Azzolini Riccardo 2020-05-28

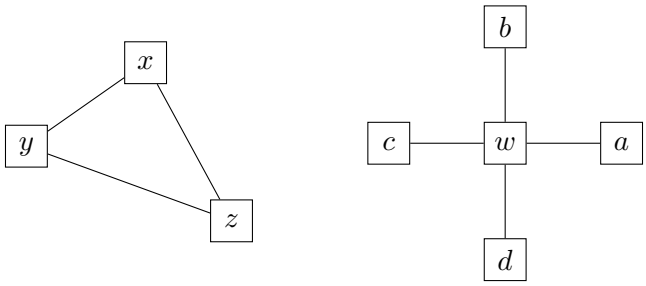
Interrogazioni basate su graph pattern e linguaggio Cypher

1 Complex graph pattern

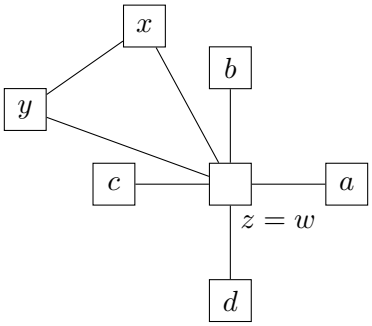
Come già anticipato, i linguaggi di interrogazione per grafi permettono di esprimere graph pattern più complessi, chiamati appunto **complex graph pattern** (cgp), costruiti a partire dai basic graph pattern (bgp) mediante appositi operatori, tra cui i principali sono:

- **proiezione** (di fatto analoga alla clausola `SELECT` di SQL);
- **join**;
- **unione**;
- **filter**, che permette di specificare dei predicati di selezione, tipicamente con connettivi quali la congiunzione, la disgiunzione, e possibilmente anche la negazione (in pratica, corrisponde al `WHERE` di SQL).

Tra questi operatori, quello che risulta meno immediato (nel contesto dei grafi) è il join. Il suo significato può essere illustrato mediante un esempio. Considerando i due bgp

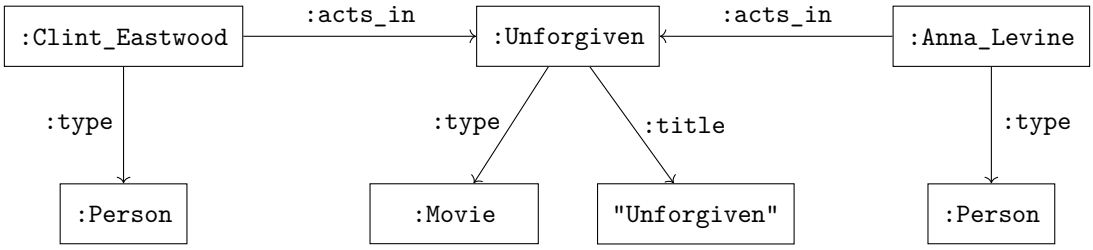


il loro join sulla condizione $x = w$ è il seguente graph pattern:

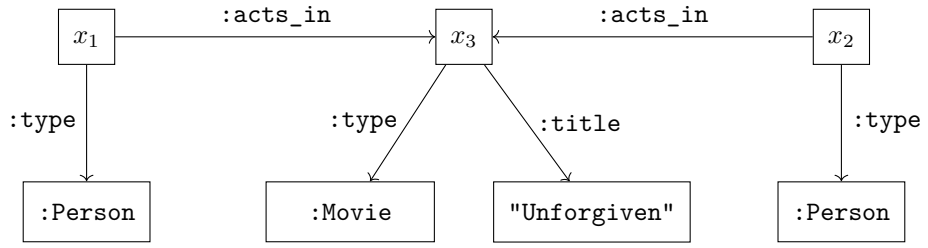


2 Esempio di complex graph pattern in SPARQL

Dato un edge-labelled graph (nello specifico, un grafo RDF) relativo ad attori e film,



si vogliono trovare tutte le coppie di attori che recitano nei film intitolati "Unforgiven". Il basic graph pattern corrispondente è:



La seguente interrogazione SPARQL specifica tale basic graph pattern, e lo combina con una proiezione (`SELECT`) che indica di restituire solo gli attori, e non il film:

```
SELECT ?x1 ?x2
WHERE {
  ?x1 :acts_in ?x3 ;
      :type :Person .
  ?x2 :acts_in ?x3 ;
```

```
      :type :Person .
  ?x3 :title "Unforgiven" ;
      :type :Movie .
}
```

Così, però, siccome SPARQL implementa il matching basato su omomorfismi, verranno restituite anche le "coppie" in cui è presente due volte lo stesso attore. Se le si vuole escludere, bisogna aggiungere esplicitamente una clausola `FILTER`:

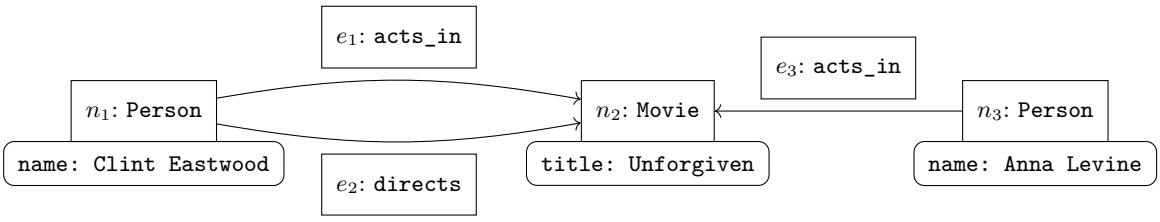
```
SELECT ?x1 ?x2
WHERE {
  ?x1 :acts_in ?x3 ;
      :type :Person .
  ?x2 :acts_in ?x3 ;
      :type :Person .
  ?x3 :title "Unforgiven" ;
      :type :Movie .
  FILTER (?x1 != ?x2)
}
```

3 Cypher

Cypher è il linguaggio di implementazione per property graph implementato dal DBMS graph-oriented *Neo4j*. Esso è un linguaggio dichiarativo (come del resto tutti i linguaggi d'interrogazione visti in precedenza), ed è anch'esso basato sul matching di graph pattern.

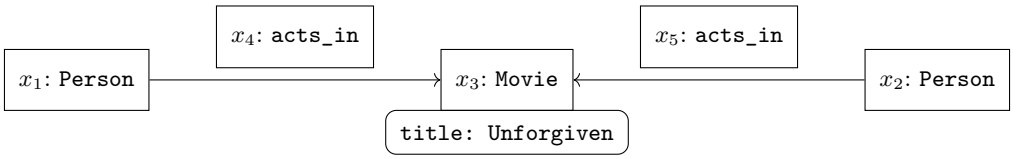
3.1 Esempio di bgp e proiezione

Come primo esempio di uso del linguaggio Cypher, si vogliono determinare ancora tutte le coppie di attori che recitano nei film intitolati "Unforgiven" (la stessa interrogazione appena mostrata in SPARQL), ma, questa volta, il grafo considerato è un property graph:[1]



[1] Questo grafo contiene alcune informazioni in più rispetto all'edge-labelled graph di prima: esse serviranno per l'esempio successivo.

Il bgp che esprime quest'interrogazione è:



La query si esprime in Cypher come segue:

```
MATCH (x1:Person) -[:acts_in]-> (:Movie {title:"Unforgiven"})
    <-[:acts_in]- (x2:Person)
RETURN x1, x2
```

Una particolarità di Cypher, già evidente in questo esempio, è che la sintassi cerca di "imitare" la rappresentazione grafica di un grafo:

- i nodi vengono specificati tra parentesi tonde;
- gli archi sono specificati tra parentesi quadre "inserite" in una freccia, `-[]->` o `<-[]-` (a seconda della direzione dell'arco).

La specifica di un nodo/arco è composta da tre elementi, tutti opzionali:

- una variabile;
- un'etichetta, scritta dopo i due punti: il nodo del graph pattern potrà corrispondere solo ai nodi del grafo interrogato che hanno quella stessa etichetta;
- i valori delle proprietà, indicati tra parentesi graffe.

Infine, la clausola `RETURN` esegue una proiezione, specificando quali variabili restituire (ma si ha anche una proiezione implicita, ottenuta omettendo le variabili relative certi nodi/archi).

Siccome il matching avviene con la semantica no-repeated-edge, vengono automaticamente escluse le coppie formate da un attore con sé stesso, per le quali i due archi `-[:acts_in]->` del pattern dovrebbero infatti essere mappati su uno stesso arco del grafo interrogato.

3.2 Esempio di union

Per mostrare un complex graph pattern più elaborato, considerando il property graph dei film dell'esempio precedente, si vogliono trovare i titoli di tutti i film *diretti o interpretati* da Clint Eastwood:

```
MATCH (:Person {name:"Clint Eastwood"}) -[:acts_in]-> (x3:Movie)
RETURN x3.title
UNION ALL
MATCH (:Person {name:"Clint Eastwood"}) -[:directs]-> (x3:Movie)
RETURN x3.title
```

Per ottenere sia i film diretti che quelli interpretati da Clint Eastwood, si usa un cgp basato sull'operatore di unione. In particolare, la parola chiave `ALL` indica di non eliminare i valori duplicati: così, ad esempio, "Unforgiven" comparirà due volte nel risultato, perché esso è sia diretto che interpretato da Clint Eastwood.

4 Cenni alle interrogazioni di navigazione

Oltre ai graph pattern, esiste un'altra categoria di interrogazioni: quelle basate sulla navigazione di grafi.

Ad esempio, dato un grafo relativo ad attori e film, un'interrogazione di questo tipo permetterebbe di trovare tutte le coppie di attori che hanno un "percorso di collaborazione" (se a_1 ha recitato in un film insieme ad a_2, a_2 ha recitato insieme ad a_3, e così via, fino ad a_n, allora esiste un percorso di collaborazione tra a_1 e a_n).