

Bancos de dados de Grafos e Neo4j

IGOR ROZANI



Igor Rozani



IgorRozani



IgorRozani



@IgorRozani



IgorRozani

Tipo de bancos

SQL



ORACLE®



PostgreSQL



NOSQL




cassandra



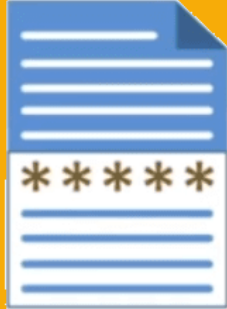
Tipos de NoSQL

Key Value



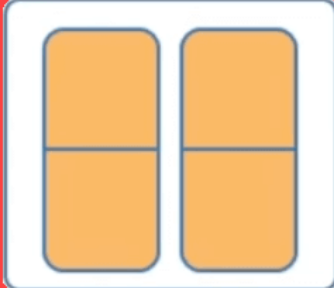
Example:
Riak, Tokyo Cabinet, Redis server, Memcached, Scalaris

Document-Based



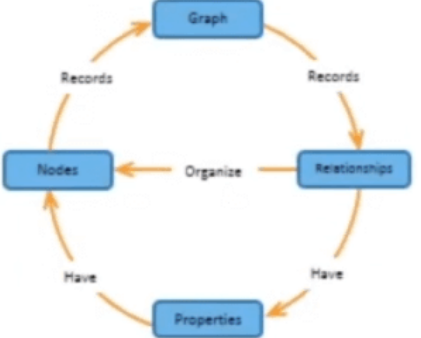
Example:
MongoDB, CouchDB, OrientDB, RavenDB

Column-Based



Example:
BigTable, Cassandra, Hbase, Hypertable

Graph-Based

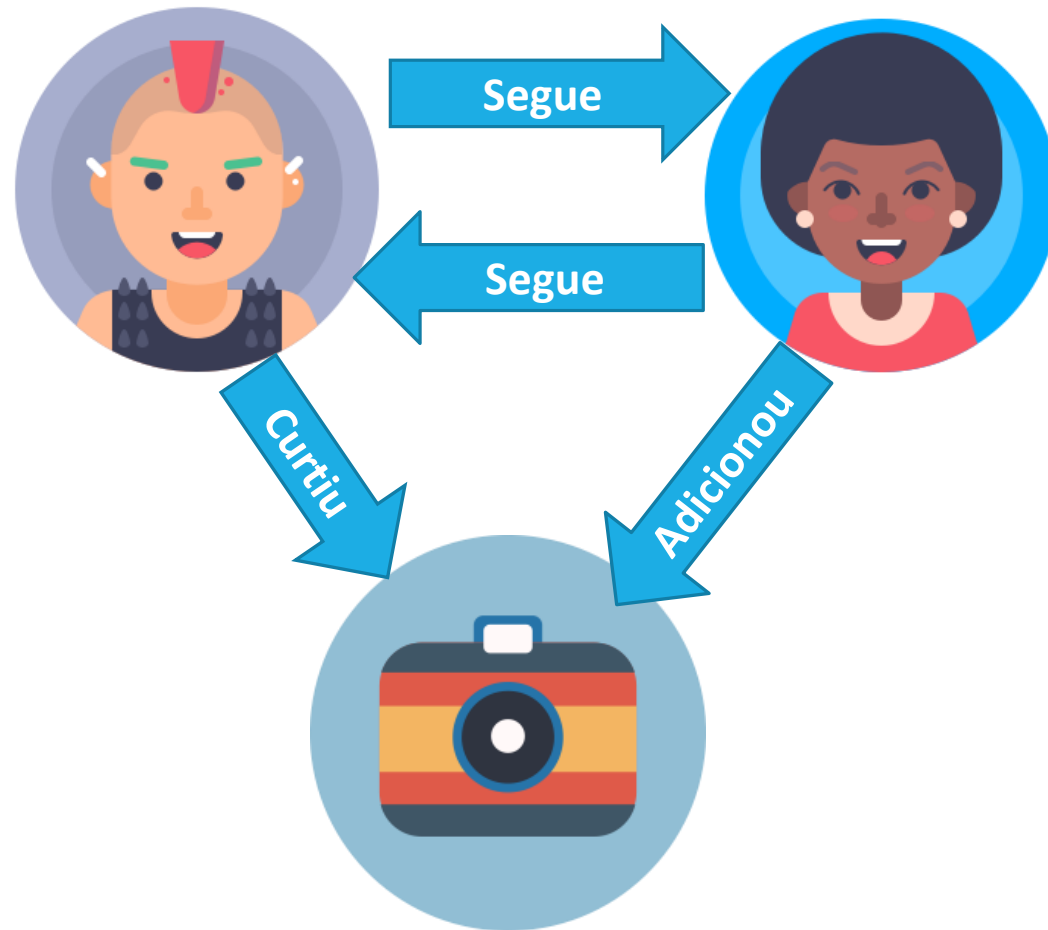


Example:
Neo4J, InfoGrid, Infinite Graph, Flock DB

Bancos de grafos

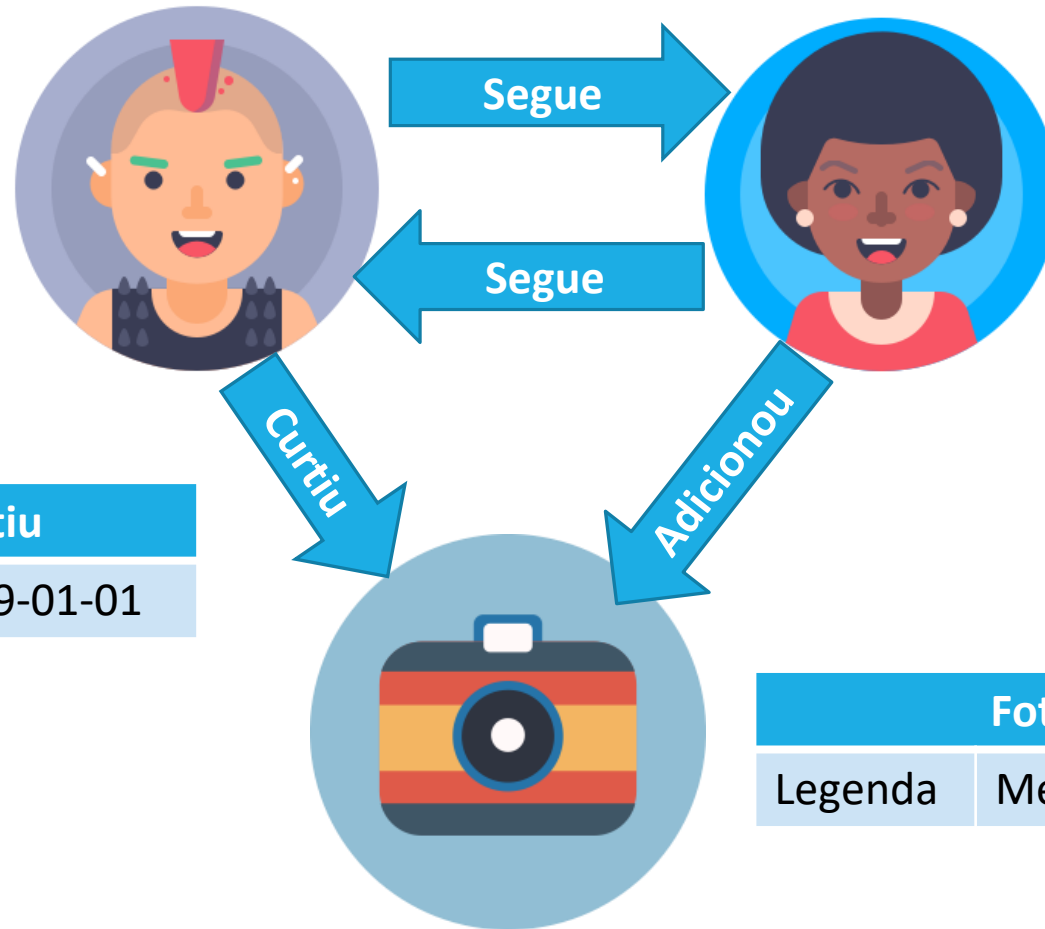


Estrutura de grafos



Estrutura de grafos

Pessoa	
Nome	Gabriel
País	Canadá
Estado civil	Namorando



Curtiu	
Data	2019-01-01

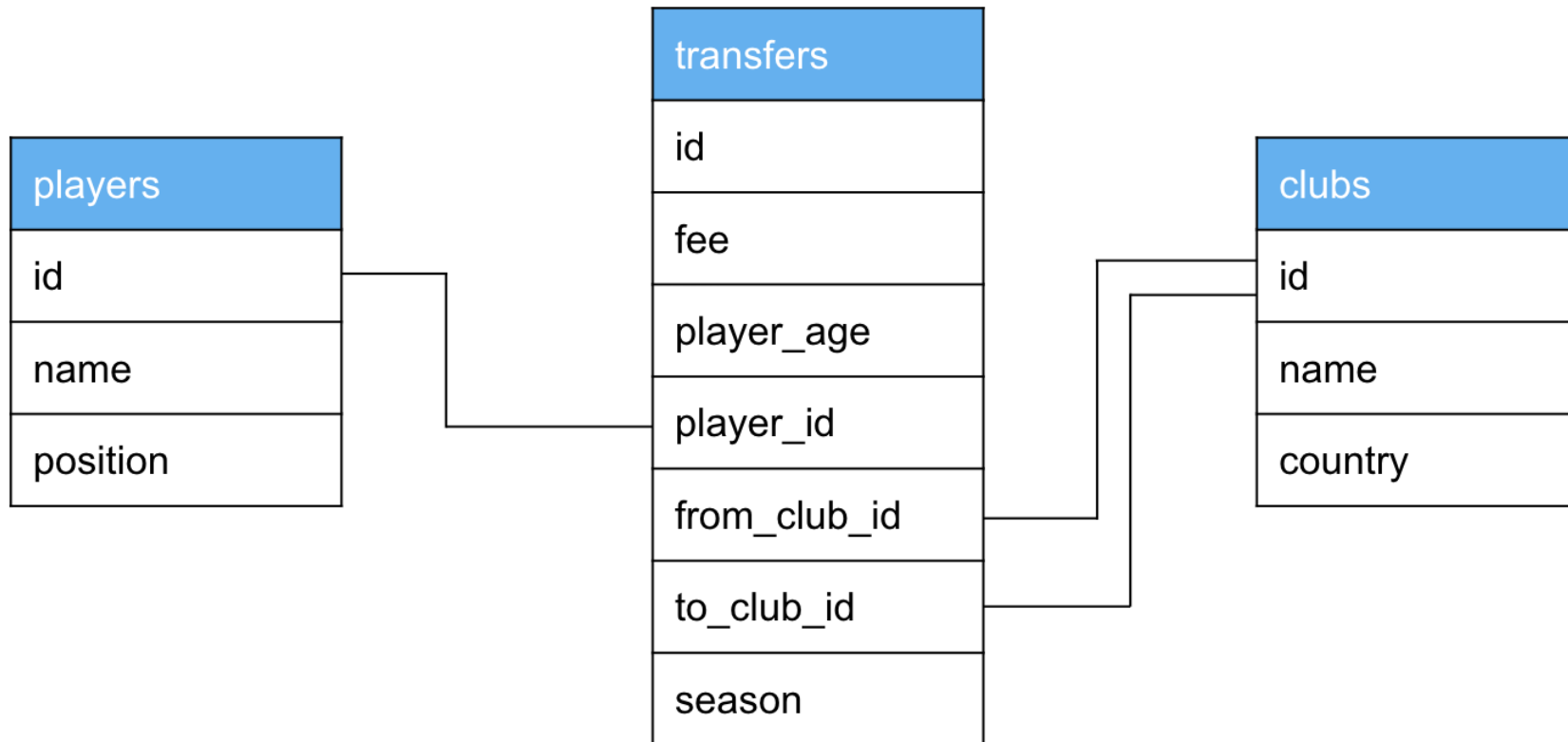
Pessoa	
Nome	Carol
País	Brasil

Foto	
Legenda	Meu cachorro

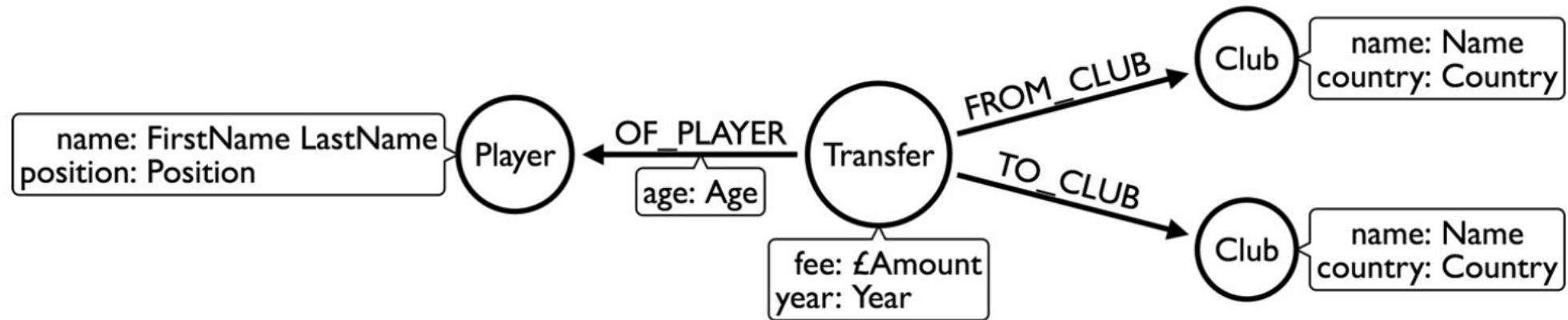
Estrutura de grafos

Relacional	Grafo
Linha	Nós
Joins	Relacionamentos
Nome da tabela	Labels
Coluna	Propriedades

Estrutura de grafos



Estrutura de grafos



Bancos de grafos



SQL



Cypher

Criação de um nó

Pessoa	
Nome	Gabriel
País	Canadá
Estado civil	Namorando



```
CREATE (:Pessoa {nome: 'Gabriel', pais: 'Canadá', estadoCivil: 'namorando'})
```

Criação de um nó

CREATE (:Pessoa {nome: 'Gabriel', pais: 'Canadá', estadoCivil: 'namorando'})

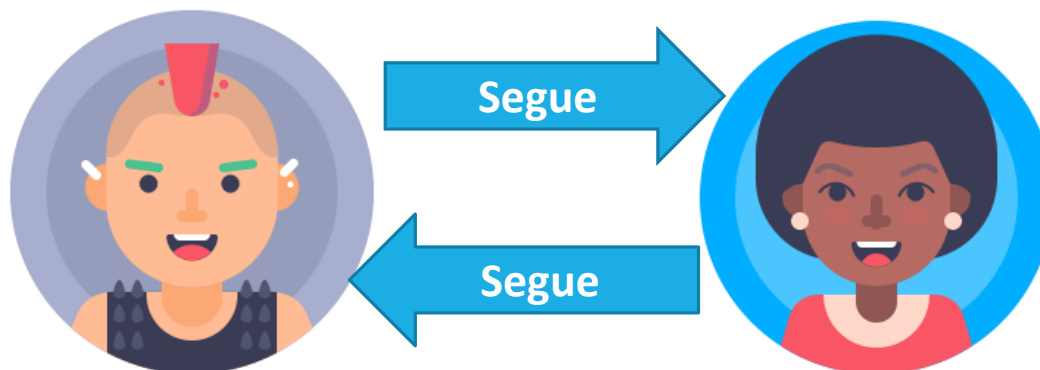
Comando
criação

Label

Propriedades

Criação de um relacionamento

Pessoa	
Nome	Gabriel
País	Canadá
Estado civil	Namorando



Pessoa	
Nome	Carol
País	Brasil

```
MATCH (p1:Pessoa{nome: 'Gabriel'}), (p2:Pessoa{nome: 'Carol'})  
CREATE (p1)-[:SEGUE]->(p2)
```

Criação de um relacionamento

Comando busca

Filtro do primeiro nó

Filtro do segundo nó

MATCH (p1:Pessoa{nome: 'Gabriel'}), (p2:Pessoa{nome: 'Carol'})

Indicação de relacionamento

Direção do relacionamento

CREATE (p1)-[:SEGUE]->(p2)

Comando criação

Nó 1

Dados do relacionamento

Nó 2

Atualizar um registro



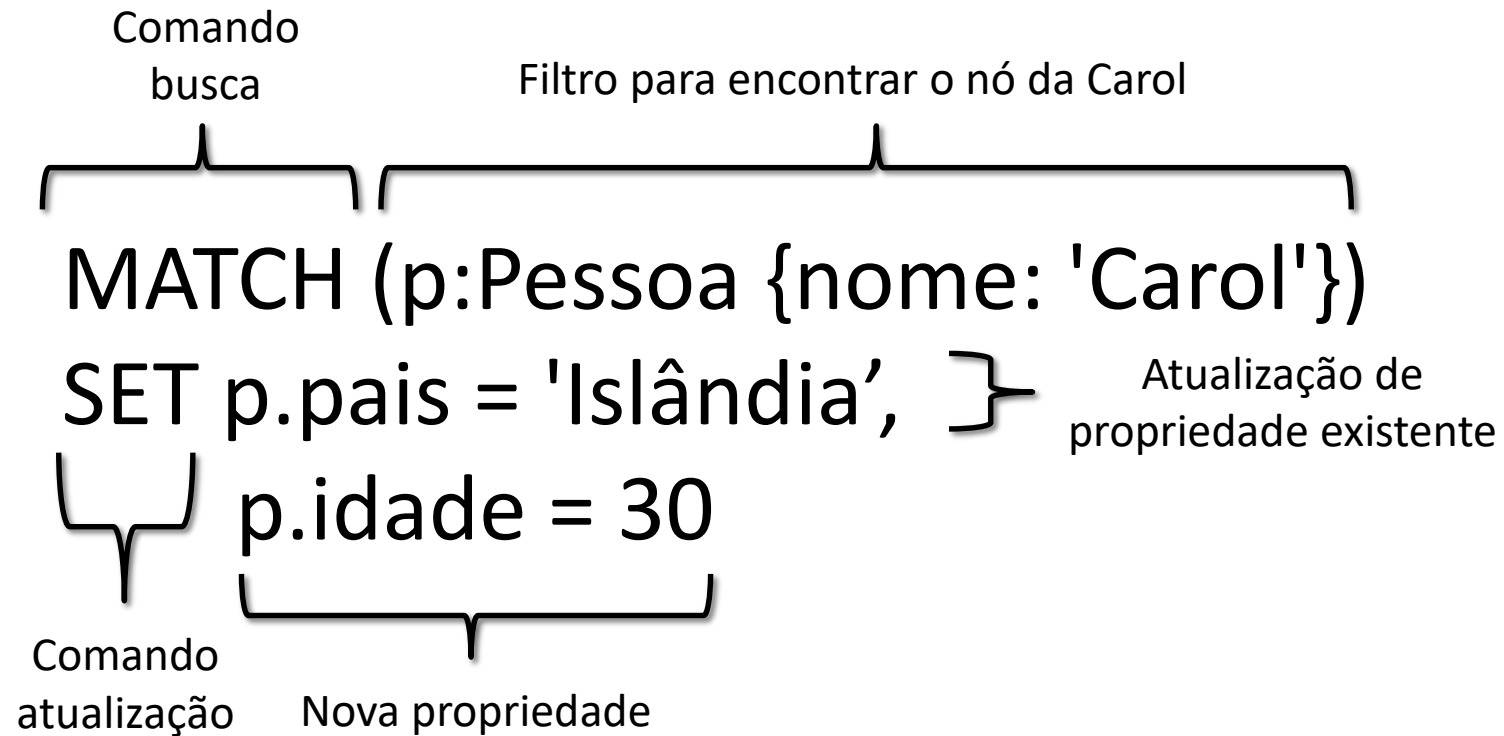
Pessoa	
Nome	Carol
País	Brasil



Pessoa	
Nome	Carol
País	Islândia
Idade	30

```
MATCH (p:Pessoa {nome: 'Carol'})  
SET p.idade = 30,  
    p.pais = 'Islândia'
```


Atualizar um registro



Excluir um nó

Comando
busca

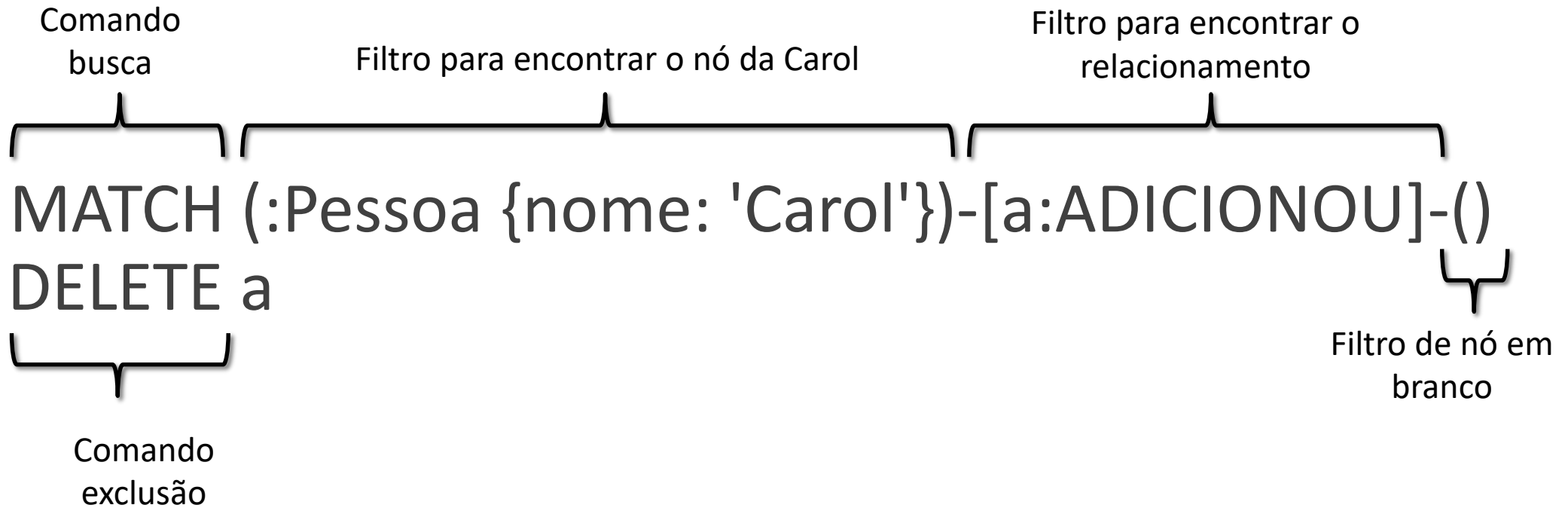
Filtro para encontrar o nó da Carol

```
MATCH (p:Pessoa {nome: 'Carol'})  
DELETE p
```

Comando
exclusão

The diagram illustrates the components of a Cypher query. A horizontal line is drawn above the query text. A bracket above the line spans from the start of 'MATCH' to the end of the filter '{nome: 'Carol'}', with the label 'Comando busca' above it. A second bracket above the line spans from the start of the filter to the end of the closing parenthesis of the filter, with the label 'Filtro para encontrar o nó da Carol' above it. A third bracket below the line spans from the start of 'DELETE' to the end of 'p', with the label 'Comando exclusão' below it.

Excluir um relacionamento



Excluir um nó e seus relacionamentos

Comando
busca

Filtro para encontrar o nó da Carol

```
MATCH (p:Pessoa {nome: 'Carol'})  
DETACH DELETE p
```

Comando para exclusão de nó e
seus relacionamentos

The diagram illustrates the components of the Cypher query. A bracket under 'MATCH (p:Pessoa {nome: 'Carol'})' is labeled 'Comando busca'. A larger bracket under 'MATCH (p:Pessoa {nome: 'Carol'})' and 'DETACH DELETE p' is labeled 'Filtro para encontrar o nó da Carol'. A bracket under 'DETACH DELETE p' is labeled 'Comando para exclusão de nó e seus relacionamentos'.

Consultar dados

Pessoa	
Nome	Gabriel
País	Canadá
Estado civil	Namorando



```
MATCH (p:Pessoa{nome:'Gabriel'})  
RETURN p
```

Consultar dados

Consulta	Retorno
<pre>MATCH (p:Pessoa {nome:'Gabriel'}) RETURN p</pre>	<pre>{ "nome": "Gabriel", "estadoCivil": "Namorando", "pais": "Canadá" }</pre>
<pre>MATCH (p:Pessoa {nome:'Gabriel'}) RETURN p.nome</pre>	<pre>"Gabriel"</pre>

Consultar dados



Gabriel



Carol

```
MATCH (:Pessoa{nome:"Gabriel"})-[:CURTIU]->(f:Foto)<-[:ADICIONOU]-(:Pessoa{nome:"Carol"})
```

```
RETURN count(f)
```

Consultar dados

Filtro para encontrar as fotos
que o Gabriel curtiu

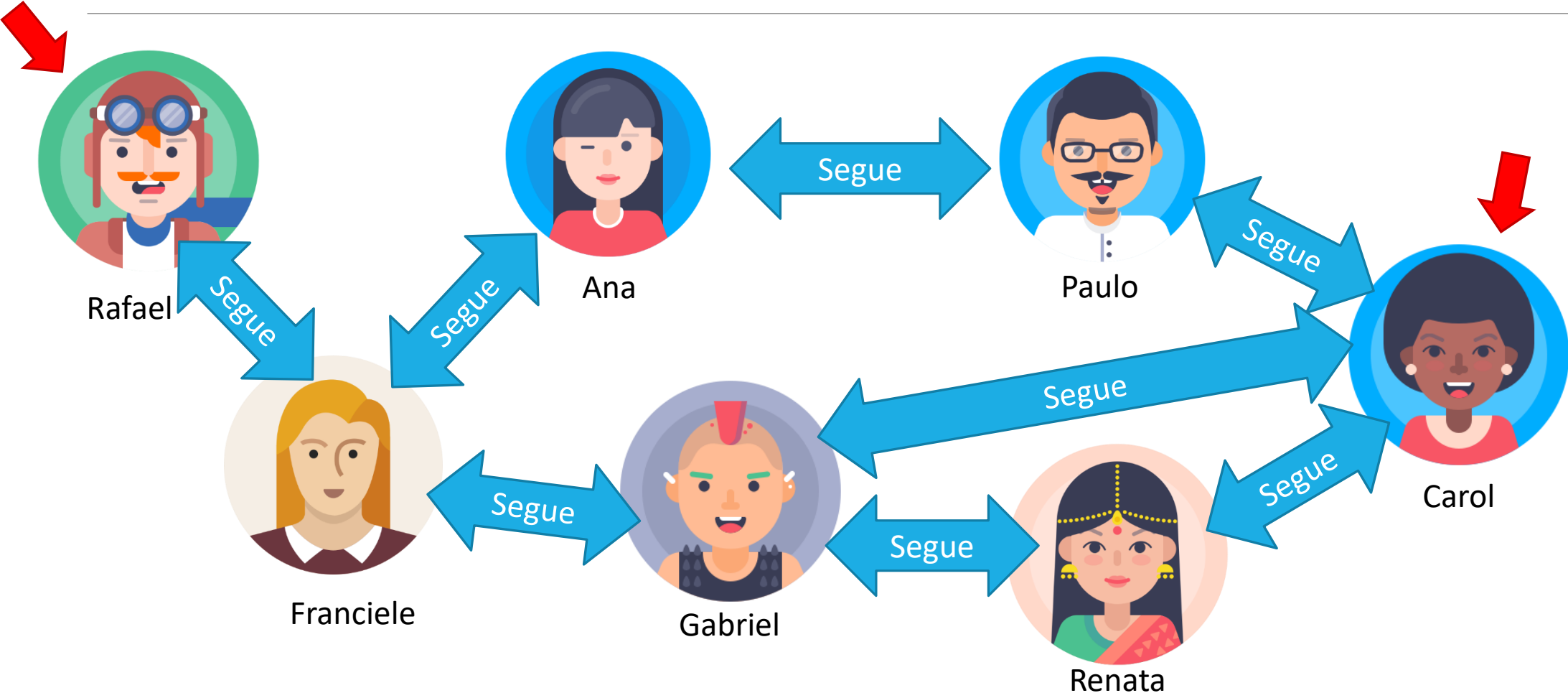
Filtro para encontrar as fotos
que a Carol adicionou

MATCH (:Pessoa{nome:"Gabriel"})-[:CURTIU]->(f:Foto)<-[:ADICIONOU]-(:Pessoa{nome:"Carol"})

RETURN count(f)

Retorno da quantidade
de fotos curtidas

Consultar dados



Consultar dados

MATCH

(p1:Pessoa{nome:'Rafael'}),

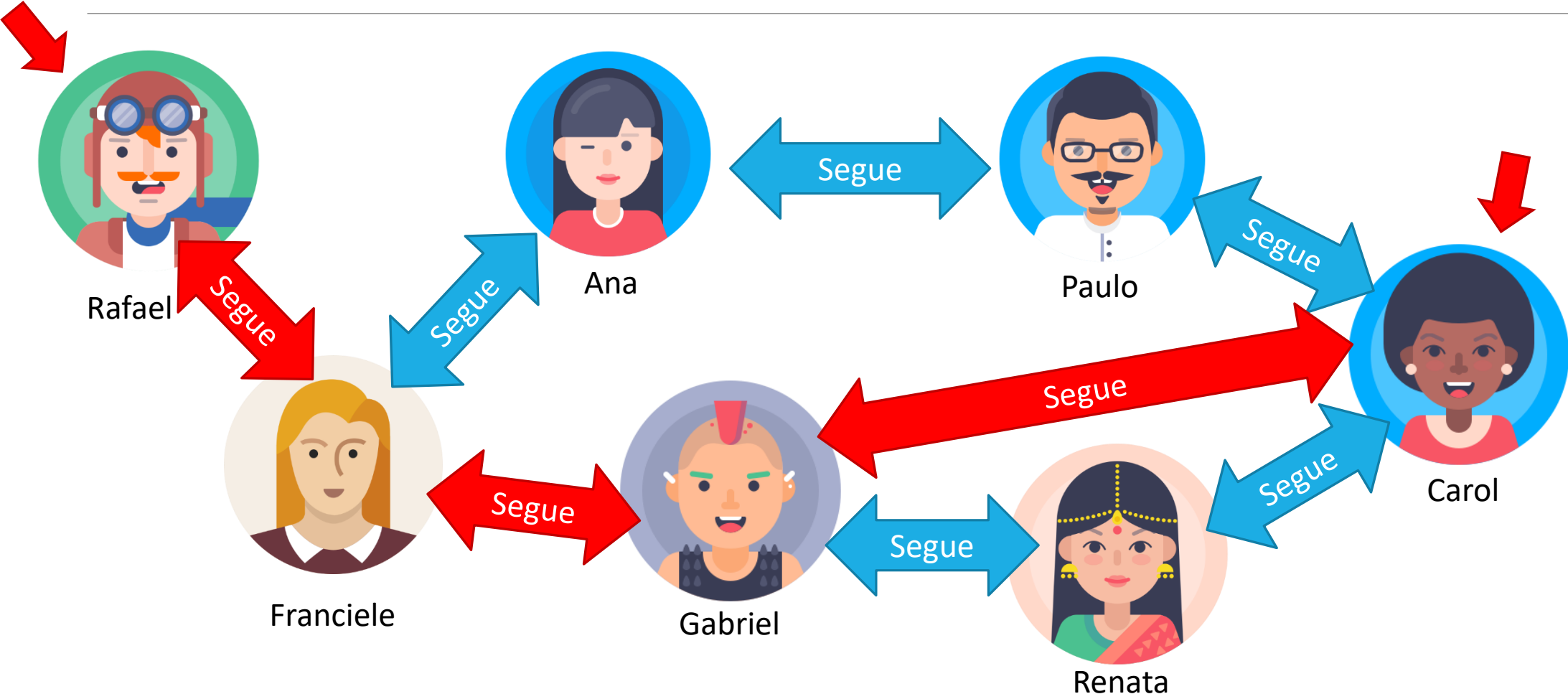
(p2:Pessoa{nome:'Carol'}),

sp = shortestPath((p1)-[:SEGUE*]-(p2))

RETURN EXTRACT(n in NODES(sp) | n.nome) AS Conexoes

Resultado: ["Rafael", "Franciele", "Gabriel", "Carol"]

Consultar dados



Performance

Their experiment used a basic social network to find friends-of-friends connections to a depth of five degrees. Their dataset included 1,000,000 people each with approximately 50 friends. The results of their experiment are listed in the table below:

Depth	RDBMS execution time(s)	Neo4j execution time(s)	Records returned
2	0.016	0.01	~2,500
3	30.267	0.168	~110,000
4	1543.505	1.359	~600,000
5	Unfinished	2.132	~800,000

FIGURE 2.2:

A performance experiment run between relational databases (RDBMS) and Neo4j shows that graph databases handle data relationships extremely efficiently.

Fonte: Livro Graph Databases for Beginners

Quando usar?

Redes sociais

Sistema de recomendação

Sistema de logística

Detecção de fraude

Controle de acesso



“Vou usar banco de grafos pra tudo”

Empresas que utilizam

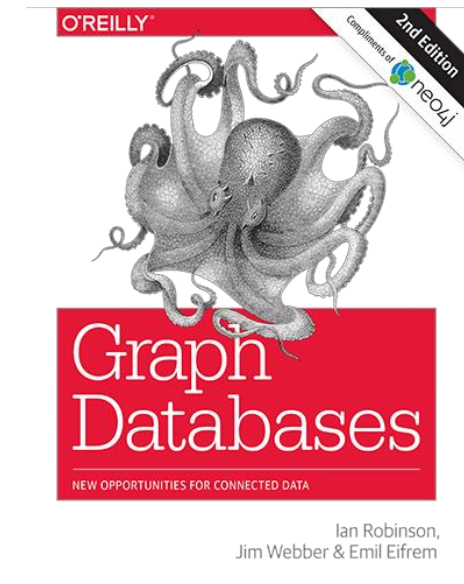
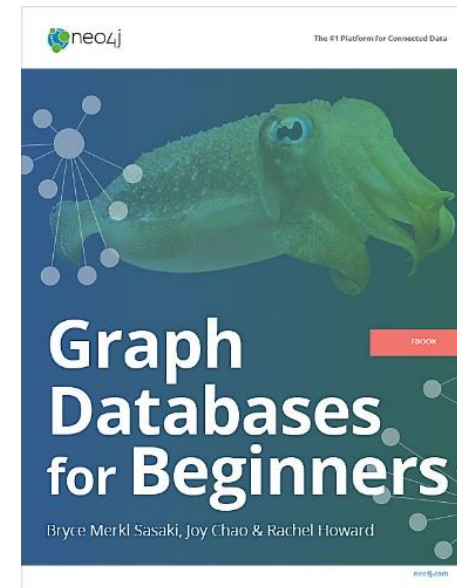


Como começar?

Documentação

Neo4j GraphGists

Online meetups



Certificação

Preço: grátis

Questões: 80

Tempo de prova: 1 hora

Nota mínima: 80%

Formato: perguntas de múltiplas escolhas

Limite de tentativas?: infinito enquanto não passar



Obrigado!

Dúvidas?

