**Cerca dati RDF con SPARQL**

**SPARQL e Jena Toolkit aprono al Web semantico**

Come più dati vengono memorizzati in formato RDF, come RSS, la necessità è sorta di un modo semplice per trovare informazioni specifiche. SPARQL, un nuovo potente linguaggio di query che riempie lo spazio, che rende facile trovare i dati necessari nel pagliaio RDF. Fate un giro di caratteristiche di SPARQL e imparare a utilizzare le query SPARQL dalle proprie applicazioni Java ™ con la Jena Semantic Web Toolkit.

[**Iniziare a costruire gratis**](https://www.bluemix.net/?cm_mmc=dw-_-bluemix-_-j-sparql-_-article)

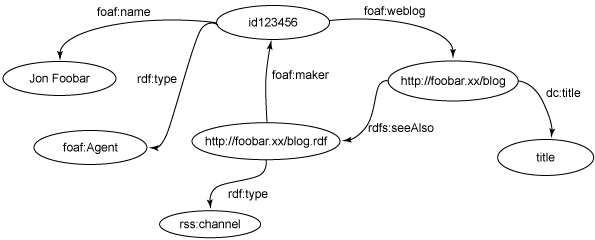
Il *Resource Description Framework,* o RDF, consente ai dati di essere decentralizzato e distribuito. Modelli RDF possono essere unite insieme facilmente, e RDF serializzato possono essere scambiati solo su HTTP.Le applicazioni possono essere debolmente accoppiati a molteplici fonti di dati RDF sul Web. A PlanetRDF.com, per esempio, noi sindacato blog di autori che forniscono il loro contenuto, come RDF in un feed RSS 1.0. Gli URL di feed degli autori sono a loro volta tenuti in un grafo RDF, chiamati *bloggers.rdf.*

Ma come si può trovare e manipolare i dati necessari entro grafici RDF? Il protocollo SPARQL e RDF Query Language (SPARQL) è attualmente in fase di discussione in un progetto di lavoro del W3C.SPARQL basa su precedenti linguaggi di interrogazione RDF, come rdfDB, RDQL, e SeRQL, e ha diverse nuove preziose caratteristiche proprie. In questo articolo, useremo i tre tipi di grafico RDF che guidano PlanetRDF - grafici FOAF descrivono collaboratori, la loro RSS 1.0 feed, ei blogger graph - per dimostrare alcune delle cose interessanti query SPARQL possono fare con i vostri dati. Esistono implementazioni di SPARQL per una varietà di piattaforme e linguaggi; questo articolo si concentrerà sulla Jena Semantic Web Toolkit per la piattaforma Java.

In questo articolo si presuppone una conoscenza di RDF, e familiarità con vocabolari RDF quali Dublin Core, FOAF, e RSS 1.0. Esso assume anche che si abbia una certa esperienza con la Jena Semantic Web Toolkit. Per arrivare fino a velocità su tutte queste tecnologie, controllare i collegamenti nella [Resources](http://www.ibm.com/developerworks/xml/library/j-sparql/" \l "resources) sezione sottostante.

**Anatomia di una semplice query SPARQL**

Cominciamo guardando il modello bloggers.rdf di PlanetRDF. E 'piuttosto semplice, utilizzando le FOAF e Dublino vocabolari core per fornire un nome, un titolo del blog e URL, e una descrizione RSS Feed per ciascun contribuente blog. La Figura 1 mostra la struttura del grafo di base per un singolo partecipante. Il modello completo ripete semplicemente questo struttura per ogni blog che ci aggreghiamo.



**Figura 1. Struttura di base grafico per un singolo contribuente in bloggers.rdf**

Ora diamo un'occhiata a una semplice query SPARQL sul modello blogger. La query, in inglese, dice: "Trova l'URL del blog da parte della persona di nome Jon Foobar," ed è mostrato nel Listato 1:

**Listato 1. SPARQL query per trovare l'URL del blog di un collaboratore**

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

SCEGLI? Url

FROM <bloggers.rdf>

DOVE {

? Collaboratore foaf: nome "Jon Foobar".

? Collaboratore foaf:? Weblog url.

}

La prima riga della query definisce semplicemente un PREFIX per lo spazio dei nomi FOAF, in modo da non dover digitare in pieno ogni volta che viene fatto riferimento. La SELEZIONA clausola specifica che cosa la query deve restituire - in questo caso, una variabile denominata url . Variabili SPARQL sono precedute da uno ? o $ - i due sono intercambiabili, ma ci limiteremo a ? in questo articolo. FROM è una clausola facoltativa che fornisce l'URI del dataset da utilizzare. Qui, è solo punta a un file locale, ma potrebbe anche indicare l'URL di un grafico da qualche parte sul web.Infine, il WHERE la clausola consiste in una serie di modelli triple, espresso utilizzando la sintassi basata su Turtle. Questi triple insieme costituiscono ciò che è noto come un *modello grafico.*

La query tenta di abbinare le triple del modello grafico per il modello. Ogni legame delle variabili del modello grafico per i nodi del modello corrispondente diventa una *soluzione di query,* ed i valori delle variabili denominate in SELEZIONA clausola di far parte dei risultati della query.

In questo esempio, la prima tripla della WHERE grafico modello di clausola corrisponde a un nodo con un foaf: nome di proprietà di "Jon Foobar," e si lega alla variabile denominata il contribuente . Nel modello bloggers.rdf, collaboratore corrisponderà foaf: agente blank-nodo in cima alla[figura 1](http://www.ibm.com/developerworks/xml/library/j-sparql/#fig1) . secondo corrispondenze triple del modello grafico l'oggetto della contributore s ' foaf: weblog di proprietà. Questo è legato alla URLvariabili, formando una soluzione di query.

**Utilizzando SPARQL con Jena**

Supporto SPARQL in Jena è attualmente disponibile tramite un modulo chiamato *ARQ.* Oltre a implementare SPARQL, motore di query di ARQ può anche analizzare le query espresse in RDQL o un suo linguaggio di query interna. ARQ è in fase di sviluppo attivo, e non è ancora parte della distribuzione standard Jena. Tuttavia, è disponibile sia da repository CVS di Jena o come download autonomo.

E 'semplice per ottenere ARQ attivo e funzionante. Basta prendere l'ultima distribuzione ARQ (vedi [Risorse](http://www.ibm.com/developerworks/xml/library/j-sparql/#resources) sezione qui sotto per un link), scompattarlo e impostare la variabile d'ambiente ARQROOT per puntare alla directory ARQ. Potrebbe anche essere necessario ripristinare lettura ed esecuzione autorizzazioni per i contenuti della directory bin ARQ. E 'conveniente per aggiungere la directory bin al percorso di esecuzione, in quanto contiene script wrapper per invocare ARQ dalla riga di comando. Per assicurarsi che siete tutti insieme, chiamare SPARQL da linea di comando, e assicuratevi di vedere il suo messaggio di utilizzo. Tutti questi passaggi sono illustrati nel Listato 2, che presuppone che si sta lavorando su una piattaforma UNIX-like, o con Cygwin sotto Windows. (ARQ anche navi con .bat script da utilizzare con Windows, ma il loro utilizzo può variare leggermente da esempio fatto qui.)

**Listato 2. Impostazione dell'ambiente da utilizzare Jena ARQ**

$ Export ARQROOT = ~ / ARQ-0.9.5

$ Chmod + rx $ ARQROOT / bin / \*

$ Export PATH = $ PATH: $ ARQROOT / bin

$ SPARQL

Uso: [URL --data] [exprString | di file --query]

**L'esecuzione di query SPARQL dalla riga di comando**

Ora siete pronti per eseguire una query SPARQL (Listato 3). Useremo il set di dati e query da [Listato 1](http://www.ibm.com/developerworks/xml/library/j-sparql/#list1) . Poiché la query utilizza il FROM parola chiave per specificare il grafico da utilizzare, è necessario solo per fornire la posizione del file query al SPARQL comando. Tuttavia, la query deve essere eseguito dalla directory che contiene il grafico, in quanto è specificato utilizzando un URL relativo alla query FROM clausola.

**Listato 3. Eseguire una query semplice con il comando SPARQL**

$ SPARQL --query jon-url.rq

----------------------------

| Url |

============================

| <Http: //foobar.xx/blog> |

----------------------------

A volte, ha senso omettere il FROM clausola da una query. Questo permette un grafico per essere passato alla query quando viene eseguito. Evitare vincolante il set di dati per la query è buona prassi quando si utilizza SPARQL dal codice applicativo - permette la stessa query per essere riutilizzato con diversi grafici, per esempio. Un grafico è specificato in fase di esecuzione nella riga di comando con il SPARQL URL --data opzione, dove URL è la posizione del grafico. Questo URL potrebbe essere o la posizione di un file locale o l'indirizzo Web di un grafico a distanza.

**L'esecuzione di query SPARQL con l'API Jena**

Mentre la linea di comando SPARQL strumento è utile per l'esecuzione di query standalone, applicazioni Java possono chiamare sulle capacità SPARQL di Jena direttamente. Query SPARQL sono creati ed eseguiti con Jena tramite classi nel com.hp.hpl.jena.query pacchetto. UtilizzandoQueryFactory è l'approccio più semplice. QueryFactory ha diverse creare () metodi per leggere una query di testo da un file o da una stringa .Questi creano () metodi restituiscono una query oggetto che incapsula una query analizzato.

Il passo successivo è quello di creare un'istanza di QueryExecution , una classe che rappresenta una singola esecuzione di una query. Per ottenere un QueryExecution , chiamare QueryExecutionFactory.create (query, modello) , passando il Query da eseguire e il modello per eseguire contro. Poiché i dati per la query è previsto a livello di programmazione, la query non ha bisogno di un FROM clausola.

Ci sono diversi metodi per eseguire QueryExecution , ogni esecuzione di un diverso tipo di query (vedi la barra laterale intitolata " [Altri tipi di query SPARQL](http://www.ibm.com/developerworks/xml/library/j-sparql/#side3) "per ulteriori informazioni). Per una semplice SELEZIONA interrogazione, chiamare execSelect () , che restituisce un ResultSet . IlResultSet consente di eseguire iterazioni su ogni QuerySolution restituito dalla query, fornitura di accesso a valore di ogni variabile legata. In alternativa, ResultSetFormatter può essere utilizzato per i risultati dell'interrogazione uscita in vari formati.

Listato 4 mostra un modo semplice per mettere questi passi insieme. Si esegue una query contro bloggers.rdf ed emette i risultati alla console.

**Listato 4. Eseguire una query semplice utilizzando API di Jena**

// Aprire i blogger RDF grafico dal filesystem

InputStream in = new FileInputStream (new File ("bloggers.rdf"));

// Creare un modello vuoto in-memory e compilarlo dal grafico

Modello Modello = ModelFactory.createMemModelMaker () CreateModel ().;

model.read (in, null); // Base nullo URI, dal modello URI sono assoluti

in.close ();

// Creare una nuova query

String queryString =

"PREFIX foaf: <http://xmlns.com/foaf/0.1/>" +

"SELECT? Url" +

"WHERE {" +

"Collaboratore foaf:? Nome \" Jon Foobar \ + "."

"? Collaboratore foaf: weblog url?". +

"}";

Query di Query = QueryFactory.create (queryString);

// Eseguire la query e ottenere risultati

QueryExecution qe = QueryExecutionFactory.create (query, modello);

Risultati ResultSet = qe.execSelect ();

// risultati delle query in uscita

ResultSetFormatter.out (System.out, risultati, query);

// importanti - liberare le risorse utilizzate in esecuzione della query

qe.close ();

**Affinamento risultati di una query**

Per perfezionare ulteriormente i risultati di una query, SPARQL ha DISTINCT , LIMIT , OFFSET , e ORDER BYparole chiave che funzionano più o meno come le loro controparti di SQL. DISTINCT può essere utilizzato solo con SELEZIONARE query, in forma SELECT DISTINCT .Questo eliminerà le soluzioni di query duplicate dal set di risultati, lasciando ogni soluzione che rimane unico.Le altre parole chiave sono tutti disposti dopo la WHEREla clausola di una query. LIMIT *n* ​​limita il numero di risultati restituiti da una query al *n,* mentre OFFSET *n*omette i primi *n* risultati. ORDER BY? var sarà ordinare i risultati per l'ordinamento naturale di ? var - lessicalmente se var è una stringa letterale, per esempio. ASC [var?] e DESC [var?] può essere utilizzato per specificare la direzione del genere.

Oppure, naturalmente, è possibile combinare DISTINCT, LIMIT , OFFSET e ORDER BY in una query. Ad esempio,ORDER BY DESC [? Data] LIMIT 10 potrebbe essere usato per trovare le dieci voci più recenti in un feed RSS.

**Scrivere query più complesse**

Finora hai visto due modi per eseguire una semplice query SPARQL: utilizzando la linea di comando SPARQL utility, e utilizzando il codice Java con l'API Jena. In questa sezione, ti presento più delle caratteristiche di SPARQL, e le query più complesse che consentono.

RDF viene spesso utilizzato per rappresentare *semi-strutturati* dati. Ciò significa che due nodi dello stesso tipo in un modello possono avere diversi insiemi di proprietà. Ad esempio, la descrizione di una persona in un modello FOAF può consistere solo un indirizzo e-mail; in alternativa, si potrebbe inserire un nome vero, soprannomi IRC, gli URL di foto raffiguranti l'individuo, e così via.

Listato 5 mostra un piccolo grafico FOAF, espressa nella sintassi Turtle. Contiene le descrizioni di quattro persone di fantasia, ma le descrizioni ciascuno avere diversi insiemi di proprietà.

**Listato 5. Un piccolo grafico FOAF descrivere quattro persone**

prefix foaf: <http://xmlns.com/foaf/0.1/>.

\_: Un foaf: nome "Jon Foobar";

foaf: mbox <mailto: jon@foobar.xx>;

foaf: rappresentazione <http: //foobar.xx/2005/04/jon.jpg>.

\_: B foaf: nome "AN O'Ther";

foaf: mbox <mailto: another@example.net>;

foaf: rappresentazione <http://example.net/photos/an-2005.jpg>.

\_: C foaf: nome "Liz Somebody";

foaf: mbox\_sha1sum "3f01fa9929df769aff173f57dec2fe0c2290aeea"

\_: D foaf: nome "M Benn";

foaf: rappresentazione <http: //mbe.nn/pics/me.jpeg>.

**Partite opzionali**

Si supponga di voler scrivere una query che restituisce il nome di ogni persona descritta dal grafico nel Listato 5, con un link a una foto per ciascuno, se è disponibile. A SELEZIONE query cui modello grafico incluso foaf: rappresentazione troverebbe solo tre soluzioni. Liz Qualcuno non formare una soluzione, perché ha un foaf: nome , ma no foaf: rappresentazione di proprietà, e ha bisogno sia per abbinare la query.

L'aiuto è a portata di mano in forma di del SPARQL OPTIONAL parola chiave. *blocchi opzionali* definiscono i modelli di grafici aggiuntivi che non causano soluzioni viene respinto se non sono abbinati, ma si legano al grafico quando possono essere abbinati. Listato 6 mostra una query che trova il foaf: nome di ogni persona nei dati FOAF nel Listato 5, e poi trova opzionalmente l'accompagnamento FOAF: rappresentazione .

**Listato 6. Interrogazione dati FOAF con un blocco opzionale**

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

SELEZIONA? Nome? Raffigurazione

DOVE {

? Persona foaf:? Nome nome.

OPTIONAL {

? Persona foaf:? Raffigurazione rappresentazione.

}.

}

Listato 7 mostra i risultati dell'esecuzione della query nel Listato 6. Mentre tutte le soluzioni di query contiene il nome della persona, il modello grafico opzionale è legato solo quando un foaf: rappresentazione esiste proprietà; altrimenti, esso è semplicemente omesso dalla soluzione. A questo proposito, la query è simile a un join esterno sinistro in SQL.

**Listato 7. Risultati della query da Listato 6**

-------------------------------------------------- ----------

| Nome | rappresentazione |

================================================== ==========

| "AN O'Ther" | <http://example.net/photos/an-2005.jpg> |

| "Jon Foobar" | <http: //foobar.xx/2005/04/jon.jpg> |

| "Liz Somebody" | |

| "M Benn" | <http: //mbe.nn/pics/me.jpeg> |

-------------------------------------------------- ----------

Un blocco opzionale può contenere qualsiasi modello grafico, non solo un singolo tripla come mostrato nel Listato 6. L'intero modello query in un blocco opzionale deve essere abbinato in modo che il modello opzionale per formare parte di una soluzione di query. Se una query ha più blocchi opzionali, essi agiscono indipendentemente l'uno dall'altro - ognuno può essere omesse, o presentare in una soluzione. Blocchi opzionali possono anche essere nidificati, nel qual caso il blocco opzionale interna viene considerato solo quando lo schema del blocco opzionale esterno corrisponde il grafico.

**Partite alternative**

Grafici FOAF utilizzano indirizzi e-mail delle persone a loro identificare in modo univoco. Per motivi di privacy, alcune persone preferiscono utilizzare codici hash di indirizzi e-mail. Testo Plain indirizzi e-mail sono espressi utilizzando l' foaf: mbox proprietà, mentre codici hash di indirizzi e-mail sono espressi utilizzando l' foaf: mbox\_sha1sum proprietà; i due sono di solito escludono in una descrizione FOAF di una persona. In situazioni come questa, è possibile utilizzare di SPARQL *partite alternativi* dispongono di scrivere query che restituiscono a seconda di quale delle proprietà è disponibile.

Partite alternativi sono definiti indicando più pattern grafico alternative, con la UNION parola chiave tra di loro. La query mostrato nel Listato 8 trova il nome di ogni persona nel grafico FOAF di [Listato 5](http://www.ibm.com/developerworks/xml/library/j-sparql/#list5) , insieme a due loro foaf: mbox o loro foaf: mbox\_sha1sum . M Benn non è una soluzione di query, perché ha né un foaf: mbox o un foaf: mbox\_sha1sum proprietà. In contrasto con OPTIONAL modelli grafico, *esattamente* una delle alternative deve corrispondere una soluzione di query; se entrambi i rami del UNION partita, verranno generate due soluzioni.

**Listing 8. Una query con i fiammiferi alternativi, ei suoi risultati**

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

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

SELEZIONA? Nome? Mbox

DOVE {

? Persona foaf:? Nome nome.

{

{Persona foaf:?? Mbox mbox} {UNIONE persona foaf:?? Mbox\_sha1sum mbox}

}

}

-------------------------------------------------- -------------------

| Nome | mbox |

================================================== ===================

| "Jon Foobar" | <mailto: jon@foobar.xx> |

| "AN O'Ther" | <mailto: another@example.net> |

| "Liz Somebody" | "3f01fa9929df769aff173f57dec2fe0c2290aeea" |

-------------------------------------------------- -------------------

**Vincoli di valore**

Il FILTER parola chiave in SPARQL limita i risultati di una query imponendo vincoli su valori di variabili legate. Questi vincoli di valore sono espressioni logiche che restituiscono valori booleani, e possono essere combinati con logiche && e || operatori. Ad esempio, una query che restituisce un elenco di nomi potrebbe essere modificato con un filtro per restituire solo nomi che corrispondono a una data espressione regolare. O, come mostrato nel Listato 9, gli elementi di un feed RSS pubblicati tra due date possono essere trovati con un filtro che pone limiti per date di pubblicazione articoli. Listato 9 mostra anche come funzione di fusione XPath stile di SPARQL è utilizzato (qui per lanciare la data di variabile in un XML Schema dateTime valore), e in che modo è possibile specificare lo stesso tipo di dati sulle stringhe letterali di data con ^^ xsd: dateTime .Questo assicura che il confronto data viene utilizzato nella query, piuttosto che il confronto stringa standard.

**Listato 9. Usando un filtro per recuperare elementi del feed RSS pubblicati nell'aprile 2005**

PREFIX rss: <http://purl.org/rss/1.0/>

Xsd PREFIX: <http://www.w3.org/2001/XMLSchema#>

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

SELEZIONA? ITEM\_TITLE? Pub\_date

DOVE {

L'articolo rss: title ITEM\_TITLE?.

? Elemento dc:? Data pub\_date.

FILTRO xsd: dateTime (? Pub\_date)> = "2005-04-01T00: 00: 00Z" ^^ xsd: dateTime &&

xsd: dateTime (? pub\_date) <"2005-05-01T00: 00: 00Z" ^^ xsd: dateTime

}

[**Torna in alto**](http://www.ibm.com/developerworks/xml/library/j-sparql/#ibm-content)

**Lavorare con più grafici**

Finora, tutte le query che ho dimostrato avere datasets coinvolti costituiti da un unico grafico RDF. Nella terminologia SPARQL, queste query hanno eseguito contro lo *sfondo grafico.* Il grafico di fondo è quello che viene specificato da una query di FROM clausola, dal SPARQL del comando --data switch o passando un modello per QueryExecutionFactory.create () quando si utilizza API di Jena .

**Altri tipi di query SPARQL**

Oltre ai SELEZIONARE query utilizzate in questo articolo, SPARQL supporta altri tre tipi di query. ASKsemplicemente restituisce "sì", se il grafico del modello della query ha tutte le corrispondenze nel set di dati, e "no" se non. DESCRIVERE ritorna un grafico contenente informazioni relative ai nodi corrispondenti nel modello grafico. Ad esempio, DESCRIVERE persona DOVE? {persona foaf:? nome di "Jon Foobar"} restituirebbe un grafico contenente triple dal modello su Jon Foobar.Infine, CONSTRUCT viene utilizzato per produrre un modello grafico per ogni soluzione di query. Questo permette un nuovo grafico RDF ​​da creare direttamente dai risultati della ricerca. Si può pensare a unCONSTRUCT query in un grafo RDF, come in qualche modo analogo ad una trasformazione XSL di dati XML.

Oltre al grafico sfondo, SPARQL può interrogare qualsiasi numero di *grafici nominati.* Questi grafici complementari sono identificati dai loro URI, e sono ciascuno distinto all'interno di una query. Prima di esaminare i modi in cui denominate grafici possono essere utilizzati, ti spiego come fornire loro di una query. Come con il grafico di fondo, chiamati i grafici possono essere specificati all'interno della query stessa, utilizzando FROM chiamato <URI> , dove URI specifica il grafico. In alternativa, denominati grafici possono essere forniti al SPARQL comando con URL --named , con l'URL dando la posizione del grafico. Infine, di Jena DataSetFactory classe può essere utilizzato per specificare denominati grafici da interrogare programmazione.

Grafici Named vengono utilizzati all'interno di una query SPARQL con la GRAPH parola chiave, in combinazione sia con un grafico URI o un nome di variabile. Questa parola chiave è seguita da un modello grafico per abbinare contro il grafico in questione.

**Trovare le partite in un grafico specifico**

Quando il GRAPH parola chiave viene usata con URI di un grafico (o una variabile già associato ad URI di un grafico), il modello grafico viene applicato a qualsiasi grafico viene identificato da tale URI. Se sono riscontrate nel grafico specificato, fanno parte di una soluzione di query. Nel Listato 10, due grafici FOAF denominati vengono passati alla query. Le soluzioni di query sono i nomi di quelle persone che si trovano in entrambi i grafici. Si noti che i nodi rappresentano le persone in ciascuno dei due grafici FOAF sono nodi vuoti, e hanno la portata solo nel grafico che li contiene. Ciò significa che lo stesso nodo di persona non può esistere in entrambi i grafici citati nella domanda, e così diverse variabili ( x e y ) deve essere utilizzato a rappresentarli.

**Listato 10. Una query per trovare le persone descritte in due grafici FOAF denominati**

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

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

SCEGLI? Nome

DA NAMED <jon-foaf.rdf>

DA NAMED <liz-foaf.rdf>

DOVE {

GRAFICO <jon-foaf.rdf> {

? X rdf: Tipo foaf: Person.

? X foaf: nome nome?.

}.

GRAFICO <liz-foaf.rdf> {

? Y rdf: Tipo foaf: Person.

? Y foaf: nome nome?.

}.

}

**Trovare il grafico che contiene un pattern**

L'altro modo di utilizzare GRAFICO è quello di seguire con una variabile non legato. In tal caso, il modello grafico viene applicata a ciascuno dei grafici denominati disponibili alla query. Se il modello corrisponde contro uno dei grafici di nome, quindi URI del grafico è legato al GRAPH variabili s '.Listing 11 di GRAFICO clausola corrisponde in ogni nodo persona nei grafici FOAF denominati dato alla query. Il nome della persona a te abbinata è legato al nome di variabile e il graph\_uri variabili lega al URI del grafico che ha trovato il modello. Sono mostrati i risultati della ricerca. Un nome, AN O'Ther, è abbinato due volte, perché quella persona è descritto in entrambi jon-foaf.rdf e liz-foaf.rdf.

**Listato 11. Determinare quale grafico descrive diverse persone**

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

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

SELEZIONA? Nome? Graph\_uri

DA NAMED <jon-foaf.rdf>

DA NAMED <liz-foaf.rdf>

DOVE {

GRAFICO? Graph\_uri {

? X rdf: Tipo foaf: Person.

? X foaf: nome nome?.

}

}

-------------------------------------------------- ------

| Nome | graph\_uri |

================================================== ======

| "Liz Somebody" | <file: //.../jon-foaf.rdf> |

| "AN O'Ther" | <file: //.../jon-foaf.rdf> |

| "Jon Foobar" | <file: //.../liz-foaf.rdf> |

| "AN O'Ther" | <file: //.../liz-foaf.rdf> |

-------------------------------------------------- ------

**I risultati delle query in formato XML**

SPARQL permette risultati della query di essere restituiti come XML, in un semplice formato noto come*SPARQL Risultati rilegatura variabili formato XML.*Questo schema definito agisce formato come un ponte tra le query RDF e strumenti XML e le librerie.

Ci sono un certo numero di potenziali usi per questa funzionalità. Si potrebbe trasformare i risultati di una query SPARQL in una pagina Web o RSS feed tramite XSLT, accedere ai risultati via XPath, o restituire il documento risultato a un SOAP o client AJAX. Per i risultati della query in uscita come XML, utilizzare ilResultSetFormatter.outputAsXML () il metodo, o specificare --results rs / xml nella riga di comando.

**Combinando i dati di sfondo e denominati grafici**

Le query possono inoltre utilizzare i dati di fondo insieme a denominati grafici. Listato 12 usa il live aggregato feed RSS da PlanetRDF.com come dati di fondo, e le query in combinazione con un grafico di nome contenente il mio profilo FOAF. L'idea è di creare un feed personalizzato trovando gli ultimi dieci messaggi di blogger che conosco. La prima parte della query trova il nodo che mi rappresenta nel file FOAF, e poi trova i nomi delle persone che il grafico dice che conosco. La seconda parte cerca voci del feed RSS che vengono creati da quelle persone. Infine, il set di risultati è ordinato per data di creazione degli oggetti ', e limitato a restituire solo dieci risultati.

**Listato 12. Ottenere un feed PlanetRDF vivo personalizzato**

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

PREFIX rss: <http://purl.org/rss/1.0/>

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

SELEZIONA? Titolo? Known\_name? Link

FROM <http://planetrdf.com/index.rdf>

DA NAMED <phil-foaf.rdf>

DOVE {

GRAFICO <phil-foaf.rdf> {

? Mi foaf: nome "Phil McCarthy".

? Mi foaf: sa known\_person?.

? Known\_person foaf:? Nome known\_name.

}.

? Elemento dc: creator known\_name.

? Elemento rss: title titolo.

Link Link: voce rss?.

? Elemento dc: data Data.

}

ORDER BY DESC [? Data] LIMITARE 10

Mentre questa query restituisce semplicemente un elenco di titoli, nomi e gli URL, una query più complessa potrebbe estrarre tutti i dati presenti nelle voci corrispondenti. Utilizzando i risultati in formato XML di SPARQL (vedi la barra laterale intitolata " [I risultati delle query in formato XML](http://www.ibm.com/developerworks/xml/library/j-sparql/#side2) ") in combinazione con un foglio di stile XSL, sarebbe possibile creare una visualizzazione HTML personalizzata degli articoli del blog, o addirittura produrre un altro feed RSS.

**Sommario**

Gli esempi in questo articolo dovrebbero aiutare a capire le caratteristiche fondamentali e la sintassi di SPARQL, insieme con i benefici che porta alle applicazioni RDF. Hai visto come SPARQL permette di avvicinarsi alla natura semi-strutturata di mondo reale grafici RDF, con l'aiuto di partite opzionali e alternative. Esempi di utilizzo di nome grafici vi hanno mostrato come la combinazione di molteplici grafici in SPARQL apre le opzioni di interrogazione. Avete anche visto come sia semplice per eseguire query SPARQL dal codice Java utilizzando API di Jena.

C'è molto di più di quanto SPARQL è possibile coprire qui, in modo da utilizzare le [risorse](http://www.ibm.com/developerworks/xml/library/j-sparql/#resources) qui sotto per saperne di più sulle caratteristiche di SPARQL. Guardate le specifiche SPARQL per conoscere nel dettaglio le funzioni di SPARQL built-in, gli operatori, le forme di query, e la sintassi, o per vedere molti più esempi di query SPARQL.

Naturalmente, il modo migliore per conoscere SPARQL è scrivendo alcune query di tua scelta. Basta prendere alcuni dati RDF dal Web, scaricare il modulo di Jena ARQ, e iniziare a sperimentare!