UNIVERSITA' DEGLI STUDI DI ROMA

TorVergata Università

SPARQL 1.1

Andrea Turbati <turbati@info.uniroma2.it>

Breve storia di SPARQL 1.1

- SPARQL 1.0 ha varie limitazioni
- SPARQL 1.0 introdotto nel 2008, SPARQL 1.1 nel 2013
- SPARQL 1.1 ha la stessa sintassi di SPARQL 1.0 ed è retrocompatibile, ma introduce una serie di nuove funzionalità

Nuove Features

- Aggregazione
- Subquery
- Negazione
- Property Path
- Assegnazione

- Update

Andrea Turbati turbati@info.uniroma2.it
http://art.uniroma2.it/turbati

Aggregazione

- Possibilità di aggregare i risultati mediante *GROUP_BY*
- Una volta che si è aggregato, si possono effettuare ulteriori operazioni grazie ai seguenti operatori: *COUNT*, *SUM*, *MIN*, *MAX*, *AVG*, *GROUP_CONCAT*, *SAMPLE*
- Si può usare *HAVING* per filtrare sui gruppi (in modo simile a *FILTER*, ma sui gruppi e ponendo quindi *HAVING* dopo la WHERE)

Aggregazione

```
PREFIX : <http://data.example/>
SELECT ?x (AVG(?size) AS ?asize)
WHERE {
        ?x :size ?size
}
GROUP BY ?x
HAVING(AVG(?size) > 10)
```

Si raggruppano i vari risultati in base al valore della variabile ?x (cioè si fa un gruppo per ogni valore diverso di ?x), si escludono quei gruppi che hanno un valore medio di ?size minore o uguale a 10 e poi si restituisce il valore ?x (che identifica il gruppo) e il valore medio di ?size

Subquery

- Possibilità di effettuare query annidate (anche a più livelli)
- Verrà sempre eseguita la query più interna per poi procedere verso quelle più esterne (utile per forzare l'esecuzione in un determinato ordine)
- Le query annidate possono usare gli operatori di aggregazione

Subquery

```
PREFIX : <http://people.example/>
PREFIX : <http://people.example/>
SELECT ?y ?minName
WHERE {
  :alice :knows ?y .
  {
    SELECT ?y (MIN(?name) AS ?minName)
    WHERE {
      ?y :name ?name .
    } GROUP BY ?y
  }
}
```

```
@prefix : <http://people.example/> .

:alice :name "Alice", "Alice Foo", "A. Foo" .
:alice :knows :bob, :carol .
:bob :name "Bob", "Bob Bar", "B. Bar" .
:carol :name "Carol", "Carol Baz", "C. Baz" .
```

y	minName
:bob	"B. Bar"
:carol	"C. Baz"

Negazione

- In SPARQL 1.0 poteva essere eseguita la negazione mediante il costrutto *OPTIONAL + !BOUND*

- In SPARQL 1.1 sono stati introdotti
  - *NOT EXISTS* (da usare dentro *FILTER*), verifica che un graph pattern non unifica (dato il valore assegnato alle variabili)
  - *MINUS*, restituisce le soluzioni a cui vengono tolte quelle calcolate nel graph pattern dentro la *MINUS*

Negazione (NOT EXISTS)


```
PREFIX  rdf:   <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX  foaf:  <http://xmlns.com/foaf/0.1/>

SELECT ?person
WHERE  {
    ?person rdf:type  foaf:Person .
    FILTER NOT EXISTS { ?person foaf:name ?name }
}
```

```
@prefix  :       <http://example/> .
@prefix  rdf:    <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix  foaf:   <http://xmlns.com/foaf/0.1/> .

:alice  rdf:type   foaf:Person .
:alice  foaf:name  "Alice" .
:bob    rdf:type   foaf:Person .
```

person
<http://example/bob>

Negazione (MINUS)

```
PREFIX :       <http://example/>
PREFIX foaf:   <http://xmlns.com/foaf/0.1/>

SELECT DISTINCT ?s
WHERE {
   ?s ?p ?o .
   MINUS {
      ?s foaf:givenName "Bob" .
   }
}
```

```
@prefix :       <http://example/> .
@prefix foaf:   <http://xmlns.com/foaf/0.1/> .

:alice  foaf:givenName "Alice" ;
        foaf:familyName "Smith" .

:bob    foaf:givenName "Bob" ;
        foaf:familyName "Jones" .

:carol  foaf:givenName "Carol" ;
        foaf:familyName "Smith" .
```

s
<http://example/carol>
<http://example/alice>

MINUS e NOT EXISTS

- Entrambe sono usate per eliminare dei risultati, ma hanno due approcci profondamente diversi (il più delle volte producono gli stessi risultati e sono quindi intercambiabili)
- MINUS e NOT EXISTS tornano gli stessi risultati quando almeno una variabile usata al loro interno viene usata anche nel resto della query (nella WHERE)

```
PREFIX  rdf:    <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX  foaf:   <http://xmlns.com/foaf/0.1/>

SELECT ?person
WHERE
{
    ?person rdf:type  foaf:Person .
    FILTER EXISTS { ?person foaf:name ?name }
}
```

MINUS e NOT EXISTS

- In questo caso i risultati sono diversi visto che le variabili in "NOT EXISTS" e "MINUS" non sono usate nel resto della query:

```
SELECT *
{
  ?s ?p ?o
  FILTER NOT EXISTS { ?x ?y ?z }
}
```

```
SELECT *
{
    ?s ?p ?o
    MINUS
       { ?x ?y ?z }
}
```

- La query con la FILTER NOT EXISTS non torna niente, mentre quella con MINUS torna tutto

Property Path

UNIVERSITA' DEGLI STUDI DI ROMA TorVergata Università

- Concatenare più predicate in modi diversi

Syntax Form	Property Path Expression Name	Matches
iri	PredicatePath	An IRI. A path of length one.
^*elt*	InversePath	Inverse path (object to subject).
elt1 / *elt2*	SequencePath	A sequence path of *elt1* followed by *elt2*.
elt1 | *elt2*	AlternativePath	A alternative path of *elt1* or *elt2* (all possibilities are tried).
*elt**	ZeroOrMorePath	A path that connects the subject and object of the path by zero or more matches of *elt*.
elt+	OneOrMorePath	A path that connects the subject and object of the path by one or more matches of *elt*.
elt?	ZeroOrOnePath	A path that connects the subject and object of the path by zero or one matches of *elt*.
!*iri* or !(iri_1| ...|iri_n)	NegatedPropertySet	Negated property set. An IRI which is not one of iri_j. !*iri* is short for !(*iri*).
!^*iri* or !(^iri_1| ...|^iri_n)	NegatedPropertySet	Negated property set where the excluded matches are based on reversed path. That is, not one of iri_1...iri_n as reverse paths. !^*iri* is short for !(^*iri*).
!(iri_1| ...|iri_j|^iri_{j+1}| ...|^iri_n)	NegatedPropertySet	A combination of forward and reverse properties in a negated property set.
(*elt*)		A group path *elt*, brackets control precedence.

Andrea Turbati turbati@info.uniroma2.it
http://art.uniroma2.it/turbati

Property Path

```
SELECT ?x ?name
{
  ?x  foaf:mbox <mailto:alice@example> .
  ?x  foaf:knows ?a1 .
  ?a1 foaf:knows ?a2 .
  ?a2 foaf:name ?name .
}
```

Equivalente a:

```
{
  ?x foaf:mbox <mailto:alice@example> .
  ?x foaf:knows/foaf:knows/foaf:name ?name .
}
```

Property Path

```
{
  ?x foaf:knows ?gen1 .
  ?y foaf:knows ?gen1 .
  FILTER(?x != ?y)
}
```

Equivalente a:

```
{
  ?x foaf:knows/^foaf:knows ?y .
  FILTER(?x != ?y)
}
```

Property Path

I foaf:name di tutti coloro collegati tramite foaf:knows con ?x sia in modo diretto che in modo transitivo (sempre con foaf:knows)

```
{
  ?x foaf:mbox <mailto:alice@example> .
  ?x foaf:knows+/foaf:name ?name .
}
```

Le triple aventi come predicato le sottoproprietà (in modo transitivo) di :property

```
{ ?x ?p ?v . ?p rdfs:subPropertyOf* :property }
```

Gli rdf:type di tutte le super classi (in modo transitivo) per ogni possible istanza (cioè, per ogni istanza, tutte le sue classi di appartenenza, consideranzo anche quelle inferite per la gerarchia)

```
{ ?x rdf:type/rdfs:subClassOf* ?type }
```

Tutti i valori appartenenti alla lista :list (ricordarsi che in RDF le liste sono formate da una testa e da una coda che è a sua volta una lista contenente il resto della lista in questione e così in modo ricorsivo)

```
{ :list rdf:rest*/rdf:first ?element }
```

Assegnazione

- È possible assegnare il valore di un'espressione ad una variabile

- Due modi per fare ciò:
  - BIND
  - VALUES

- Le variabili così create non devono essere state usate fino a quel punto nella query

Assegnazione (BIND)

Nella variabile ?price viene posto il valore ottenuto dall'espressione ?p*(1-?discount), cioè si effettua tale operazione aritmetica e si pone il literal risultante in ?price

```
PREFIX  dc:  <http://purl.org/dc/elements/1.1/>
PREFIX  ns:  <http://example.org/ns#>

SELECT  ?title ?price
{  ?x ns:price ?p .
   ?x ns:discount ?discount
   BIND (?p*(1-?discount) AS ?price)
   FILTER(?price < 20)
   ?x dc:title ?title .
}
```

In ?name si mette il literal risultante dalla concatenzione tra il valore di ?G e di ?S (ponendo uno spazio tra di loro)

```
PREFIX foaf:   <http://xmlns.com/foaf/0.1/>
SELECT ?name
WHERE  {
   ?P foaf:givenName ?G ;
      foaf:surname ?S
   BIND(CONCAT(?G, " ", ?S) AS ?name)
}
```

Assegnazione (Values)

In ?book si pongono i valori :book1 e :book3, cioè ?book avrà prima il valore :book1 e poi il valore :book3

```
PREFIX dc:   <http://purl.org/dc/elements/1.1/>
PREFIX :     <http://example.org/book/>
PREFIX ns:   <http://example.org/ns#>

SELECT ?book ?title ?price
{
   VALUES ?book { :book1 :book3 }
   ?book dc:title ?title ;
         ns:price ?price .
}
```


Andrea Turbati turbati@info.uniroma2.it
http://art.uniroma2.it/turbati

UPDATE

- Con SPARQL 1.1 è possible modificare direttamente i dati RDF

- Possibili modifiche:
  - INSERT / INSERT DATA (la INSERT necessità di unificare le variabili tramite una WHERE, invece la INSERT DATA accetta solo risorse RDF già specificate, cioè niente variabili)
  - DELETE / DELETE DATA (discorso analogo al precedente)
  - LOAD (per caricare dei dati da un file/URL nel repository)
  - CLEAR (rimuovere tutti i dati dal grafo specificato)

INSERT/INSERT DATA


```
PREFIX dc: <http://purl.org/dc/elements/1.1/>
INSERT DATA
{
  <http://example/book1> dc:title "A new book" ;
                         dc:creator "A.N.Other" .
}
```

```
PREFIX dc:  <http://purl.org/dc/elements/1.1/>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>

INSERT
  { GRAPH <http://example/bookStore2> { ?book ?p ?v } }
WHERE
  { GRAPH  <http://example/bookStore>
      { ?book dc:date ?date .
        FILTER ( ?date > "1970-01-01T00:00:00-02:00"^^xsd:dateTime )
        ?book ?p ?v
  } }
```

DELETE / DELETE DATA

```
PREFIX dc: <http://purl.org/dc/elements/1.1/>

DELETE DATA
{
  <http://example/book2> dc:title "David Copperfield" ;
                         dc:creator "Edmund Wells" .
}
```

```
PREFIX dc:  <http://purl.org/dc/elements/1.1/>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>

DELETE
 { ?book ?p ?v }
WHERE
 { ?book dc:date ?date .
   FILTER ( ?date > "1970-01-01T00:00:00-02:00"^^xsd:dateTime )
   ?book ?p ?v
 }
```

DELETE + INSERT

È possible rimuovere e inserire dei dati RDF con una singola query, utile quando si deve fare ad esempio una rename o una replace

```
PREFIX foaf:  <http://xmlns.com/foaf/0.1/>

WITH <http://example/addresses>
DELETE { ?person foaf:givenName 'Bill' }
INSERT { ?person foaf:givenName 'William' }
WHERE
  { ?person foaf:givenName 'Bill'
  }
```

LOAD / CLEAR

```
LOAD ( SILENT )? IRIref_from ( INTO GRAPH IRIref_to )?
```

```
# Remove all triples from a specified graph.
CLEAR GRAPH IRIref
```

Equivalente a

```
# Remove all triples from the graph named with the IRI denoted by IRIref.
DELETE { GRAPH IRIref { ?s ?p ?o } } WHERE { GRAPH IRIref { ?s ?p ?o } }
```

Riferimenti

- SPARQL 1.0: https://www.w3.org/TR/rdf-sparql-query/
- SPARQL 1.1: https://www.w3.org/TR/sparql11-query/
- SPARQL 1.1 UPDATE: https://www.w3.org/TR/sparql11-update/