Neo4j / Cypher Riferimento Rapido

Query su grafi, nodi, relazioni, pattern

Basi di Cypher

Struttura delle Query

MATCH	Trova pattern nel grafo
WHERE	Filtra i risultati
RETURN	Specifica le colonne di output
CREATE	Crea nodi e relazioni
SET / REMOVE	Aggiorna proprietà ed etichette
DELETE / DETACH DELETE	Rimuove nodi e relazioni

Eseguire Query

```
// Neo4j Browser: incolla ed esegui con Ctrl+Enter
// cypher-shell:
cypher-shell -u neo4j -p secret "MATCH (n) RETURN n LIMIT 5"
```

Nodi ed Etichette

Sintassi dei Nodi

```
(n)                    // nodo anonimo
(p:Person)             // nodo con etichetta
(p:Person:Employee)    // etichette multiple
(p:Person {name: "Alice", age: 30})
```

Operazioni sulle Etichette

```
SET n:Active           // aggiunge etichetta
REMOVE n:Active        // rimuove etichetta
MATCH (n) RETURN labels(n)  // elenca etichette
```

Vincoli e Indici

```
CREATE CONSTRAINT FOR (p:Person)
  REQUIRE p.email IS UNIQUE
CREATE INDEX FOR (p:Person) ON (p.name)
SHOW INDEXES
```

Relazioni

Sintassi delle Relazioni

```
-[r]->                 // diretta (uscente)
<-[r]-                 // diretta (entrante)
-[r]-                  // non direzionata
-[:KNOWS]->            // relazione tipizzata
-[r:KNOWS {since: 2020}]->  // con proprietà
```

Percorsi a Lunghezza Variabile

```
-[:KNOWS*2]->          // esattamente 2 salti
-[:KNOWS*1..3]->       // da 1 a 3 salti
-[:KNOWS*]->           // qualsiasi numero di salti
shortestPath((a)-[*]-(b))  // percorso più breve
```

CREATE

Crea Nodi

```
CREATE (p:Person {name: "Alice", age: 30})
CREATE (p:Person {name: "Bob"})
RETURN p
```

Crea Relazioni

```
MATCH (a:Person {name: "Alice"})
MATCH (b:Person {name: "Bob"})
CREATE (a)-[:KNOWS {since: 2020}]->(b)
```

MERGE (Upsert)

```
MERGE (p:Person {email: "alice@example.com"})
ON CREATE SET p.name = "Alice", p.created = date()
ON MATCH SET p.lastSeen = date()
```

MATCH

Pattern di Base

```
MATCH (p:Person) RETURN p
MATCH (p:Person)-[:KNOWS]->(f) RETURN p, f
MATCH (a)-[r]->(b) RETURN type(r), a, b
```

OPTIONAL MATCH

```
// Restituisce null per corrispondenze mancanti (come LEFT JOIN)
MATCH (p:Person)
OPTIONAL MATCH (p)-[:OWNS]->(c:Car)
RETURN p.name, c.model
```

Comprensione dei Pattern

```
MATCH (p:Person)
RETURN p.name,
  [(p)-[:KNOWS]->(f) | f.name] AS friends
```

WHERE

Confronto e Logica

```
WHERE p.age > 25
WHERE p.age >= 18 AND p.active = true
WHERE p.name <> "Bob" OR p.role = "admin"
WHERE NOT (p)-[:BLOCKED]->()
```

Predicati su Stringhe e Liste

```
WHERE p.name STARTS WITH "Al"
WHERE p.name CONTAINS "ice"
WHERE p.name =~ "(?i)alice.*"   // regex
WHERE p.age IN [25, 30, 35]
```

Controlli su Null e Esistenza

```
WHERE p.email IS NOT NULL
WHERE p.phone IS NULL
WHERE EXISTS { (p)-[:KNOWS]->(:Person) }
```

RETURN

Opzioni di Output

```
RETURN p.name AS name, p.age AS age
RETURN DISTINCT p.city
RETURN p, collect(f) AS friends
RETURN count(*) AS total
```

Ordinamento e Paginazione

```
RETURN p.name ORDER BY p.age DESC
RETURN p SKIP 10 LIMIT 5
```

UNWIND

```
// Espande una lista in righe
UNWIND [1, 2, 3] AS x RETURN x
UNWIND $names AS name
MERGE (p:Person {name: name})
```

UPDATE e DELETE

SET Proprietà

```
MATCH (p:Person {name: "Alice"})
SET p.age = 31, p.updated = date()
SET p += {city: "NYC", active: true}
```

REMOVE

```
MATCH (p:Person {name: "Alice"})
REMOVE p.temp_field     // rimuove proprietà
REMOVE p:Inactive       // rimuove etichetta
```

DELETE

```
MATCH (p:Person {name: "Bob"})
DETACH DELETE p          // elimina nodo + tutte le rel
// DELETE p              // fallisce se il nodo ha relazioni
MATCH ()-[r:OLD_REL]->() DELETE r  // elimina relazione
```

Aggregazione

Funzioni di Aggregazione

count(x)	Numero di valori non nulli
sum(x)	Somma di valori numerici
avg(x)	Media di valori numerici
min(x) / max(x)	Valore minimo / massimo
collect(x)	Aggrega valori in una lista
percentileCont(x, 0.5)	Percentile continuo

GROUP BY (Implicito)

```
// Le colonne non aggregate diventano chiavi di raggruppamento
MATCH (p:Person)-[:LIVES_IN]->(c:City)
RETURN c.name, count(p) AS population
ORDER BY population DESC
```

WITH (Aggregazione Concatenata)

```
MATCH (p:Person)-[:KNOWS]->(f)
WITH p, count(f) AS friendCount
WHERE friendCount > 5
RETURN p.name, friendCount
```

Pattern Comuni

Trova Amici in Comune

```
MATCH (a:Person {name:"Alice"})-[:KNOWS]->(m)<-[:KNOWS]-(b:Person
{name:"Bob"})
RETURN m.name AS mutualFriend
```

Raccomandazione (Amici degli Amici)

```
MATCH (p:Person {name:"Alice"})-[:KNOWS*2]-(fof)
WHERE NOT (p)-[:KNOWS]-(fof) AND p <> fof
RETURN DISTINCT fof.name
```

Importa Dati CSV

```
LOAD CSV WITH HEADERS FROM 'file:///people.csv' AS row
MERGE (p:Person {id: row.id})
SET p.name = row.name, p.age = toInteger(row.age)
```

Informazioni sul Database

```
CALL db.labels()        // elenca tutte le etichette
CALL db.relationshipTypes()  // elenca i tipi di relazione
CALL db.schema.visualization()
```