

Capítulo 6 – Neo4j

Dia 2 – REST, Indexes e Algoritmos


No segundo dia com o Neo4j, avançamos para recursos que permitem expor dados de forma mais eficiente e executar análises complexas em grafos.

Acesso via REST API

O Neo4j expõe suas operações também como uma **API REST**, o que facilita integração com outras aplicações.

Exemplo de requisição (usando `curl`):

```
curl -H "Content-Type: application/json"
  -d '{"statements": [{"statement": "MATCH (n:Pessoa) RETURN n LIMIT 5"}]}'
  http://localhost:7474/db/data/transaction/commit
```

 Resposta vem em **JSON**, ideal para aplicações web/mobile.

Indexes


Índices tornam buscas mais rápidas, evitando varreduras completas no grafo.

Criando índice para nomes de pessoas:

```
CREATE INDEX FOR (p:Pessoa) ON (p.nome)
```

Removendo índice:

```
DROP INDEX nome_index
```

 Índices são fundamentais em bases grandes para acelerar consultas.

Consultas Avançadas

- Buscar amigos em comum:

```
MATCH (a:Pessoa)-[:AMIGO_DE]->(c:Pessoa)<-[:AMIGO_DE]-(b:Pessoa)
WHERE a.nome = "Douglas" AND b.nome = "Ana"
RETURN c.nome
```

- Caminho mais curto entre duas pessoas:

```
MATCH p = shortestPath((a:Pessoa {nome:"Douglas"})-[:AMIGO_DE*]-(b:Pessoa {nome:"Carlos"}))
RETURN p
```



Algoritmos de Grafos

O Neo4j possui bibliotecas de algoritmos prontos (Graph Data Science Library – GDS). Exemplos:

- **PageRank** (influência em redes):

```
CALL gds.pageRank.stream({
  nodeProjection: 'Pessoa',
  relationshipProjection: 'AMIGO_DE'
})
YIELD nodeId, score
RETURN gds.util.asNode(nodeId).nome AS pessoa, score
ORDER BY score DESC
```

- **Comunidades (Louvain):**

```
CALL gds.louvain.stream({
  nodeProjection: 'Pessoa',
  relationshipProjection: 'AMIGO_DE'
})
YIELD nodeId, communityId
RETURN gds.util.asNode(nodeId).nome AS pessoa, communityId
```




Esses algoritmos são úteis para análise de redes sociais, detecção de clusters e influência.



Resumo do Dia 2

- API REST → integração fácil com sistemas externos.
- Índices → aceleram buscas por propriedades.
- Consultas avançadas em Cypher → amigos em comum, caminhos mínimos.
- Algoritmos de grafos → insights profundos (influência, comunidades, rotas).

 No próximo passo: **Dia 3 – Distributed High Availability**, onde veremos como o Neo4j garante escalabilidade e tolerância a falhas em ambientes distribuídos.