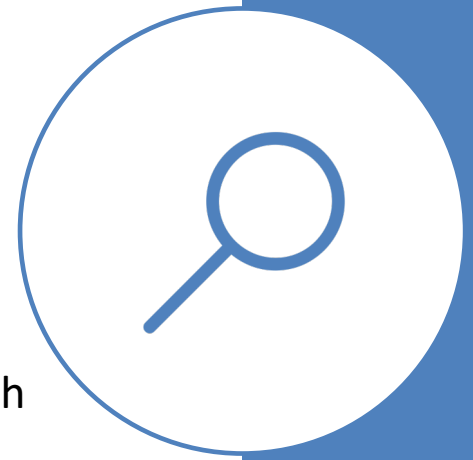
SPARQL

Linguaggio di interrogazione
per RDF

SPARQL 1.1

https://www.w3.org/TR/sparql11-query

- SPARQL è un linguaggio di interrogazione per RDF
- La versione attuale è la 1.1, rilasciata nel 2013
- Le query possono essere diverse:
  - selezione di triple secondo un semplice pattern
  - query complesse costruite con filtri, aggregatori, path expressions, ecc.
  - query booleane (ASK)
- SPARQL permette di creare nuovi grafi con le triple estratte



Specifiche di SPARQL

- SPARQL 1.1 Overview
- SPARQL 1.1 Query Language
- SPARQL 1.1 Update
- SPARQL 1.1 Service Description
- SPARQL 1.1 Federated Query
- SPARQL 1.1 Query Results JSON Format
- SPARQL 1.1 Query Results CSV and TSV Formats
- SPARQL Query Results XML Format (Second Edition)
- SPARQL 1.1 Entailment Regimes
- SPARQL 1.1 Protocol
- SPARQL 1.1 Graph Store HTTP Protocol

SPARQL: risultati

Il risultato di una query può essere un insieme di triple o uno o più grafi RDF

SPARQL supporta i formati più comuni:

- Extensible Markup Language (XML)
- JavaScript Object Notation (JSON)
- Comma Separated Values (CSV)
- Tab Separated Values (TSV)

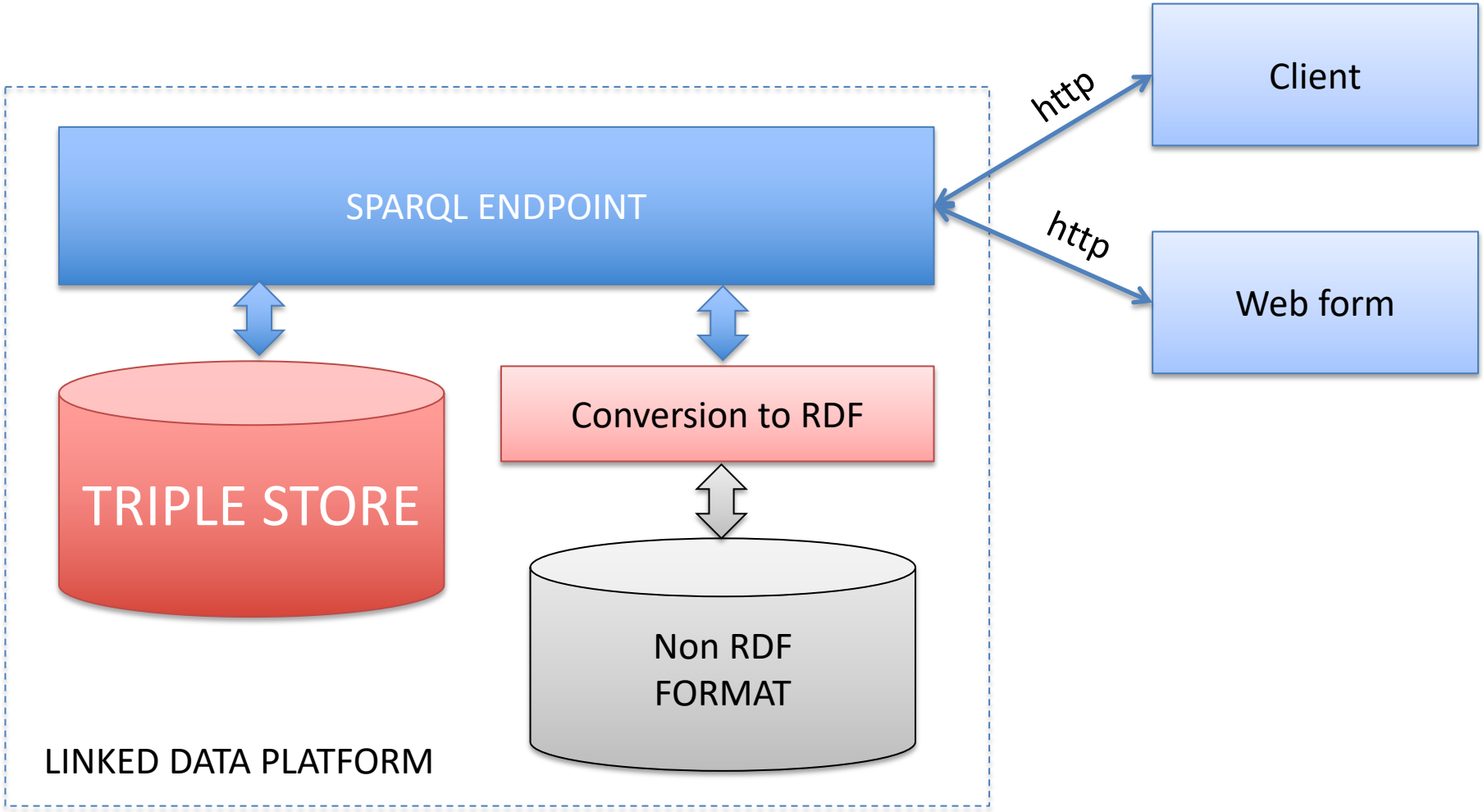
SPARQL endpoint

- Le query vengono indirizzate a indirizzo http che ospita un Endpoint SPARQL
- L'Endpoint esegue le query su uno o più dataset contenuti in uno store di triple (triple store)
  - Da specifiche, basta una http GET per ottenere il risultato della query in formato RDF

- Molte Linked Data Platforms (LDP) hanno un'interfaccia in cui è possibile inserire manualmente le query.

- In Protégé esiste un apposito tab per SPARQL

SPARQL Endpoint



Schema di una query SPARQL

```
# prefix declarations
PREFIX foo: <http://example.com/resources/>
...
# dataset definition
FROM ...

# result clause
SELECT ...

# query pattern
WHERE {
   ...
}

# query modifiers
ORDER BY ...
```

WHERE

- Nella clausola WHERE si trovano una o più triple che contengono variabili ?v
  → Un sottoinsieme di queste variabili compare nella clausola SELECT
- Le triple formano un *pattern* che corrisponde a un insieme di grafi
- Il pattern viene unificato con le triple presenti nel repository rdf
- Il risultato nella query è dato da tutte le occorrenze delle variabili contenute nella clausola WHERE (proiezione) nei grafi trovati

Query minima

Lo schema minimo per la query SPARQL è il seguente:

```
SELECT ?x
WHERE
{
      ?x predicato_1 ?y.
      ?x predicato_2 ?z
}
```

- I predicati delle triple possono essere object properties, data properties o predicati rdf:type
- Soggetti o oggetti delle triple possono essere variabili, classi o individui, oppure letterali

Cinema 1

- Seleziona tutti gli individui che appartengono alla classe Persona

→Settare il prefisso cinema

```
PREFIX cinema: <http://…/cinema#>

SELECT ?subject ?object
      WHERE { ?subject rdf:type cinema:Persona}
```

→ Personaggio?

Interrogare classi definite

La query

```
SELECT ?r
  WHERE { ?r rdf:type cinema:RegistaFilm}
```

Restituisce un insieme di risultati vuoto su alcuni endpoint (per es. Protégé) e non su altri che supportano inferenze (per es. GraphDB).

Città 1

- Trova l'individuo che ha come popolazione la popolazione di Torino ed è una città

```
SELECT ?subject ?object
    WHERE {?subject :haPopolazione :popolazioneDiTorino;
                    rdf:type :Città}
```

Città 2

- Trova le città che hanno una popolazione grande

```
SELECT ?c ?p
    WHERE  {
        ?c :haPopolazione ?p.
        ?p rdf:type :PopolazioneAmpia.
        }
```

Città 2

- Trova gli individui che hanno una popolazione grande e sono in Asia

```
SELECT ?c ?p
    WHERE {
        ?c  :haPopolazione ?p;
            :situatoIn ?n.
        ?n  rdf:type :NazioneAsiatica.
        ?p  rdf:type :PopolazioneAmpia.
        }
```

Cinema 2

- Trova le persone che sono registi

```
SELECT ?p ?f
      WHERE {
            ?p   rdf:type cinema:Persona;
                 cinema:haRuolo ?r.
            ?r   rdf:type cinema:Regista;
                 cinema:ruoloInFilm ?f.
}
```

Agenzia 1

- Cerca tutti gli appartamenti e i locali di ognuno

```
SELECT ?appartamento ?stanza
    WHERE { ?appartamento agenzia:compostoDa ?stanza
} order by ?appartamento ?stanza
```

- Restituisce tutte le coppie appartamento stanza che soddisfano il pattern
- Lo stesso appartamento è ripetuto per ogni stanza

Agenzia 2

Trova gli appartamenti situati in comuni turistici

```
SELECT ?appartamento ?comune
    WHERE {
        ?appartamento agenzia:èSituatoIn ?comune.
        ?comune a agenzia:ComuneTuristico
                    }
```

Restituisce tutti gli appartamenti che siano situati in un comune turistico

SPARQL: Filtri

- I filtri sono elementi opzionali che si possono includere nella clausola WHERE per selezionare un sottoinsieme nell'insieme dei risultati.

- Per esempio, è possibile confrontare un dato con una stringa o un numero

```
SELECT ?c ?p
      WHERE     {
            ?c     :haPopolazione ?p;
                   :situatoIn ?n;
                   :nome ?o.
            ?n rdf:type :NazioneAsiatica.
            ?p rdf:type :PopolazioneAmpia.
            FILTER regex(?o, "Tokio").
      }
```

Città 3

Selezione sulla base dell'anno di fondazione

Metropoli asiatiche fondate dopo l'anno 800.

```
SELECT ?c
    WHERE {
        ?c   :situatoIn ?n;
             :dataFondazione ?a.
        ?n rdf:type :NazioneAsiatica.

    FILTER (?a>800).
    }
```

Operazioni sui risultati

E’ possibile raggruppare, ordinare e contare gli elementi inclusi nel risultato:

- OrderBy
- Distinct
- Group
- Count

ORDER BY

Ordina i risultati delle query sulla base di una delle variabili

```
SELECT ?f  ?p
WHERE {
    ?f cinema:haPersonaggio  ?p .
} ORDER BY ?f
```

GROUP BY

- Raggruppa i risultati sulla base di una risorsa
  - Utilizzata spesso in associazione con COUNT
  - Le variabili che appaiono nella clausola SELECT devono apparire anche nella clausola GROUP BY

```
SELECT ?subject (COUNT(?object) AS ?count)
WHERE {
   ?subject relation ?object .
} GROUP BY ?subject
```

COUNT – GROUP BY

Estrarre e contarre gli individui che hanno determinati requisiti

```
SELECT ?f (COUNT(?p) AS ?count)
WHERE {
   ?f cinema:haPersonaggio  ?p .
}
GROUP BY ?f
ORDER BY ?f
```

BLANK NODE

Seleziona tutte le entità *situate da "*qualche parte" (→ [] indica un blank node)

```
SELECT ?c
WHERE {
      ?c :situatoIn [].
}
```

Per estrarre tutte le classi (incluse quelle dichiarate come equivalenti)

```
SELECT ?e
WHERE {
      ?e a ?c
}
```

HAVING

Cerca le case di vacanze grandi

```
SELECT ?appartamento (COUNT (?stanza) AS ?count)
WHERE {
   ?appartamento agenzia:compostoDa  ?stanza ;
      agenzia:èSituatoIn ?comune.
   ?comune a agenzia:ComuneTuristico.
}
GROUP BY ?appartamento
HAVING (?count > 3)
```

La clausola HAVING permettere di filtrare i risultati dell'aggregazione (appartamenti con almeno due stanze)

DBPedia: esempio

- http://dbpedia.org/sparql

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

SELECT ?person ?caption
WHERE { ?person     rdf:type dbp:Person ;
                                    dbpprop:name ?name ;
                                    dbpprop:caption ?caption .
FILTER ( regex(?name, 'Manet' ))
        }
LIMIT 5
```

Immagini di persone con didascalia contenente la parola “Manet”

Utilizzo di UNION

- UNION permette di fornire pattern alternativi nella query

- Seleziona tutte le città europee e le città asiatiche

```
SELECT ?subject ?object
    WHERE {
        {?subject rdf:type  citta:NazioneEuropea}
        UNION
        {?subject rdf:type  citta:NazioneAsiatica}
    }
```

OPTIONAL

- Aggiungere porzione opzionale al pattern

- Seleziona città italiane e i loro monumenti, se ne hanno

```
SELECT ?cit ?mon
    WHERE { ?cit citta:situatoIn  citta:Italia
    OPTIONAL {?cit citta:possiedeMonumento ?mon}
}
```

AS

- Permette di rinominare una variabile
- Spesso usato insieme a operazioni sui risultati (somma, minimo, media, ecc.)

```
AS
SELECT ?cit (?mon ?monumento)
    WHERE { ?cit citta:situatoIn  citta:Italia
    OPTIONAL {?cit citta:possiedeMonumento ?mon}
}
```

Aggregatori: MIN/MAX

- Funzioni per calcolare valore minimo o massimo su un gruppo risultati

- Trova la data di fondazione di città più antica per ogni nazione

```
SELECT ?nazione ?nome (MIN(?data) AS ?piùantica)
      WHERE {
            ?cit citta:dataFondazione  ?data.
            ?cit citta:situatoIn ?nazione.
      OPTIONAL {?cit citta:nome ?nome}
}
GROUP BY ?nazione ?nome
```

MAX

- Trova il compenso più alto per ogni attore

```
SELECT ?persona  (MAX(?compenso) AS
?piùpagato)
    WHERE  {
        ?persona cinema:haRuolo ?ruolo.
        ?ruolo cinema:compenso ?compenso
        }
GROUP BY ?persona
```

Subqueries

- In SPARQL è possibile annidare più queries
- Le queries verranno eseguite dalla più interna alla più esterna

```
SELECT ?ruolo ?count                                        QUANTI PERSONAGGI
WHERE {                                                     PER FILM DI SPIELBERG
  cinema:StevenSpielberg cinema:haRuolo ?ruolo .
  ?ruolo cinema:ruoloInFilm ?film.
      {SELECT ?film (COUNT (?personaggio) AS ?count)        QUANTIPERSONAGGI PER
FILM
      WHERE    {
            ?film cinema:haPersonaggio ?personaggio.
            }
      GROUP BY ?film}
FILTER (?count > 1)
}
```

SPARQL: aspetti avanzati

- SPARQL comprende molte funzionalità, che permettono di lavorare con i grafi in modo simile a un database.
  - https://www.w3.org/TR/sparql11-overview/
  - non sono supportate da tutti gli endpoint
- Alcuni aspetti avanzati:
  - Inserimento, cancellazione e modifica di triple
  - Query booleane

SPARQL Insert

- Fa parte di Update language (SPARUL)

```
PREFIX cinema: <http://…cinema#>

INSERT {
      <http://…cinema#Luke_Skywalker>
      cinema:personaggioDi
      <http:…cinema#StarWars7>. }
WHERE {}
```

Inserisce nel grafo un individuo #Luke_Skywalker che è personaggioDi #StarWars7

SPARQL: Delete

- Delete cancella una o più triple dal grafo

PREFIX cinema: <http://…cinema#>

DELETE { ?subj ?prop ?val }

WHERE { ?subj ?prop ?val ; rdf:type cinema:Personaggio. }

Questa tripla cancella tutti gli individui che appartengono alla classe Personaggio

Attenzione: non cancella gli individui che erano già presenti nel grafo al caricamento

SPARQL: Ask

- Ask permette di ottenere una risposta si/no rispetto all'esistenza di una o più triple

```
ASK  {
      <http://www…cinema#HanSolo>
      cinema:personaggioDi <http://
      www….cinema#StarWars7>  .
}
```

La risposta ha sempre la forma seguente:

```
{
   "head":  {},
   "boolean":  true
}
```

Relazioni con inferenze

Se il triple store è in grado di compiere inferenze, le operazione di update potrebbero essere limitate ai dati inferiti.

Per esempio, un'operazione di DELETE potrebbe cancellare le triple inferite senza produrre effetti

→ Sono implicate, perciò sono soggette a cancellazione