFIX uni:http://www.dsi.unifi.it/CMSAteneoCompetence#
<owl:Class rdf:about="uni:Professor">
        <rdfs:label xml:lang="en">full professor</rdfs:label>
        <rdfs:label xml:lang="it">professore ordinario</rdfs:label>
        <rdfs:subClassOf:resource="uni:Teacher"/>
</owl:Class>
```

Esempio di Inferenza con Owlim/GraphDB

- Esempio di inferenza: *subPropertyOf*

Explore (<urn:u-gov:unifi:AC_AB0:8cf8e70205520a44e90211a34e6b7a9e>)

The results shown maybe truncated.

Subject	Predicate	Object	Context
<urn:u-gov:unifi:AC_AB0:8cf8e70205520a44e90211a34e6b7a9e>	rdf:type	i.o:Person	<file://C:/fakepath/OntoAnagraficaUnifi.nt>
<urn:u-gov:unifi:AC_AB0:8cf8e70205520a44e90211a34e6b7a9e>	rdf:type	i.o:Person	<file://C:/fakepath/CinecaRegistered.nt>
<urn:u-gov:unifi:AC_AB0:8cf8e70205520a44e90211a34e6b7a9e>	rdf:type	i.o:Person	<file://C:/fakepath/Collaborations.nt>
<urn:u-gov:unifi:AC_AB0:8cf8e70205520a44e90211a34e6b7a9e>	rdf:type	i.o:Person	<file://C:/fakepath/dumpo1.xml>
<urn:u-gov:unifi:AC_AB0:8cf8e70205520a44e90211a34e6b7a9e>	rdf:type	owl:NamedIndividual	<file://C:/fakepath/unifiCF.owl>
<urn:u-gov:unifi:AC_AB0:8cf8e70205520a44e90211a34e6b7a9e>	rdf:type	CMSAteneoCompetence:Professor	<file://C:/fakepath/unifiCF.owl>
<urn:u-gov:unifi:AC_AB0:8cf8e70205520a44e90211a34e6b7a9e>	rdf:type	CMSAteneoCompetence:Professor	<file://C:/fakepath/dumpo1.xml>
<urn:u-gov:unifi:AC_AB0:8cf8e70205520a44e90211a34e6b7a9e>	rdf:type	CMSAteneoCompetence:Teacher	
<urn:u-gov:unifi:AC_AB0:8cf8e70205520a44e90211a34e6b7a9e>	rdfs:label	"Paolo Nesi"	<file://C:/fakepath/OntoAnagraficaUnifi.nt>
<urn:u-gov:unifi:AC_AB0:8cf8e70205520a44e90211a34e6b7a9e>	rdfs:label	"paolo nesi"^^xsd:string	<file://C:/fakepath/unifiCF.owl>
<urn:u-gov:unifi:AC_AB0:8cf8e70205520a44e90211a34e6b7a9e>	i.o:name	"Paolo Nesi"	<file://C:/fakepath/OntoAnagraficaUnifi.nt>
<urn:u-gov:unifi:AC_AB0:8cf8e70205520a44e90211a34e6b7a9e>	i.o:name	"Paolo Nesi"	<file://C:/fakepath/dumpo1.xml>
<urn:u-gov:unifi:AC_AB0:8cf8e70205520a44e90211a34e6b7a9e>	CMSAteneoCompetence:haSSD	CMSAteneoCompetence:ING-INF_05	
<urn:u-gov:unifi:AC_AB0:8cf8e70205520a44e90211a34e6b7a9e>	CMSAteneoCompetence:haSSD	CMSAteneoCompetence:ING-INF_05	<file://C:/fakepath/OntoAnagraficaUnifi.nt>
<urn:u-gov:unifi:AC_AB0:8cf8e70205520a44e90211a34e6b7a9e>	CMSAteneoCompetence:haSSD	CMSAteneoCompetence:K05A	<file://C:/fakepath/OntoAnagraficaUnifi.nt>
<urn:u-gov:unifi:AC_AB0:8cf8e70205520a44e90211a34e6b7a9e>	CMSAteneoCompetence:isAffiliatedOf	<http://www.unifi.it/cercachi/show.php?f=s&codice=051400&fonte=dipartimento%20di%20sistemi%20e%20informatica>	<file://C:/fakepath/unifiCF.owl>
<urn:u-gov:unifi:AC_AB0:8cf8e70205520a44e90211a34e6b7a9e>	CMSAteneoCompetence:isAffiliatedOf	<http://www.disit.dsi.unifi.it/>	
<urn:u-gov:unifi:AC_AB0:8cf8e70205520a44e90211a34e6b7a9e>	CMSAteneoCompetence:isWorkingFor	<http://www.disit.dsi.unifi.it/>	<file://C:/fakepath/unifiCF.owl>

```
PREFIX uni:http://www.dsi.unifi.it/CMSAteneoCompetence#
<owl:ObjectProperty rdf:about="uni:isWorkingFor">
        <rdfs:range rdf:resource="uni:Laboratory"/>
        <owl:inverseOf rdf:resource="uni:hasWorker"/>
        <rdfs:subPropertyOf rdf:resource="uni:isAffiliatedOf"/>
</owl:ObjectProperty>
```

Inferenza con Virtuoso

- Virtuoso è un RDF Store e un DBMS SQL molto efficiente (scritto in C) che trasforma le query in SQL
- Inferenza viene eseguita al momento della **query**
- La query viene trasformata sulla base dell'ontologia (aumenta il tempo di query ma query fatta su meno dati)
- *Meno espressivo, supporta RDFS e alcune cose di OWL*

Big RDF store

- Alcuni degli RDF Store supportano la memorizzazione di diversi **milioni di triple** fino a **miliardi di triple** con performance di interrogazione ancora discrete.
- Le strategie sono:
  - **comprimere il più possibile** in modo da poter tenere il più possibile dati in memoria e su un unico computer
  - **usare indici** basati su alberi B o hash per accedere velocemente ai dati su disco
  - **partizionare i dati** tra più macchine
  - **dividere la query** in più parti da eseguire in parallelo

BIG RDF Stores

- **AllegroGraph** 1 trilione di triple (240 CPU, 1.28 TB RAM)
- **Oracle** 1 trilione di triple (8 nodi)
- **Virtuoso** 150 miliardi di triple (cluster 8 nodi)
- **Blazegraph** 50 miliardi di triple su un singolo nodo
- **GraphDB** 10 miliardi di triple su singolo nodo
- Benchmarks per valutazione prestazioni, dataset sintetico+query
  - LUBM - **Lehigh University Benchmark** (università)
  - BSBM - **Berlin SPARQL Benchmark** (e-commerce)
  - LDBC - **Linked Data Benchmark Council** (social media, semantic publishing)
  - ...

Linked Data

- Entità rappresentate da delle URL
  - es: http://dbpedia.org/resource/Florence
- Tramite protocollo HTTP si può ottenere dalla URL una descrizione della entità in forma:
  - *human readable* (HTML)
  - oppure *machine readable*
- La descrizione *machine readable* della entità viene data usando RDF/XML o simili
- Si hanno link verso altre entità gestite da altri
  - es:
    http://dbpedia.org/resource/Florence **owl:sameAs**
    http://sws.geonames.org/3176959/

Linked Open Data (LOD)

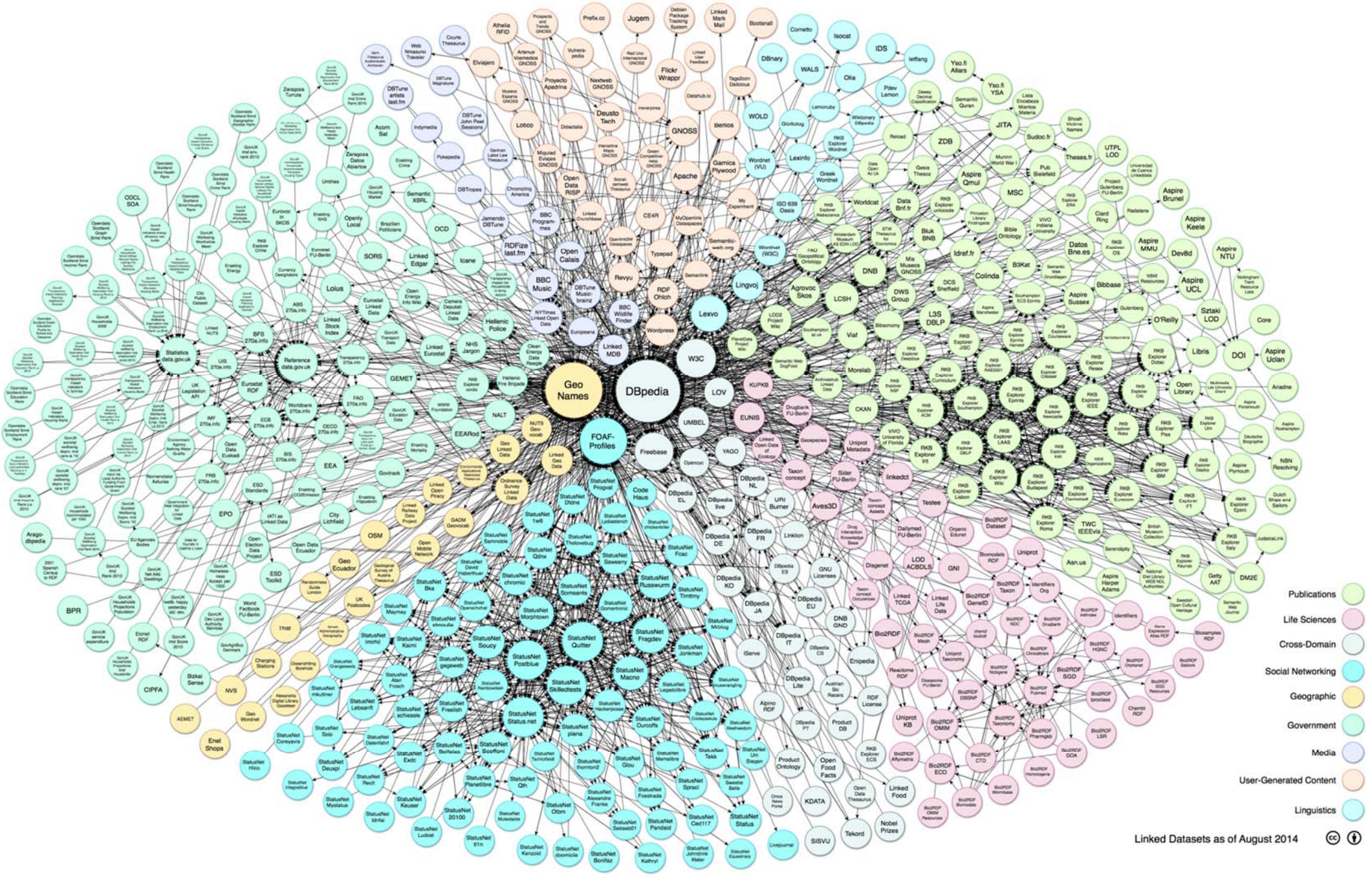
- LOD = Linked Data + Oped Data
- dati disponibili con licenza aperta (es. creative common attribution)
- LOD Cloud
  - dati interconnessi
  - vedi prossima slide

UNIVERSITÀ DEGLI STUDI FIRENZE

DINFO
DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

DISIT
DISTRIBUTED SYSTEMS AND INTERNET TECHNOLOGIES LAB
http://www.disit.org

UNIVERSITÀ DEGLI STUDI FIRENZE
MABIDA

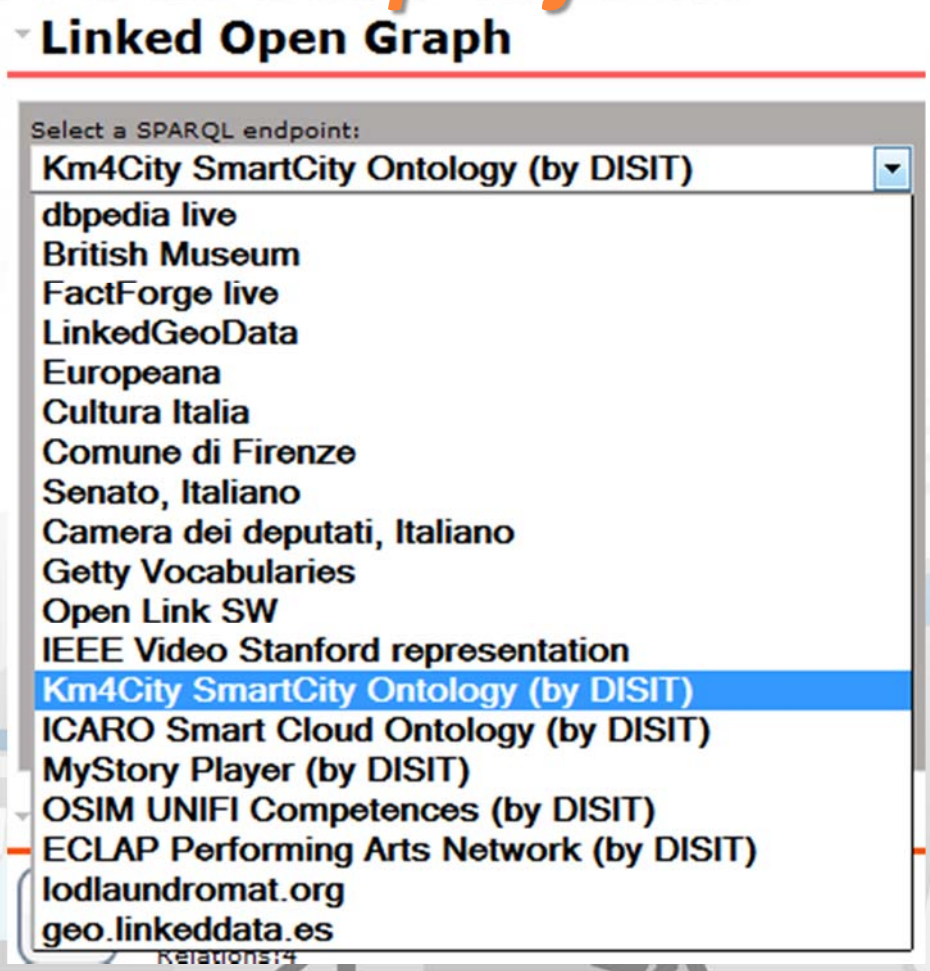


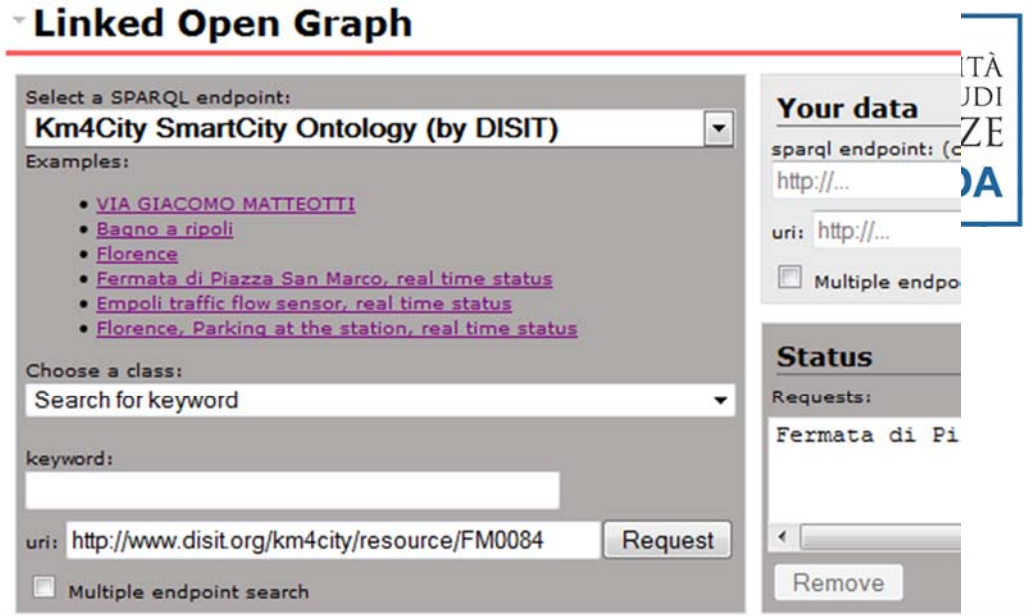
UNIVERSITÀ DEGLI STUDI FIRENZE | DINFO DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE | DISIT DISTRIBUTED SYSTEMS AND INTERNET TECHNOLOGIES LAB http://www.disit.org

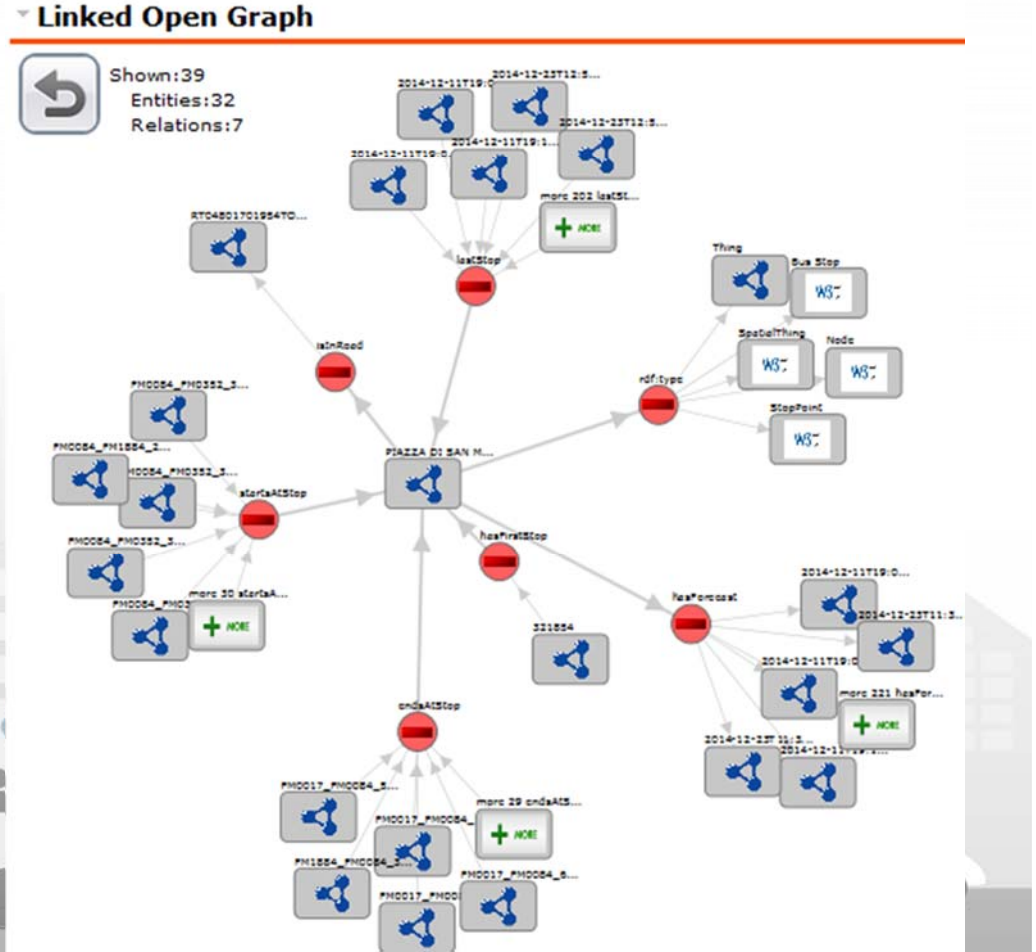
Linked Open Graph

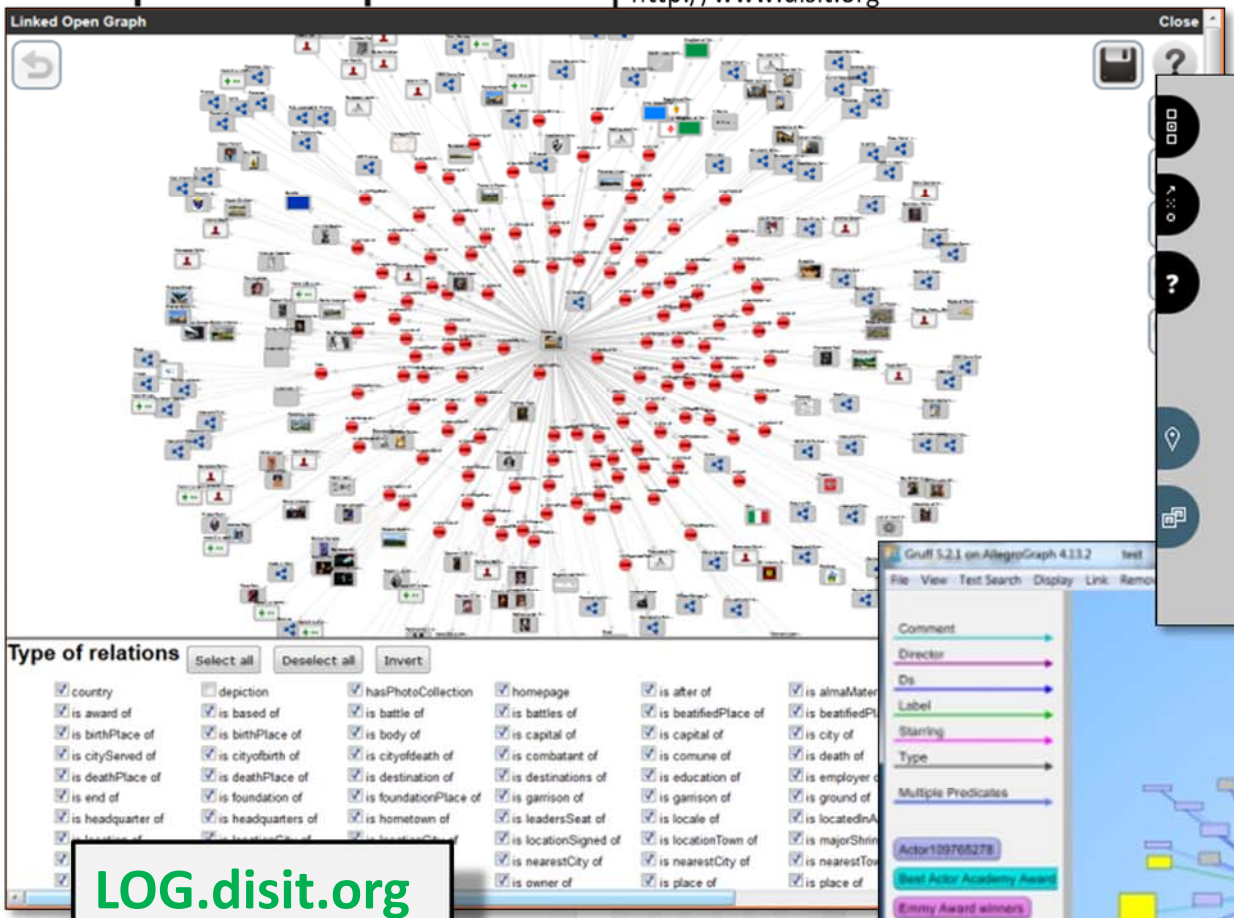
http://log.disit.org

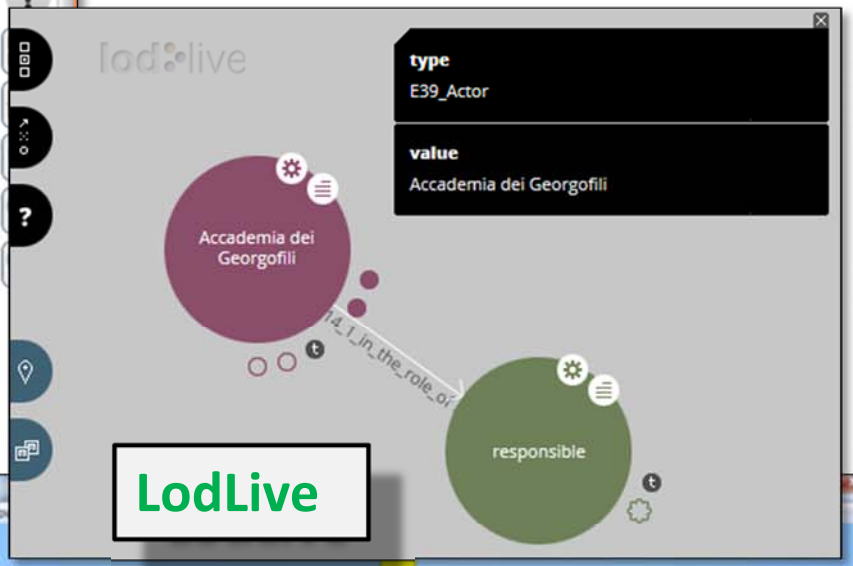
A bus stop info….

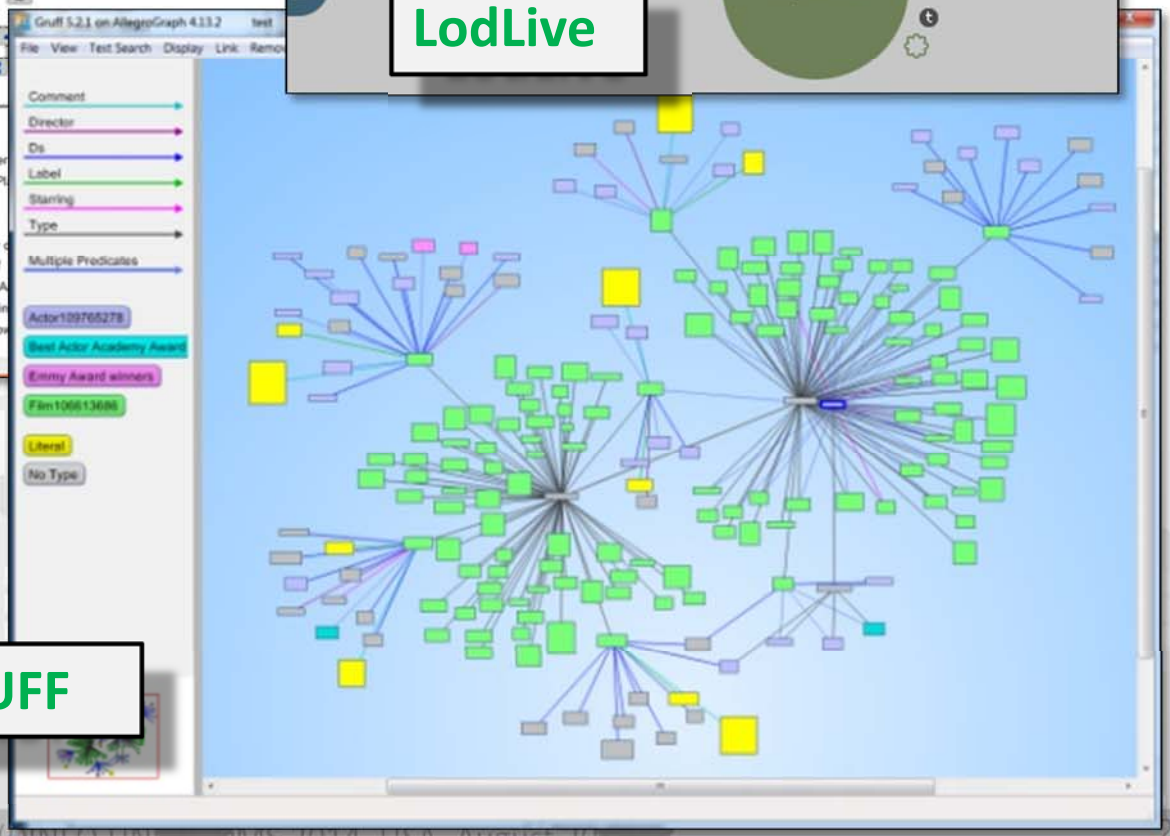







Linked Open Graph
Close
Type of relations
Select all
Deselect all
Invert
country
is award of
is birthPlace of
is cityServed of
is deathPlace of
is end of
is headquarter of
depiction
is based of
is birthPlace of
is cityofbirth of
is deathPlace of
is foundation of
is headquarters of
is hometown of
hasPhotoCollection
is battle of
is body of
is cityofdeath of
is destination of
is foundationPlace 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 death of
is city of
is ground of
LOG.disit.org


lodlive
type
E39_Actor
value
Accademia dei Georgofili
Accademia dei Georgofili
responsible
LodLive


Comment
Director
Ds
Label
Starring
Type
Multiple Predicates
Literal
No Type
GRUFF

SPARQL

- Standard **W3C**
- Linguaggio di interrogazione basato su graph matching
- Prefissi: semplificano la scrittura delle URI nelle query
  - PREFIX dbr:<http://dbpedia.org/resource/>
  - PREFIX dbo:<http://dbpedia.org/ontology/>
  - Permette di scrivere **dbr:Florence** invece di <http://dbpedia.org/resource/Florence>
- **Variabili**: ?a, ?v, ?subj
- **Triple pattern**: tripla dove almeno uno degli elementi è una variabile
  - ?s ***rdf:type*** *dbo:Film*.
  - dbr:Florence ?p ?o
  - ?s ?p ?o

SPARQL

- Esempio:

```
PREFIX dbo:<http://dbpedia.org/ontology/>
PREFIX rdf:<http://www.w3.org/1999/02/22-rdf-syntax-ns#>

SELECT ?f WHERE {
  ?f rdf:type dbo:Film.
} LIMIT 100
```

SPARQL

- Esempio equivalente:

```
PREFIX dbo:<http://dbpedia.org/ontology/>
SELECT ?f WHERE {
  ?f a dbo:Film.
} LIMIT 100
```

- Provarlo su:

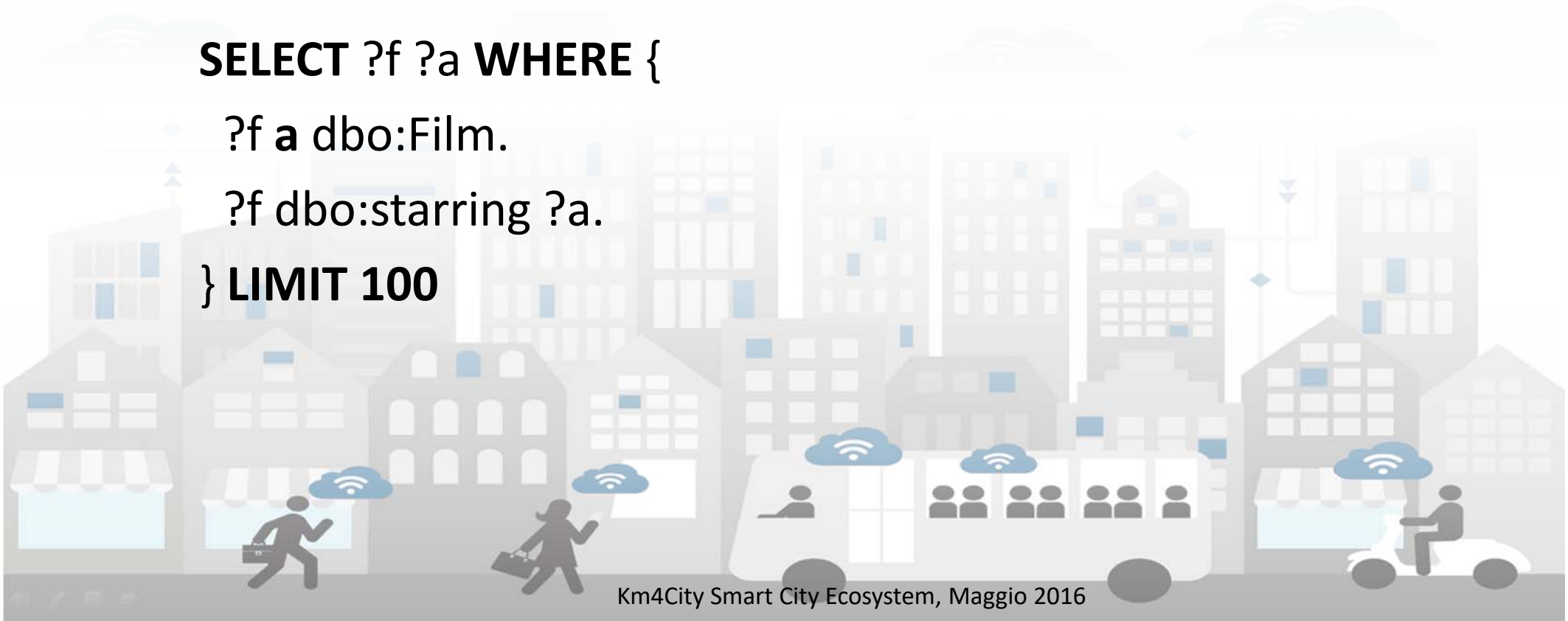
  http://dbpedia.org/sparql

  http://dbpedia-live.openlinksw.com/sparql

SPARQL

- Si possono aggiungere più "triple pattern" che sono valutati in AND
- Es:

```
SELECT ?f ?a WHERE {
  ?f a dbo:Film.
  ?f dbo:starring ?a.
} LIMIT 100
```



Risultato

dbpedia.org/sparql?default-graph-uri=http%3A%2F%2Fdbpedia.org&query=SELE

f	a
http://dbpedia.org/resource/Army_of_Darkness	http://dbpedia.org/resource/Bruce_Campbell
http://dbpedia.org/resource/Army_of_Darkness	http://dbpedia.org/resource/Embeth_Davidtz
http://dbpedia.org/resource/Barry_Lyndon	http://dbpedia.org/resource/Ryan_O'Neal
http://dbpedia.org/resource/Barry_Lyndon	http://dbpedia.org/resource/Gay_Hamilton
http://dbpedia.org/resource/Barry_Lyndon	http://dbpedia.org/resource/Diana_Körner
http://dbpedia.org/resource/Barry_Lyndon	http://dbpedia.org/resource/Marisa_Berenson
http://dbpedia.org/resource/Barry_Lyndon	http://dbpedia.org/resource/Patrick_Magee_(actor)
http://dbpedia.org/resource/Barry_Lyndon	http://dbpedia.org/resource/Hardy_Krüger
http://dbpedia.org/resource/Batman_&_Robin_(film)	http://dbpedia.org/resource/Arnold_Schwarzenegger
http://dbpedia.org/resource/Batman_&_Robin_(film)	http://dbpedia.org/resource/Pat_Hingle
http://dbpedia.org/resource/Batman_&_Robin_(film)	http://dbpedia.org/resource/George_Clooney
http://dbpedia.org/resource/Batman_&_Robin_(film)	http://dbpedia.org/resource/Michael_Gough
http://dbpedia.org/resource/Batman_&_Robin_(film)	http://dbpedia.org/resource/Elle_Macpherson
http://dbpedia.org/resource/Batman_&_Robin_(film)	http://dbpedia.org/resource/Chris_O'Donnell
http://dbpedia.org/resource/Batman_&_Robin_(film)	http://dbpedia.org/resource/John_Glover_(actor)
http://dbpedia.org/resource/Batman_&_Robin_(film)	http://dbpedia.org/resource/Uma_Thurman
http://dbpedia.org/resource/Batman_&_Robin_(film)	http://dbpedia.org/resource/Alicia_Silverstone
http://dbpedia.org/resource/Batman_(1966_film)	http://dbpedia.org/resource/Frank_Gorshin
http://dbpedia.org/resource/Batman_(1966_film)	http://dbpedia.org/resource/Adam_West
http://dbpedia.org/resource/Batman_(1966_film)	http://dbpedia.org/resource/Burgess_Meredith
http://dbpedia.org/resource/Batman_(1966_film)	http://dbpedia.org/resource/Burt_Ward
http://dbpedia.org/resource/Batman_(1966_film)	http://dbpedia.org/resource/Cesar_Romero
http://dbpedia.org/resource/Batman_(1966_film)	http://dbpedia.org/resource/Lee_Meriwether

Attenzione

- Se si **sbaglia** a scrivere il **nome di una classe** o di una **proprietà** o la definizione di un **prefisso** la query non genera errori ma un risultato vuoto o incompleto:

```
PREFIX dbo:<http://dbpedia.org/ontology>
SELECT * WHERE {
        ?f a dbo:Movie.
        ?f dbo:staring ?a.
}
```

SPARQL FILTER

- Usando FILTER(...condizione...) si possono filtrare le righe ritornate

```
SELECT * WHERE {
  ?f a dbo:Film.
  ?f dbo:starring ?a.
  ?a dbo:birthDate ?bd.
  FILTER(?bd>=xsd:date("1980-01-01"))
} LIMIT 100
```

SPARQL - OPTIONAL

- OPTIONAL serve a far si che uno o più triple pattern siano opzionali

```
SELECT * WHERE {
  ?f a dbo:Film.
  ?f dbo:writer ?w.
  ?w dbo:deathDate ?y.
  OPTIONAL {?f dbo:budget ?b}
}
```

quindi si potranno avere righe in cui alla colonna "b" non è associato un valore. Senza optional le righe senza valore per b sono rimosse.

SPARQL – FILTER NOT EXISTS

- Come trovare film scritti da persone ancora in vita (non morte)?

```
SELECT * WHERE {
  ?f a dbo:Film.
  ?f dbo:writer ?w.
  FILTER NOT EXISTS { ?w dbo:deathYear ?y }
}
```

SPARQL ORDER BY

- Usando ORDER BY si possono ordinare le righe ritornate

```
SELECT * WHERE {
  ?f a dbo:Film.
  ?f dbo:starring ?a.
  ?a dbo:birthDate ?bd.
  FILTER(?bd>=xsd:date("1980-01-01") && datatype(?bd)=xsd:date)
} ORDER BY ?bd LIMIT 100
```

SPARQL – UNION

- UNION serve ad unire i risultati di più triple patterns:

```
SELECT * WHERE {
  {?f a dbo:Film.} UNION {?f a schema:MusicAlbum}
  ?f foaf:name ?n.
} ORDER BY ?n
```

SPARQL - COUNT

- Ci sono gli opratori di aggregazione di SQL COUNT, AVG, MIN e MAX
- Ma quanti film ci sono su dbpedia?

```
SELECT (COUNT(*) AS ?nFilm)  WHERE {
  ?f rdf:type dbo:Film.
}
```

- Ma quante triple ci sono su dbpedia?

```
SELECT (COUNT(*) AS ?c)  WHERE {
  ?s ?p ?o.
}
```

SPARQL - DISTINCT

- Ma quali classi ci sono su dbpedia?

```
SELECT DISTINCT ?c  WHERE {
  ?s rdf:type ?c.
}
```

- Ma quali proprietà hanno gli oggetti della classe Film?

```
SELECT DISTINCT ?p  WHERE {
  ?s a dbo:Film.
  ?s ?p ?o.
}
```

SPARQL – GROUP BY

- Si può usare operatore GROUP BY in modo simile a SQL
- Ma quali classi sono le più popolose?

```
SELECT ?c (COUNT(*) AS ?n) WHERE {
  ?s rdf:type ?c.
}
GROUP BY ?c
ORDER BY DESC(?n)
LIMIT 100
```

SPARQL - GRAPH

- Si può avere un "GRAPH pattern" per vincolare delle triple ad appartenere ad un grafo
  - GRAPH <http://mygraph.org> {?s a dbo:Place }
  - GRAPH ?g { ?s a dbo:Film }
- Quali grafi sono presenti e quante triple contengono?

```
SELECT ?g (COUNT(*) AS ?n) WHERE {
  GRAPH ?g {?s ?p ?o}
}
GROUP BY ?g
ORDER BY DESC(?n)
```

SPARQL – SUB QUERY

- All'interno di una query si possono eseguire altre query (esecuzione bottom-up)
- Sintassi **{** SELECT ... WHERE {...} **}**
- **Esempio:**

```
SELECT * WHERE {
 {
   SELECT ?a (COUNT(*) AS ?n) WHERE {
    ?f a dbo:Film.
    ?f dbo:starring ?a.
   } GROUP BY ?a ORDER BY DESC(?n) LIMIT 5
 }
 ?a rdfs:label ?name.
 ?a dbo:birthDate ?bd.
 FILTER(lang(?name)="en")
} ORDER BY DESC(?n)
```

SPARQL - SERVICE

- Le sub query possono essere fatte anche in altro RDF store (Query Federate)
- Sintassi: SERVICE <url del servizio > { query }

- **Esempio:** http://log.disit.org/sparql_query_frontend/

```
PREFIX dbo:<http://dbpedia.org/ontology/>
SELECT DISTINCT * WHERE {
  ?s a km4c:Municipality.
  ?s foaf:name ?name.
  SERVICE <http://it.dbpedia.org/sparql> {
    ?sx a dbo:Place.
    ?sx foaf:name ?n.
    ?sx dbo:administrativeDistrict <http://it.dbpedia.org/resource/Toscana>.
    ?sx dbo:populationTotal ?pp.
    ?sx dbo:abstract ?a
  }
  FILTER(STR(UCASE(?n))=?name)
} ORDER BY DESC(?pp)
LIMIT 1000
```

SPARQL – Property Paths

- I property paths permettono di percorrere le proprietà tra due nodi tramite operatori /,|,^,*,+,?
- Esempio:

  ?f dbo:starring/dbo:birthDate ?bd

  – Equivalente di:

    ?f dbo:starring **?a. ?a** dbo:birthDate ?bd.

SPARQL – Property Paths

- **Scelta tra due o più proprietà/path**
  - <S> p1 | p2 <E>
  - Esempi:

    ?x dc:title | rdfs:label ?y
- **Proprietà inversa**
  - <S> ^p1 <E>
  - Equivalente di: <E> p1 <S>
  - Esempio:

    ?a ^dbo:starring ?f.

SPARQL – Property Paths

- **Proprietà diversa da una proprietà data**
  - <S> !p1 <E>
  - Esempi:

    ?a !dbo:birthDate ?nobd
- **Proprietà opzionale**
  - <S> p1? <E>
  - Esempio:

    ?x rdfs:subClassOf? ?c.

SPARQL – Property Paths

- **Sequenza di proprietà di lunghezza arbitraria >=0**
  - <S> p1* <E>
  - Esempi:
    ?x foaf:knows* ?y
  - «equivalente a»
    ?x foaf:knows/foaf:knows/.../foaf:knows ?y
- **Sequenza di proprietà di lunghezza arbitraria >=1**
  - <S> p1+ <E>
  - Esempi:
    ?x foaf:knows+ ?y
  - Equivalente a:
    ?x foaf:knows/foaf:knows* ?y

SPARQL – Property Paths

- Si possono usare () per raggruppare e comporre operatori
  - dbr:Florence (a | !a)+ ?x
- Nei Property Path non si possono usare variabili
  - ?x dbo:starring/?p ?y.
- Si possono usare per simulare alcuni tipi di inferenza:
  - ?x rdf:type/rdfs:subClassOf* ?c.
  - ?x transitiveProp+ ?y
  - ?x reflxAndTransProp* ?y
  - ?x reflxProp? ?y

Esempi complessi

- Trovare le coppie di film nei quali hanno partecipato gli stessi tre attori
- Trovare i film nei quali hanno lavorato coppie sposate (dbo:spouse)
- Trovare le persone che sono nate lo stesso giorno.

Fine