Azzolini Riccardo 2020-05-12

RDF

1 Concetti principali

I concetti fondamentali del modello RDF (che si ritrovano anche nelle sue serializzazioni) sono i seguenti:

Risorse: sono i soggetti/oggetti e i predicati (cioè, in generale, le entità che si vuole rappresentare). Ciascuna risorsa è identificata da un URI.

Proprietà: sono un tipo particolare di risorse – quelle che rappresentano i predicati.

Nota: Il termine "proprietà" viene usato (al posto di "predicato") soprattutto quando si specifica il valore di un attributo di una risorsa.

Statement: sono espressioni che specificano le proprietà delle risorse. Ogni statement è una **tripla** *soggetto–predicato–oggetto*, o *soggetto–proprietà–valore*, o *risorsa–proprietà–valore*, nella quale:

- il soggetto è una risorsa;
- il predicato è una proprietà;
- l'oggetto/valore può essere anch'esso una risorsa, oppure un **letterale** (un valore atomico – in pratica, una stringa).

L'approccio di rappresentazione delle informazioni che consiste nel descrivere le relazioni che intercorrono tra varie entità è una vecchia idea, già seguita dalle ricerche sull'intelligenza artificiale degli anni '70 (e indicata, in questo ambito, con il termine "reti semantiche").

2 Serializzazione RDF/XML

Per utilizzare in pratica il modello dei dati RDF, è necessaria una serializzazione. A tale scopo, il W3C ha definito, insieme al modello RDF vero e proprio, anche la serializzazione **RDF/XML**, che permette di rappresentare gli statement RDF attraverso un opportuno insieme di tag XML.

Ad esempio, lo statement

Mario Rossi è il proprietario della pagina web `http://example.org/mrossi`

viene serializzato nel seguente documento:

```
<?xml version="1.0" encoding="UTF-8"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
  <rdf:Description rdf:about="Mario Rossi">
    <proprietario>
      http://example.org/mrossi
    </proprietario>
  </rdf:Description>
</rdf:RDF>
```

- l'elemento `rdf:Description` rappresenta l'intero statement;
- l'attributo `rdf:about` indica il soggetto;
- l'elemento `proprietario` corrisponde al predicato;
- il contenuto di `proprietario` (cioè l'URL della pagina web) è l'oggetto.

Come ci si accorge subito, la serializzazione di uno statement RDF in RDF/XML non è particolarmente immediata. Il problema principale è che essa "nasconde" il fatto che uno statement è una tripla soggetto–predicato–oggetto.

3 Letterali tipati

I letterali di RDF sono tipati: ad esempio, è possibile utilizzare i tipi di XML Schema.

In RDF/XML, il tipo di un letterale (se non è una stringa) viene indicato con l'attributo `rdf:datatype`, il cui valore deve essere un URI che identifica un tipo appartenente a un determinato namespace. Ad esempio, la serializzazione

```
<rdf:Description rdf:about="Mario Rossi">
  <age rdf:datatype="http://www.w3.org/2001/XMLSchema#integer">
    27
  </age>
</rdf:Description>
```

corrisponde allo statement

Mario Rossi ha 27 anni

il cui oggetto, 27, è un letterale del tipo `integer` di XML Schema.

4 Reificazione

L'oggetto di uno statement può essere a sua volta uno statement. Ad esempio:

> Mauro Ferrari crede che *Modelli Innovativi è insegnato da Alberto Trombetta*

La tecnica che lo permette è chiamata **reificazione**: si rappresenta uno statement attraverso una risorsa, e si associa a essa un URI, che può essere usato per fare riferimento a questo statement all'interno di altri statement.

Nella serializzazione RDF/XML, ad esempio, lo statement

```
<rdf:Description rdf:about="MIGD">
  <insegnatoDa>Alberto Trombetta</insegnatoDa>
</rdf:Description>
```

può essere reificato come:

```
<rdf:Statement rdf:about="StatementAboutMIGD">
  <rdf:subject rdf:resource="MIGD"/>
  <rdf:predicate rdf:resource="insegnatoDa"/>
  <rdf:object>Alberto Trombetta</rdf:object>
</rdf:Statement>
```

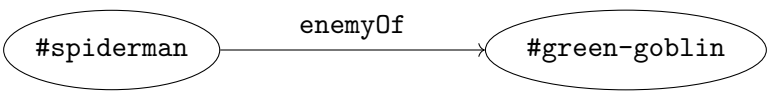
5 Serializzazione Turtle

Siccome la serializzazione RDF/XML è troppo complessa, sono stati definiti numerosi formati di serializzazione alternativi per il modello RDF. Uno di questi è **Turtle** (*Terse RDF Triple Language*), anch'esso standardizzato dal W3C (nel 2014).

Turtle rappresenta gli statement direttamente sotto forma di triple soggetto–predicato–oggetto; i tre componenti della tripla sono separati da spazi, e lo statement è terminato da un punto. Ad esempio, la serializzazione

```
<http://example.org/#spiderman> <http://example.org/enemyOf>
    <http://example.org/#green-goblin> .
```

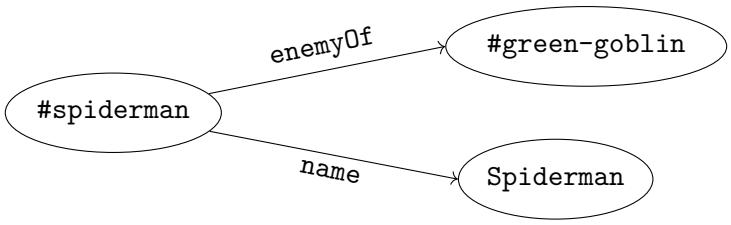
indica lo statement



5.1 Predicate list

Spesso, si hanno più statement con predicati e oggetti diversi, ma con lo stesso soggetto. In tal caso, Turtle permette di indicare il soggetto una volta sola, seguito da una **predicate list**: una lista di coppie predicato–oggetto, separate da punto e virgola.

Ad esempio, il seguente grafo, composto da due statement con lo stesso soggetto,



può essere rappresentato con una predicate list,

```
<http://example.org/#spiderman>
    <http://example.org/enemyOf> <http://example.org/#green-goblin> ;
    <http://example.org/name> "Spiderman" .
```

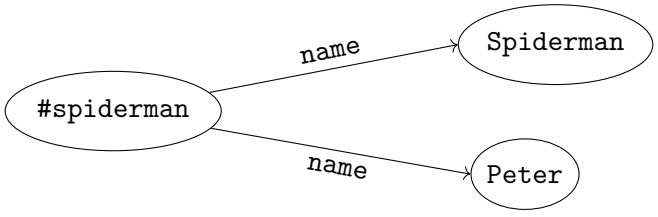
che equivale a queste due triple:

```
<http://example.org/#spiderman> <http://example.org/enemyOf>
    <http://example.org/#green-goblin> .
<http://example.org/#spiderman> <http://example.org/name> "Spiderman" .
```

5.2 Object list

Un altro caso comune è quello di più statement aventi lo stesso soggetto e anche lo stesso predicato, ma oggetti diversi. Anche qui Turtle fornisce una sintassi abbreviata, chiamata **object list**: dopo aver indicato (una volta sola) il soggetto e il predicato, si elencano i vari oggetti, separati da virgole.

Ad esempio, il grafo



può essere serializzato usando una object list,

```
<http://example.org/#spiderman> <http://example.org/name>
    "Spiderman", "Peter".
```

che equivale alle seguenti due triple:

```
<http://example.org/#spiderman> <http://example.org/name> "Spiderman".
<http://example.org/#spiderman> <http://example.org/name> "Peter".
```