Referência Rápida de Neo4j / Cypher

Consultas em banco de dados de grafo, nós, relacionamentos, padrões

Básico do Cypher

Estrutura da Consulta

MATCH	Encontrar padrões no grafo
WHERE	Filtrar resultados
RETURN	Especificar colunas de saída
CREATE	Criar nós e relacionamentos
SET / REMOVE	Atualizar propriedades e labels
DELETE / DETACH DELETE	Remover nós e relacionamentos

Executando Consultas

```
// Neo4j Browser: paste and run with Ctrl+Enter
// cypher-shell:
cypher-shell -u neo4j -p secret "MATCH (n) RETURN n LIMIT 5"
```

Nós e Labels

Sintaxe de Nó

```
(n)                    // anonymous node
(p:Person)             // node with label
(p:Person:Employee)    // multiple labels
(p:Person {name: "Alice", age: 30})
```

Operações com Label

```
SET n:Active           // add label
REMOVE n:Active        // remove label
MATCH (n) RETURN labels(n)  // list labels
```

Restrições e Índices

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

Relacionamentos

Sintaxe de Relacionamento

```
-[r]->                 // directed (outgoing)
<-[r]-                 // directed (incoming)
-[r]-                  // undirected
-[:KNOWS]->            // typed relationship
-[r:KNOWS {since: 2020}]->  // with properties
```

Caminhos de Comprimento Variável

```
-[:KNOWS*2]->          // exactly 2 hops
-[:KNOWS*1..3]->       // 1 to 3 hops
-[:KNOWS*]->           // any number of hops
shortestPath((a)-[*]-(b))  // shortest path
```

CREATE

Criar Nós

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

Criar Relacionamentos

```
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

Padrões Básicos

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

OPTIONAL MATCH

```
// Returns null for missing matches (like LEFT JOIN)
MATCH (p:Person)
OPTIONAL MATCH (p)-[:OWNS]->(c:Car)
RETURN p.name, c.model
```

Compreensão de Padrão

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

WHERE

Comparação e Lógica

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

Predicados de String e Lista

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

Verificações de Nulo e Existência

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

RETURN

Opções de Saída

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

Ordenação e Paginação

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

UNWIND

```
// Expand a list into rows
UNWIND [1, 2, 3] AS x RETURN x
UNWIND $names AS name
MERGE (p:Person {name: name})
```

ATUALIZAR e EXCLUIR

SET de Propriedades

```
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     // remove property
REMOVE p:Inactive       // remove label
```

DELETE

```
MATCH (p:Person {name: "Bob"})
DETACH DELETE p          // delete node + all rels
// DELETE p              // fails if node has rels
MATCH ()-[r:OLD_REL]->() DELETE r  // delete rel
```

Agregação

Funções de Agregação

count(x)	Número de valores não-nulos
sum(x)	Soma de valores numéricos
avg(x)	Média de valores numéricos
min(x) / max(x)	Valor mínimo / máximo
collect(x)	Agregar valores em uma lista
percentileCont(x, 0.5)	Percentil contínuo

GROUP BY (Implícito)

```
// Non-aggregated columns become grouping keys
MATCH (p:Person)-[:LIVES_IN]->(c:City)
RETURN c.name, count(p) AS population
ORDER BY population DESC
```

WITH (Agregação Encadeada)

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

Padrões Comuns

Encontrar Amigos em Comum

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

Recomendação (Amigos de Amigos)

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

Importar Dados 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)
```

Informações do Banco de Dados

```
CALL db.labels()        // list all labels
CALL db.relationshipTypes()  // list rel types
CALL db.schema.visualization()
```