

Neo4j

GRAPH DATABASE

Neo4j

Database a grafo

- <https://neo4j.com/download/other-releases/>

GUI di base

- Avviare docker e i database
- <http://127.0.0.1:7474/>
 - User: neo4j
 - Pwd: fitstic

Completo di tutorial per iniziare ad utilizzarlo

[Obsoleto] Installazione

Database a grafo

- <https://neo4j.com/download/other-releases/>

GUI di base

- <http://127.0.0.1:7474/browser/>
- Password iniziale: neo4j

Completo di tutorial per iniziare ad utilizzarlo

[Obsoleto] Installazione

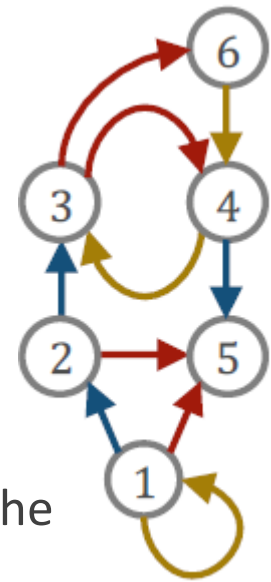
Istruzioni

1. Scompattare il file .zip della Community Edition
2. Installare il servizio da linea di comando (CMD):
 - Con CMD, posizionarsi dentro alla cartella **neo4j/bin**
 - Installare col comando **neo4j install-service**
3. Controllare tra i servizi di Windows che il servizio sia installato e avviato
4. Collegarsi col browser all'indirizzo <http://localhost:7474/>
 - Username e password: **neo4j**
5. Al primo accesso bisogna cambiare la password
 - Suggerimento password: **fitstic**

Database a grafo - Concetti

In un database a grafo esistono tre concetti fondamentali:

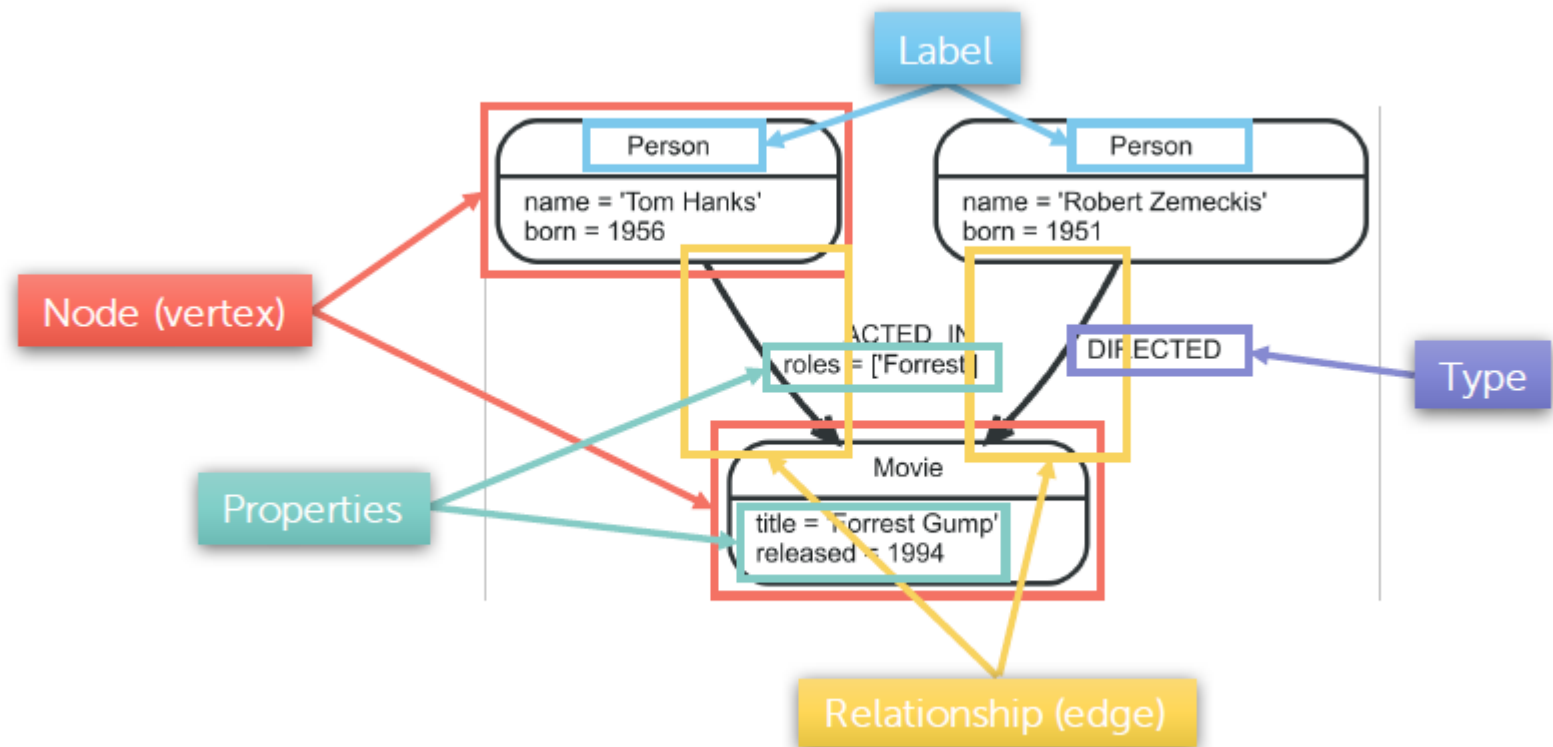
- Nodi: record, unità di dati
- Relazioni (o archi): collegamenti **direzionati** tra i nodi
- Proprietà: valori (con una determinata etichetta) associati ad un nodo o ad una relazione



Le relazioni sono puntatori contenuti dentro ad un nodo e che rimandano direttamente ad un altro nodo

- Meccanismo molto diverso da quello di foreign key negli RDBMS
- Molto più efficiente per determinati tipi di interrogazione

Database a grafo - Concetti



Database a grafo - Concetti

Percorso

- Sequenza di archi distinti che connettono due nodi

Cammino

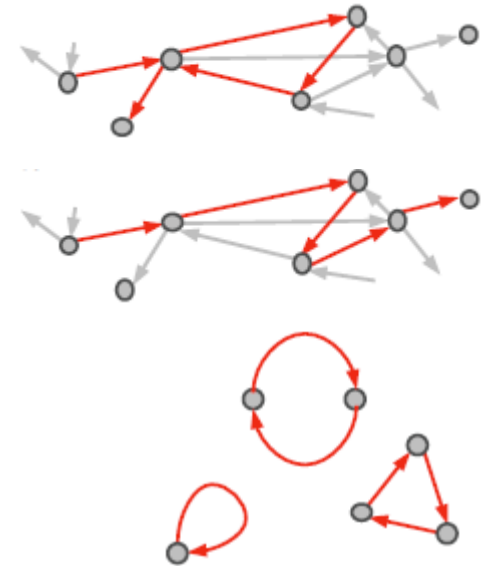
- Percorso che passa attraverso nodi distinti

Ciclo

- Cammino che inizia e termina nello stesso nodo

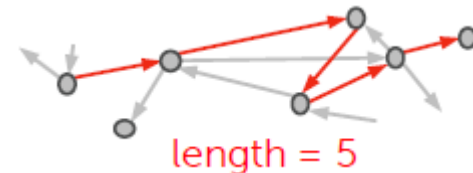
Distanza tra due nodi

- Numero minimo di archi che collegano due nodi



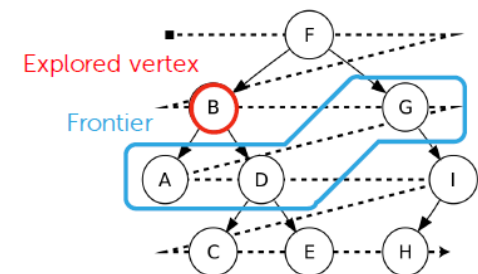
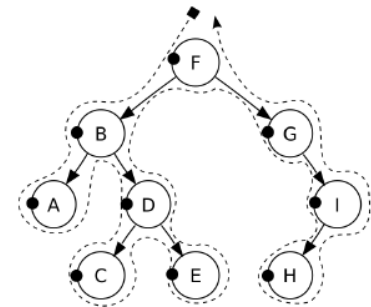
Database a grafo - Concetti

Una delle query più utilizzate è quella della ricerca del cammino più breve tra due nodi (shortest path)



Due metodi principali:

- In profondità (depth-first search)
 - Esamina tutti i nodi figli prima di esaminare i nodi fratelli
 - Richiede meno risorse
 - Occorre esaminare tutto il grafo per trovare la soluzione giusta
- In ampiezza (breadth-first search)
 - Esamina tutti i nodi fratelli prima di esaminare i nodi figli
 - Richiede più risorse
 - La prima soluzione che trova è quella giusta



Database a grafo - Concetti

Un'altra interrogazione frequente è quella per definire delle misure di centralità

Betweenness centrality (A)

- Numero di cammini più brevi tra due nodi che passano da un certo nodo

Closeness centrality (B)

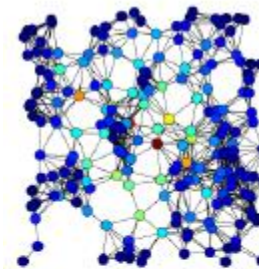
- Somma delle distanze da tutti gli altri nodi

Eigenvector centrality (C)

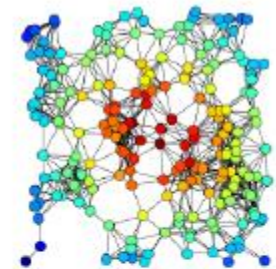
- Il punteggio di un nodo è influenzato dal punteggio dei nodi adiacenti (Page rank)

Degree centrality (D)

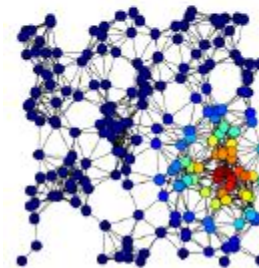
- Numero di nodi adiacenti



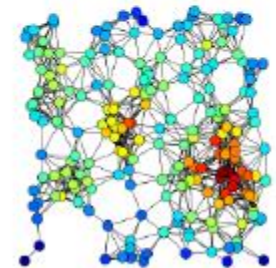
A



B



C



D

Query language: Cypher

Due clausole principali: **match** e **return**

Match

- Clausola primaria per estrapolare dati
- Permette di specificare dei pattern
- Possibile utilizzare più clausole di match
- Corrisponde (più o meno) alla combinazione di WHERE e JOIN in SQL

Return

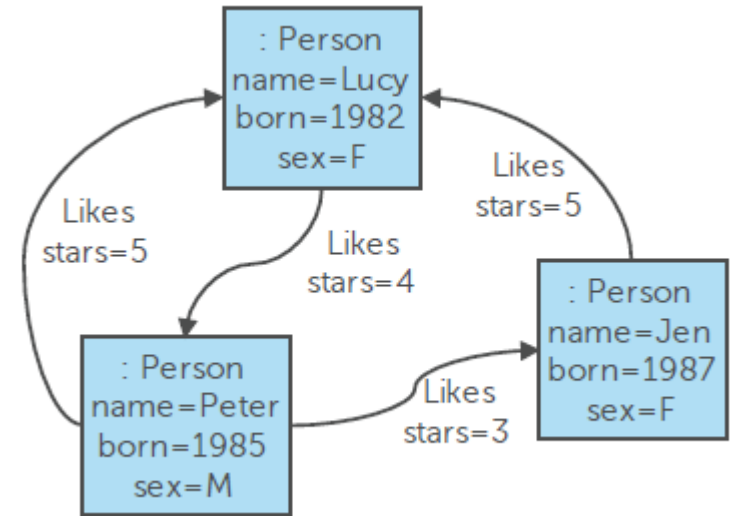
- Clausola per indicare i dati da restituire (nodi, archi, proprietà, espressioni)
- Un'unica clausola per query
- Corrisponde alla SELECT in SQL

Cypher - Esempi

Esempi

- `MATCH (p:Person)-[:Likes]->(f:Person)`
`RETURN p.name, f.sex`

p.name	f.sex
Lucy	M
Peter	F
Jen	F
Peter	F



- `MATCH (p:Person)-[:Likes]->(:Person) -[:Likes]->(fof:Person)`
`RETURN p.name, fof.name`

p.name	fof.name
Lucy	Jen
Peter	Lucy
Peter	Peter
Jen	Peter
Lucy	Lucy

Cypher – Sintassi dei pattern

Sintassi per i nodi

- ()
nodo non identificato
- (matrix)
nodo identificato dalla variabile *matrix*
- (:Movie)
nodo non identificato di classe Movie
- (matrix:Movie:Action)
nodo con classi Movie e Action identificato dalla variabile *matrix*
- (matrix:Movie {title: "The Matrix"})
+ con una proprietà *title* uguale a "The Matrix"
- (matrix:Movie {title: "The Matrix", released: 1997})
+ con una proprietà *released* uguale a 1997

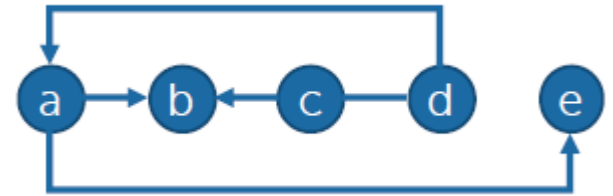
Sintassi per gli archi

- -->
arco non identificato
- --
arco non identificato senza direzione
- -[role]->
arco identificato dalla variabile *role*
- -[:ACTED_IN]->
arco non identificato di classe ACTED_IN
- -[role:ACTED_IN]->
arco di classe ACTED_IN identificato dalla variabile *role*
- -[role:ACTED_IN {roles: ["Neo"]}]>
+ con proprietà *roles* che contiene "Neo"

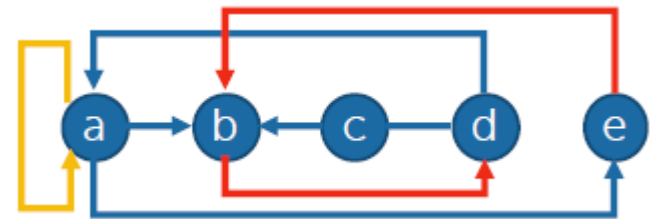
Cypher – Sintassi dei pattern

Sintassi per i percorsi

- Un percorso è una stringa in cui si alternano nodi ed archi
- Un percorso inizia e termina sempre con un nodo
- `(a)-->(b)<--(c)--(d)-->(a)-->(e)`
- `(keanu:Person:Actor {name: "Keanu Reeves"})
- [role:ACTED_IN {roles: ["Neo"]}]->
(matrix:Movie {title: "The Matrix"})`



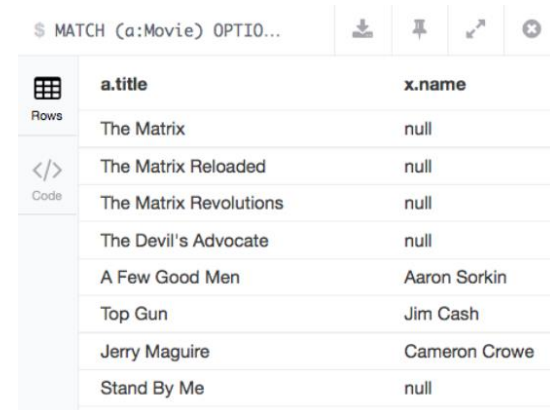
- Si possono specificare più percorsi, a patto che siano collegati da almeno una variabile condivisa
- `(a)-->(b)<--(c)--(d)-->(a)-->(e),
(e)-->(b)-->(d),
(a)-->(a)`



Match opzionale & Where

Clausola di Match opzionale

- Funziona come un left outer join
- Se il pattern non ha una corrispondenza, restituisce null
- MATCH (a:Movie)
OPTIONAL MATCH (a)-[:WROTE]-(x)
RETURN a.title, x.name



	a.title	x.name
Rows	The Matrix	null
	The Matrix Reloaded	null
	The Matrix Revolutions	null
	The Devil's Advocate	null
	A Few Good Men	Aaron Sorkin
	Top Gun	Jim Cash
	Jerry Maguire	Cameron Crowe
	Stand By Me	null

Clausola Where

- Aggiunge delle condizioni che devono essere rispettate dal pattern
- Più espressivo delle condizioni che possono essere specificate nella Match
- MATCH (n)
WHERE n.name = 'Matteo' XOR (n.age < 30 AND n.name = 'Enrico')
OR NOT (n.name ~='Enr.*' OR n.name CONTAINS 'att')
RETURN n

Percorsi di lunghezza variabile

E' possibile seguire in maniera iterativa lo stesso tipo di arco specificando quanti "salti" si vogliono fare

- Il carattere * precede la dichiarazione della lunghezza
 - (a)-[:x*2]->(b) Esattamente due salti: (a)-[:x]->()-[:x]->(b)
 - (a)-[*3..5]->(b) Minimo 3, massimo 5
 - (a)-[*3..]->(b) Minimo 3
 - (a)-[*..5]->(b) Massimo 5
 - (a)-[*]->(b) Nessun limite
- Un esempio completo
 - MATCH (me)-[:KNOWS*1..2]->(remote_friend)
WHERE me.name = "Enrico" RETURN remote_friend.name
 - Restituisce gli amici diretti e gli amici di amici
 - Attenzione: se un amico diretto e anche amico di amici, verrà restituito due volte!

Percorsi di lunghezza variabile

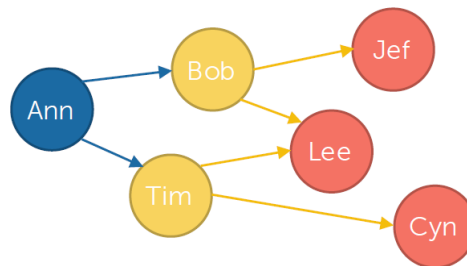
E' possibile cercare il percorso più breve tra due nodi

- MATCH (m { name:"Martin Sheen" }),
 (o { name:"Oliver Stone" }),
 p = shortestPath((m)-[*..15]-(o))
RETURN p

Aggregazione

La clausola di group-by è implicita

- Le espressioni nella RETURN senza funzioni di aggregazione sono chiavi di raggruppamento
- Le espressioni nella RETURN con funzioni di aggregazione producono aggregati
- MATCH (me:Person {name:'Ann'})-->(friend:Person)-->(friend_of_friend:Person)
RETURN me.name, count(DISTINCT friend_of_friend), count(friend_of_friend)



Result

me	COUNT DISTINCT	COUNT
Ann	3	4