

BANCOS DE DADOS II

Cap.3 - Bancos de Dados Orientados a
Grafos



Prof. Me. Renzo P. Mesquita

Objetivos

- Compreendermos características de BDs NoSQL Orientados a Grafos;
- Trabalhar este conceito por meio do BD Neo4j e sua ferramenta para visualização, busca e manipulação de dados: o Neo4j Browser;
- Aprender recursos essenciais da linguagem Cypher, utilizada para definição, manipulação e consultas em BDs Orientados a Grafos;



Capítulo 3

Bancos de Dados Orientados a Grafos

- 3.1. O que são grafos?**
- 3.2. Bancos Orientados a Grafos e o Neo4j;**
- 3.3. Primeiras Impressões com o Neo4j Sandbox;**
- 3.4. Criando Nós e Relacionamentos;**
- 3.5. Excluindo e Atualizando elementos nos Grafos;**
- 3.6. Fundamentos de Buscas: Nós e Relacionamentos;**

3.1. O que é um Grafo?

Em sua forma mais simples, o GRAFO nada mais é que uma estrutura gráfica formada por uma coleção de NÓS (NODES) e RELACIONAMENTOS (RELATIONSHIPS).

Exemplo:



:Pessoa



:Casa



:Veiculo
:Carro

- Um Nô nada mais é que uma "coisa" no mundo real, como por exemplo o Julius, sua casa e seu carro;
- Um Nô é formado por uma ou mais etiquetas (Labels), utilizadas para identificar o seu tipo e facilitar em buscas futuras.

3.1. O que é um Grafo?

Nós existem por si só, já os Relacionamentos existem para conectar estes Nós de alguma forma.

Exemplo:

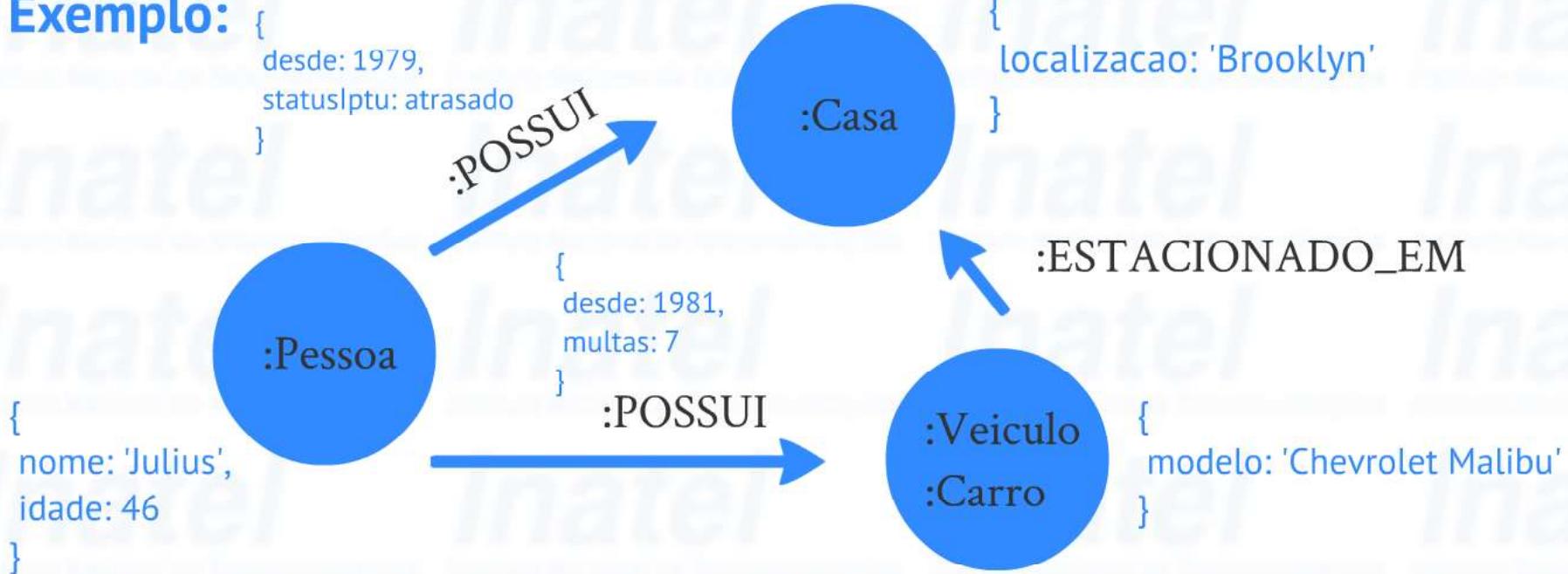


- Todo Relacionamento deve possuir uma direção;
- Deve possuir um Tipo (**Type**), como as palavras POSSUI e ESTACIONADO_EM;
- Diferente dos Nós que podem ter múltiplos Labels, Relacionamentos devem possuir um único Tipo;

3.1. O que é um Grafo?

Mas como saber que a Pessoa neste exemplo se chama 'Julius', o carro é um 'Chevrolet Malibu' e a Casa se localiza no 'Brooklyn'? É aí que entra o conceito de PROPRIEDADES (PROPERTIES).

Exemplo:



- Propriedades são pares de CHAVE:VALOR (KEY:VALUE) que podem ser adicionados a Nós e Relacionamentos;
- Cada Nô ou Relacionamento pode ter múltiplas Propriedades;
- Os Valores das Propriedades podem ser do tipo String, Boolean, Number, List, entre outros que veremos em breve.

3.1. O que é um Grafo?

Labels de Nós, Tipos de Relacionamentos e Propriedades são Case Sensitive, por exemplo, 'nome' é diferente de 'Nome'. É altamente recomendável seguir as convenções citadas abaixo para criação dos nossos Grafos daqui pra frente:

ENTIDADE	ESTILO RECOMENDADO	EXEMPLO
NÓ (NODE)	: (dois pontos), seguido de palavras juntas em Camel Case, começando com uma letra maiúscula.	:NoPessoa
RELACIONAMENTO (RELATIONSHIP)	: (dois pontos), seguido de palavra em Upper Case, usando _ para separar palavras.	:ESTACIONADO_EM
PROPRIEDADE (PROPERTY)	Palavras juntas em Camel Case, começando com uma letra minúscula.	statusIptu

3.1. O que é um Grafo?

Exercício Proposto

Agora que já possui uma noção de como um Grafo é estruturado, crie um Grafo para guardar informações de uma loja de artigos Geek.

Possíveis Nós, Relacionamentos e Propriedades que o Grafo deve representar:

- Alguns **Produtos da Loja**: Lancheira do Homem Aranha com preço, capacidade em litros e acessórios; Pacote de Fraldas do Superhomem com preço e a sua quantidade de unidades; Escova de Dentes do Homem de Ferro com preço e maciez;
- Observe que temos superheróis de duas **Franquias** diferentes: Marvel e DC. Cada Franquia cobra um valor de comissão sobre os produtos: a Marvel cobra 35% e a DC 30%;
- Fraldas são recomendadas para crianças com **Faixa Etária** até 3 anos. Lancheiras e Escovas são recomendadas para crianças de 4 a 8 anos;
- Cada Faixa Etária também guarda o nome da categoria que pode ser Baby (0 a 3 anos) ou Kids (4 a 8 anos).

Dica: utilize o website [draw.io \(Blank Diagram\)](https://draw.io) para desenhar seu grafo por agora de forma mais livre ;)

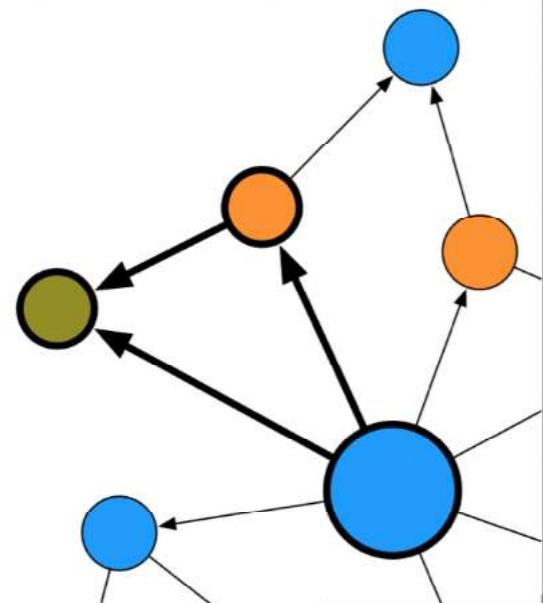


3.2. Bancos Orientados a Grafos

Um Banco Orientado a Grafos é uma ferramenta de armazenamento desenhada para tratar os relacionamentos entre os dados de forma tão importante como os próprios dados.

- Enquanto BDs Relacionais realizam relacionamentos em tempo de busca e geralmente por meio de JOINS, um **BD Orientado a Grafo já guarda as conexões juntamente com os dados**;
- Acessos aos Nós e Relacionamentos nestes BDs são operações eficientes e que permitem **examinar milhões de conexões por segundo**;
- Independente da quantidade de dados, estes BDs foram criados para, a partir de um ponto de início, coletar e agregar informações com a máxima eficiência e **evitando caminhar por conexões irrelevantes**;
- Dos vários SGBDs disponíveis para se trabalhar com esta filosofia, destaca-se o **Neo4j**;

Vamos conhecê-lo um pouco mais?



3.2. Bancos Orientados a Grafos

O Neo4j é um BD NoSQL Orientado a Grafos, feito em Java, open-source e líder da sua categoria.

Alguns dos seus principais recursos e características são:

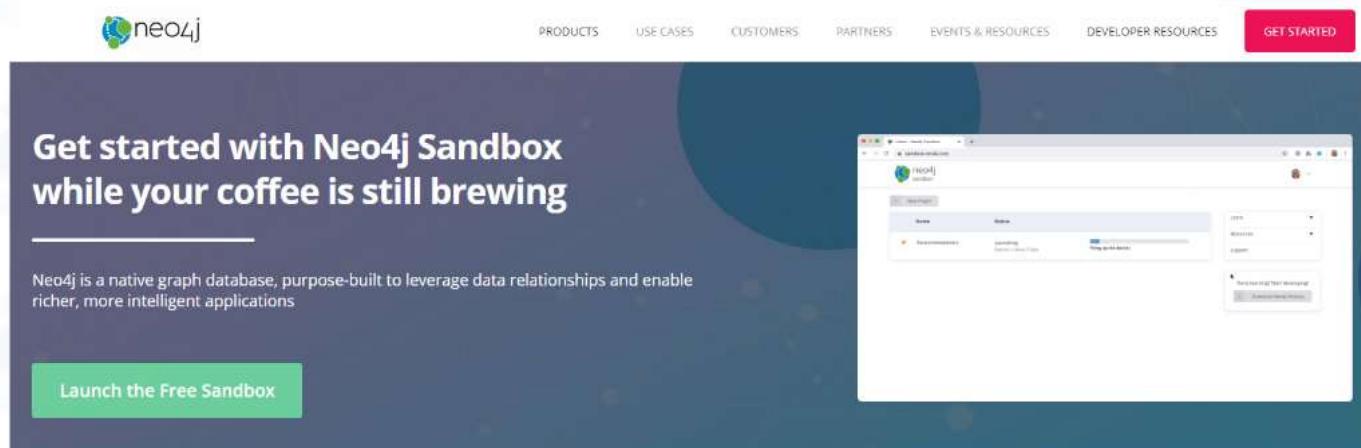
- Usa uma linguagem própria para busca de dados chamada Cypher, similar ao SQL mas otimizada para o contexto de grafos;
- Tempo de busca constante, mesmo em Grafos gigantescos e com alta escalabilidade;
- Altamente flexível, como qualquer BD NoSQL (Ex: Schemaless);
- Oferece Drivers de conexão facéis com as principais Linguagens de Programação;
- Ótima documentação disponível pelo link: <https://neo4j.com/docs/>



3.3. Primeiras Impressões com o Neo4j Sandbox

O Neo4j Sandbox é um recurso online, rápido e objetivo para começarmos a trabalhar com o BD Neo4j. Ele cria uma instância isolada para cada usuário para que possam criar e manipular um BD Orientado a Grafos.

Vamos visitar <https://neo4j.com/sandbox/> para criarmos um Sandbox gratuito.



- Será necessário realizar um cadastro rápido para que possa criar seu Sandbox privado;
- Após o cadastro, selecione o BD Movies para criarmos uma Instância do mesmo dentro do Sandbox.

3.3. Primeiras Impressões com o Neo4j Sandbox

Abra a Instância do BD Movies criada para você.

The screenshot shows the Neo4j Sandbox web interface. At the top, there's a logo for 'neo4j sandbox'. Below it, a 'New Project' button is visible. A table lists a single project:

Name	Status
Movies	Running Expires in about 3 days

To the right of the table is a green 'Open' button with a dropdown arrow. A red arrow points from the text above to this 'Open' button.

The screenshot shows the 'Movies Project Tutorial' section of the Neo4j Sandbox. It includes:

- A sidebar with navigation links like 'Home', 'Graph', 'Cypher', 'Logs', 'Metrics', 'Logs', 'Metrics', 'Graph', 'Cypher', 'Logs', 'Metrics', and 'Logout'.
- A title 'What is Cypher?' with a description: 'Cypher is a graph query language that is used to query the Neo4j Database. Just like you use SQL to query a MySQL database, you would use Cypher to query the Neo4j Database.'
- A code editor containing a Cypher query:

```
Match (m:Movie) where m.released > 2000 RETURN m limit 5
```

 with a hint: 'Hint: You can click on the query above to populate it in the editor.'
- An 'Expected Result' section stating: 'The above query will return all the movies that were released after the year 2000 limiting the result to 5 items.'
- A large graph visualization showing nodes (represented by orange circles) and relationships (represented by blue lines). The nodes include movie titles like 'The Godfather', 'The Shawshank Redemption', 'The Dark Knight', etc. The relationships are labeled with actions like 'ACTED_IN', 'DIRECTED', 'PRODUCED', 'WRITING', 'FOLLOWED_BY', and 'REVIEWED'.
- Two command-line sections at the bottom:
 - neo4j\$ MATCH (n) RETURN n
 - neo4j\$ MATCH (n:Movie) RETURN n LIMIT 25

Siga as dicas e explore a ideia do Neo4j de forma mais geral por meio desta Instância.

3.4. Criando Nós e Relacionamentos

A forma de se criar um Nó é muito parecida com a forma de se buscar um Nó. Mas ao invés de usar a palavra-chave MATCH, usamos a CREATE.

Ex: \$ CREATE ()

Porém, observe que o nó criado não possuirá até o momento nenhum Label e nenhuma Propriedade. Isso é chamado de Nó Anônimo.

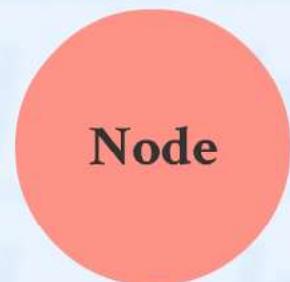
Ex: \$ MATCH (node)
RETURN node

Para se criar um Nó já com um Label, devemos passar dentro do CREATE o nome do Nó após : (dois pontos).

Ex: \$ CREATE (:Dog)

Lembrando que podemos criar nós com múltiplos labels.

Ex: \$ CREATE (:Dog:Animal)



3.4. Criando Nós e Relacionamentos

Além dos Nós possuírem Labels, eles também devem possuir as Propriedades a fim de descrever suas características/atributos.

Ex: \$ CREATE(:Dog:Animal{som:'AuAuu',comida:'racao'})

Observe as Propriedades dentro de {}.

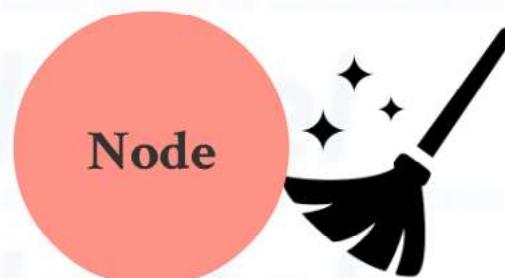
Como pode ser visto, criar Nós é um processo simples, mas se não os conectarmos de alguma forma, um Grafo não faz sentido!

Neste momento, vamos compreender como criar Nós mas que também possuam Relacionamentos.

Mas antes, vamos limpar nosso Database.

Ex: \$ MATCH (n)
DETACH DELETE n

Obs: Em breve veremos outras formas de exclusão.



3.4. Criando Nós e Relacionamentos

Já que falamos de Propriedades, seguem os tipos de dados suportados pelo Neo4j.

Type	Example
Boolean	true, false
Text	'AKA string'
Numbers	123, 56.70
Point	2D, 3D, Lat Lon, Lat Lon Height
Temporal	Date, Time, DateTime, LocalTime, LocalDateTime, Duration
Lists	['must', 'be', 'same', 'type']

3.4. Criando Nós e Relacionamentos

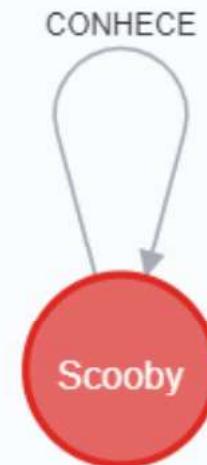
Ex de Relacionamento 1 (Sem Propriedade(s)):

```
CREATE(d:Dog{name:'Scooby'})-[:CONHECE]->(d)
```

No exemplo acima, além de criarmos um novo Nó, também estamos fazendo um relacionamento com ele mesmo.

Algumas observações importantes:

- Observe que o Relacionamento é do tipo CONHECE;
- Ele está dentro de -[]->, deve começar com : (igual os Nós) e a seta indica a direção do Relacionamento;
- A variável d é uma referência utilizada para deixar claro quem está sendo acessado;



Mas Relacionamentos também não podem ter Propriedades?

3.4. Criando Nós e Relacionamentos

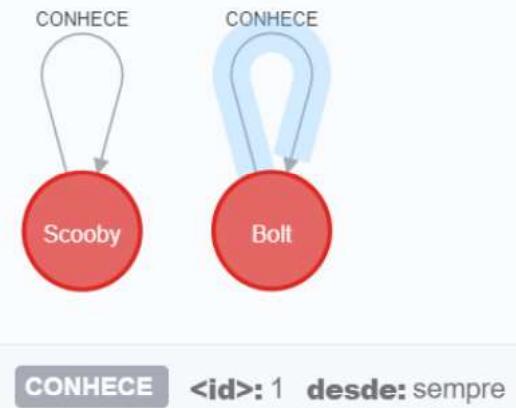
Ex de Relacionamento 2 (Com Propriedade(s)):

CREATE(d:Dog{name:'Bolt'})...

...-[:CONHECE{desde:'sempre'}]->(d)

Algumas observações importantes:

- Observe que agora ao dar um foco sobre o Relacionamento do novo Nô, a Propriedade também aparecerá;
- Veja que tanto para novos Nôs quanto Relacionamentos, o Neo4j já cria id's (<id>) automaticamente;
- Observe na aba 'Database Information' (canto superior esquerdo) que, conforme novos Nôs, Relacionamentos e Propriedades vão sendo criadas, eles vão sendo relacionados/registrados para facilitar buscas futuras.



Mas como relacionar Nôs com outros Nôs que não sejam eles mesmos?

3.4. Criando Nós e Relacionamentos

A fim de incrementar nosso exemplo, vamos criar mais dois novos Nós, mas do tipo Pessoa (Person):

Ex: \$ CREATE(:Person{name:'Penny'}),(:Person{name:'Salsicha'})

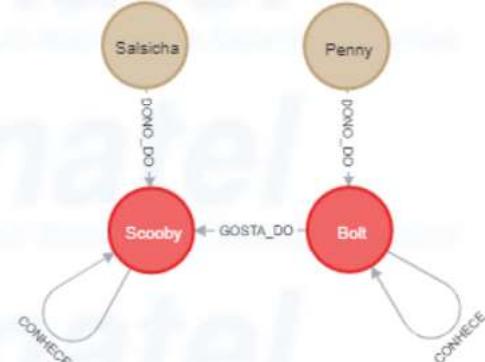
Estes Nós representam os proprietários dos dois Dogs que criamos. Mas para isso fazer sentido dentro do nosso Database, temos que Relacionar cada Dog com seu respectivo Dono.

Ex: \$ MATCH (p:Person{name: 'Salsicha'}),(d:Dog{name: 'Scooby'})
CREATE (p)-[:DONO_DO]->(d)

\$ MATCH (p:Person{name: 'Penny'}),(d:Dog{name: 'Bolt'})
CREATE (p)-[:DONO_DO]->(d)

O Bolt também é fã do Scooby ;):

\$ MATCH (d1:Dog{name: 'Bolt'}),(d2:Dog{name: 'Scooby'})
CREATE (d1)-[:GOSTA_DO{intensidade: 'muito'}]->(d2)



Para mais opções de criação, acesse:

<https://neo4j.com/docs/cypher-manual/current/clauses/create/>

3.4. Criando Nós e Relacionamentos

Exercício Proposto

Agora que já possui uma noção de como se cria Nós e Relacionamentos, crie um grafo em Neo4j para representar o que se pede.

O Spotify além de listar em seu aplicativo todas as músicas e discos de um determinado artista, também possui um serviço de recomendação muito legal que permite que o ouvinte conheça bandas que possuam alguma sinergia.

Esta funcionalidade passará por um upgrade! Será possível ver também o quão forte será este relacionamento entre as bandas. Crie um Grafo que represente esta funcionalidade envolvendo pelo menos 3 bandas.

Sugestão:

- Band: Metallica, 1981, Heavy Metal
- Album: Black Album, 1986, 12 faixas
- Music: Enter Sandman, 5:30
- possui relacionamento forte com**
- Band: Iron Maiden, 1975, Heavy Metal
- Album: Piece of Mind, 1983, 9 faixas
- Music: The Trooper, 4:12
- possui relacionamento medio com**
- Band: Bon Jovi, 1983, Rock
- Album: Crush, 2000, 13 faixas
- Music: Its My Life, 3:45



3.4. Criando Nós e Relacionamentos

Apesar do Sandbox ser uma opção mais objetiva para prática de Cypher, outra opção a ser utilizada é o Neo4j Desktop.

Faça download da aplicação pelo link: <https://neo4j.com/download/> e a instale em seu computador.

Algumas orientações importantes:

- Você deverá realizar um cadastro pessoal rápido para realizar o download;
- Após o cadastro, será gerada uma chave para você. Use-a para ativar seu Neo4j durante o processo de instalação (a sugestão é que não feche a janela do navegador onde aparece a chave até utilizá-la na instalação);
- Execute o instalador e também o Neo4j Desktop após a instalação com permissão de ADMINISTRADOR para não ter problemas;
- Após a instalação, caso a ferramenta sugira a atualização de algo, faça isso (como por exemplo, pode pedir para a atualizar o Neo4j Browser);
- Após instalação, também é sugerido que sua máquina seja reiniciada.

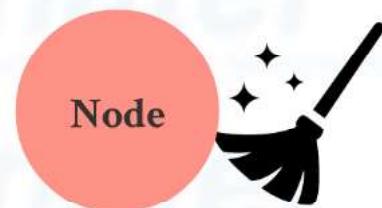


3.5. Excluindo e Atualizando elementos nos Grafos

Existem diversas formas de se realizar exclusão de elementos nos Grafos. Já vimos anteriormente como limpar todo Database. Agora seguem outros exemplos muito úteis mas de caráter mais específicos.

Deletando um Nó específico sem Relacionamentos;

Ex: \$ MATCH (d:Dog{name: 'Pluto'})
DELETE d



Deletando um Nó que possua Relacionamentos;

Ex: \$ MATCH (p:Person{name: 'Mickey'})
DETACH DELETE p

Deletando um Relacionamento;

Ex: \$ MATCH (p:Person{name: 'Mickey'})-[r:DONO_DO]->()
DELETE r

3.5. Excluindo e Atualizando elementos nos Grafos

Quando falamos de Atualizações em Grafos, estamos falando da adição/atualização/remoção de Labels e Propriedades em Nós e Relacionamentos.

Adicionando uma nova Propriedade em um Nó específico;

Ex: \$ MATCH (d:Dog{name: 'Bolt'})
SET d.company = 'Walt Disney'

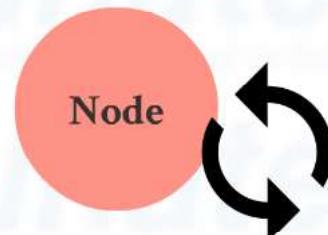
Adicionando um novo Label em um Nó específico;

Ex: \$ MATCH (d:Dog{name: 'Bolt'})
SET d:Animal

Adicionando uma nova Propriedade em um Relacionamento;

Ex: \$ MATCH (p:Person)-[dd:DONO_DO]->(:Dog)
WHERE p.name = 'Penny'
SET dd.desde = '2008'

Obs: quando um campo já existir mas precisar ter seu valor alterado, basta sobrescrevê-lo por meio do uso do SET.



3.5. Excluindo e Atualizando elementos nos Grafos

Removendo uma Propriedade em um Nó específico;

Ex: \$ MATCH (d:Dog{name: 'Bolt'})
 REMOVE d.company

Removendo um Label em um Nó específico;

Ex: \$ MATCH (d:Dog{name: 'Bolt'})
 REMOVE d:Animal

Removendo uma Propriedade de um Relacionamento específico;

Ex: \$ MATCH (p:Person)-[dd:DONO_DO]->(:Dog)
 WHERE p.name = 'Penny'
 REMOVE dd.desde

Para mais opções de atualização e exclusão, acesse:

<https://neo4j.com/docs/cypher-manual/current/clauses/delete/>

<https://neo4j.com/docs/cypher-manual/current/clauses/set/>

<https://neo4j.com/docs/cypher-manual/current/clauses/remove/>

3.6. Fundamentos de Buscas: Nós e Relacionamentos

Como já pudemos ver, para se buscar Nós no Neo4j utilizamos da cláusula MATCH.

Ex: \$ MATCH (n)
RETURN n

Neste caso buscaremos tudo que temos no nosso Database. Podemos usar a cláusula LIMIT após o RETURN para limitar o número de elementos que queremos retornar.

Ex: \$ MATCH (p:Person)
RETURN p

Neste caso já observe que estamos restringindo a busca para apena Nós com Labels do tipo Person. Poderíamos usar Labels encadeadas para buscar nós ainda mais específicos.

Ex: \$ MATCH (p:Person)
RETURN p.name

Veja que este exemplo é muito parecido com o de cima, mas com a diferença que não buscamos o Nó inteiro, mas apenas uma propriedade desejada do mesmo.

Por convenção, as cláusulas do Neo4j deverão ser escritas todas em MAIÚSCULAS.

3.6. Fundamentos de Buscas: Nós e Relacionamentos

Quando realizamos buscas de Nós relacionados com outros Nós, podemos fazer buscas mais gerais ou específicas.

Ex: \$ MATCH (c:Company)--(s:Serie)
RETURN c,s

O símbolo -- significa "relacionado a", sem levar em consideração o tipo ou a direção do(s) Relacionamento(s).

Ex: \$ MATCH (:Person{nome:'Mickey'})-[r]->(:Dog)
RETURN type(r)

A seta indica a direção de uma pergunta. Aqui estamos perguntando: "Qual o tipo do Relacionamento que Mickey tem com um Dog?".

Ex: \$ MATCH(:Serie{nome:'Scooby Doo'})<-[FAZ_PARTE]-(p:Person)
RETURN p

- Observe que o lado da seta não faz diferença, desde que o Tipo do Relacionamento faça sentido. Neste caso, estamos perguntando: "Quais são pessoas que fazem parte da Serie Scooby Doo?"
- IMPORTANTE: Observe também o retorno! Nesta pergunta poderíamos retornar tanto o nome da Serie quanto as pessoas, porém, retornamos apenas as pessoas que fazem parte da mesma (RETURN p).

3.6. Fundamentos de Buscas: Nós e Relacionamentos

Podemos realizar buscas ainda mais precisas levando em consideração múltiplos relacionamentos e/ou valores de propriedades com a cláusula WHERE.

Ex: \$ MATCH (p:Person)-[:DONO_DO]->(d:Dog)
 -[:FAZ_PARTE]->(:Serie{nome:'Mickey & Donald'})
 RETURN d

Observe os múltiplos relacionamentos restringindo a busca "Busque o Nó do Dog que faz parte da Série Mickey & Donald e que possui uma pessoa que é dono dele"

Ex: \$ MATCH (p:Person)
 WHERE p.sexo='F' AND p.nome STARTS WITH 'D'
 RETURN p

Observe o uso do WHERE para se criar filtros mais precisos se baseando nas Propriedades.

Ex: \$ MATCH (d:Dog)-[fp:FAZ_PARTE]->(s:Serie)
 WHERE fp.protagonista = true
 RETURN d

Observe a Propriedade de um Relacionamento auxiliando no filtro.

Para mais opções de busca, acesse:

<https://neo4j.com/docs/cypher-manual/current/clauses/match/>

3.6. Fundamentos de Buscas: Nós e Relacionamentos

Exercício Proposto

Agora que já possui uma noção de como se busca Nós e Relacionamentos, utilizando do nosso Database de exemplo crie scripts em Cypher que respondam às seguintes perguntas:

1. Busque qualquer tipo de Relacionamento que um Dog possa ter com uma Pessoa, ou vice-versa;
2. Busque apenas o nome das mulheres que fazem parte da série 'Scooby Doo';
3. Busque apenas os Nós da pessoas que atuam em séries da Disney e não são protagonistas;
4. Adicione o campo fundador='Walter Elias Disney' à Empresa que possua a Pixar como subsidiária;
DICA: use do operador IN no WHERE.



**FIM
DO
CAPÍTULO 3**



EXERCÍCIOS