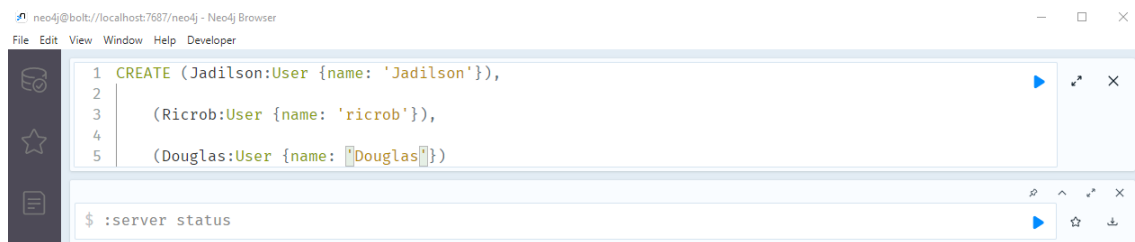


Pedro Fortunato Da Silva Neto;

Tópicos especiais em CD;

PRÁTICA 1 – SIMULADOR DE REDES SOCIAIS COM NEO4J

Neste momentos estamos criando usuários:



```
1 CREATE (Jadilson:User {name: 'Jadilson'}),
2
3   (Ricrob:User {name: 'ricrob'}),
4
5   (Douglas:User {name: 'Douglas'})
```

\$:server status

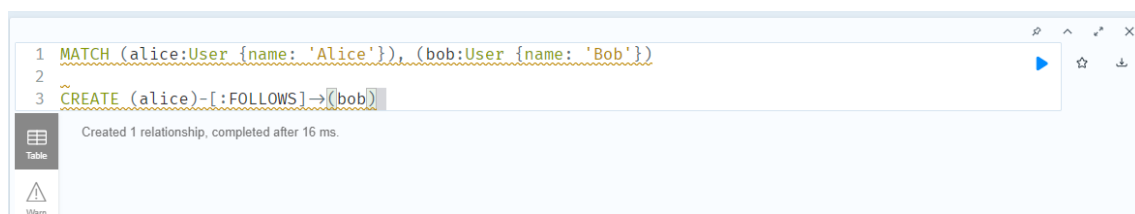
O retorno do comando mostra que os usuários(nós) foram criados com sucesso e estão prontos para os seguintes passos.



```
neo4j$ CREATE (Jadilson:User {name: 'Jadilson'}), (Ricrob:User {name: 'ricrob'}), (Douglas:User {na...
```

Added 3 labels, created 3 nodes, set 3 properties, completed after 70 ms.

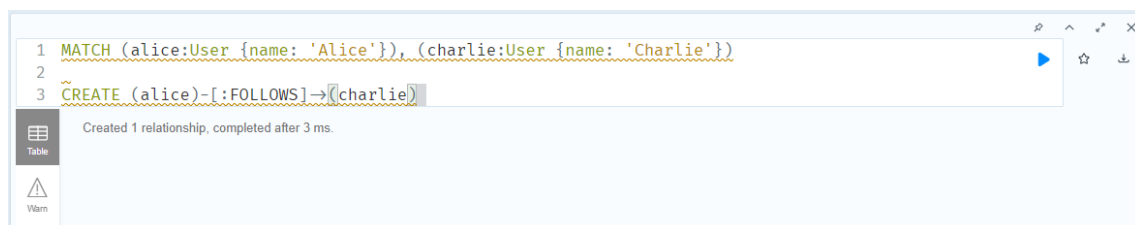
Agora iremos criar um relacionamento entre os nós “Alice” e “bob”



```
1 MATCH (alice:User {name: 'Alice'}), (bob:User {name: 'Bob'})
2
3 CREATE (alice)-[:FOLLOWS]->((bob))
```

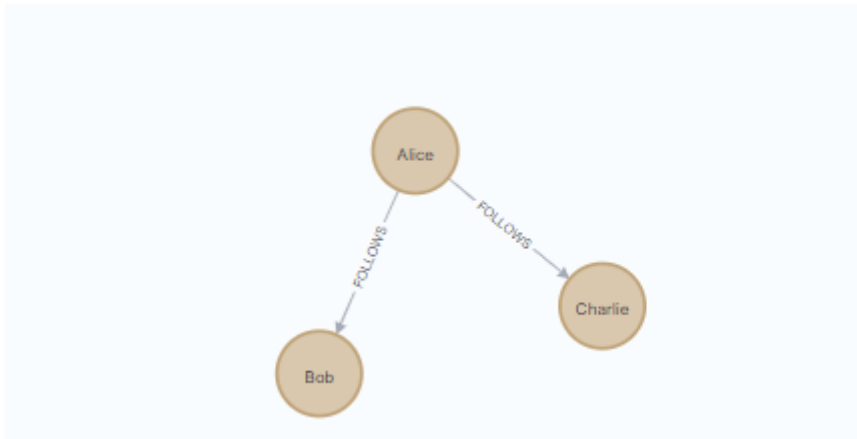
Created 1 relationship, completed after 16 ms.

Iremos criar mais conexões, iremos criar entre “Alice” e “Charlie”



```
1 MATCH (alice:User {name: 'Alice'}), (charlie:User {name: 'Charlie'})
2
3 CREATE (alice)-[:FOLLOWS]->((charlie))
```

Created 1 relationship, completed after 3 ms.



Agora Iremos relacionar a postagem ao usuário que fez a postagem

```
1 MATCH (alice:User {name: 'Alice'}), (Post1:Post {name:'Post'})
2 CREATE (alice)-[:FOLLOWS]->(Post1)
```

Como nossa prática fala sobre redes sociais, vamos criar alguns textos que simulam “posts” nas redes sociais

CONSIDERAÇÕES FINAIS: Nos criamos um simulador de rede social, onde houveram criação de posts, relacionamento entre pessoas da nossa rede social e criação de usuários. Um caso rotineiro que poderia facilmente ser aplicado a redes sociais já consolidadas no mercado.

```
1 CREATE (post1:Post {content: 'Hello, World!', timestamp: timestamp()}),
2
3 (post2:Post {content: 'Neo4j is awesome!', timestamp: timestamp()})
```

Created 1 relationship, completed after 3 ms.

Prática 02 – Sistema de gestão de escolas e faculdades

Agora simulando um sistema de faculdade, vamos começar criando os nós: alunos, professores, matérias e etc.

```
neo4j$ CREATE (alice:Aluno {nome: 'Alice'}), (bob:Aluno {nome: 'Bob'})...
```

Added 8 labels, created 8 nodes, set 8 properties, completed after 42 ms.

Agora criaremos o relacionamento entre alunos e as matérias que eles participam

```
neo4j$ MATCH (alice:Aluno {nome: 'Alice'}), (bob:Aluno {nome: 'Bob'}),...
```

Created 2 relationships, completed after 21 ms.

The graph visualization shows four nodes: Bob (blue circle), Alice (blue circle), História (yellow circle), and Matemática (yellow circle). There are two directed relationships labeled 'MATRICULADO_EM'. One relationship points from Bob to História, and the other points from Alice to Matemática.

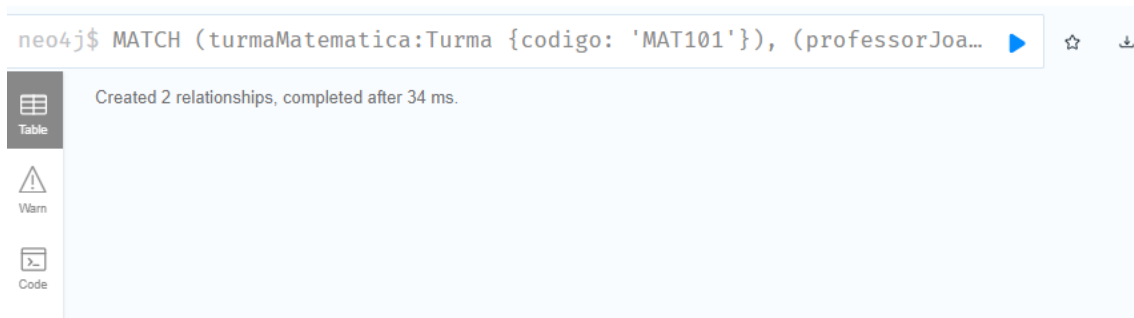
Agora iremos relacionar o curso com as turmas criadas

```
neo4j$ MATCH (cursoMatematica:Curso {nome: 'Matemática'}), (turmaMatem...
```

Created 2 relationships, completed after 12 ms.

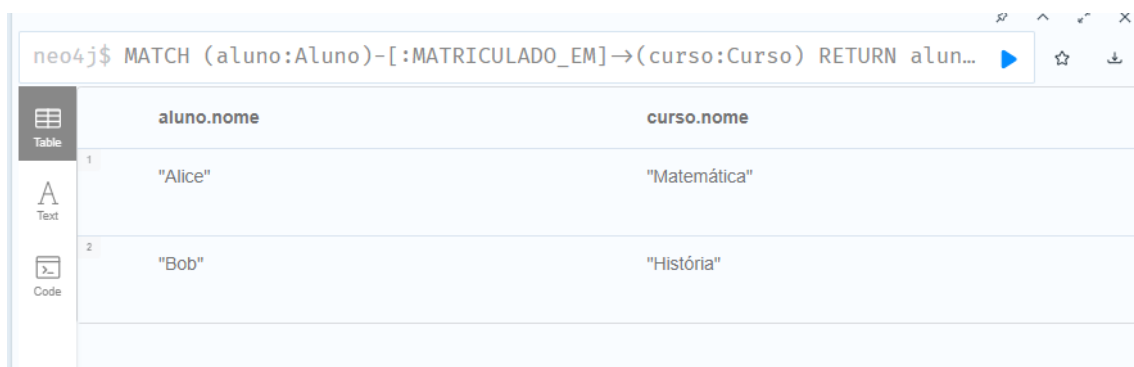
The graph visualization shows two nodes: cursoMatematica (blue circle) and turmaMatemática (yellow circle). There are two directed relationships connecting them, though the relationship labels are not visible in the image.

Agora iremos relacionar a turma com professores

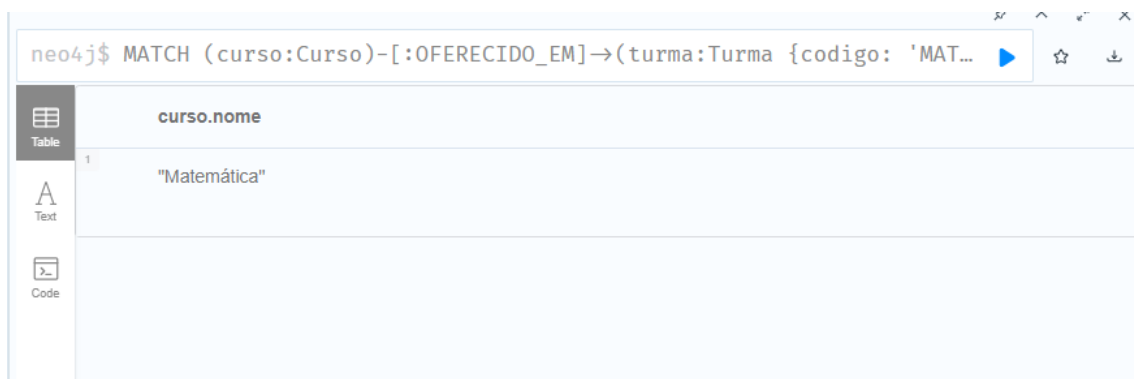


Como fazer consultas?: Vamos fazer consultas chamadas de Cypher para exploração dos dados

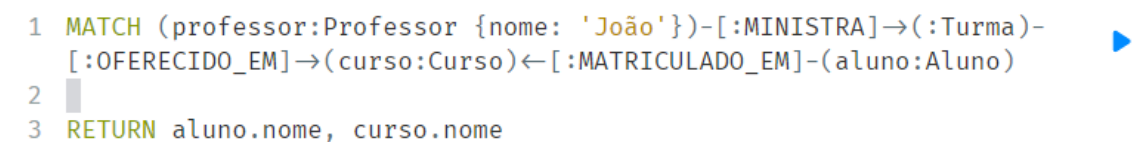
Listar todos os alunos matriculados em cursos:



Encontrar todos os cursos oferecidos em uma turma específica:



Descobrir todos os alunos matriculados em cursos ministrados por um professor específico:



Encontrar todos os professores que ministram cursos de um determinado departamento (por exemplo, Matemática):

```

1 MATCH (professor:Professor)-[:MINISTRA]→(:Turma)-
  [:OFERECIDO_EM]→(curso:Curso)
2
3 WHERE curso.nome CONTAINS 'Matemática'
4
5 RETURN DISTINCT professor.nome

```

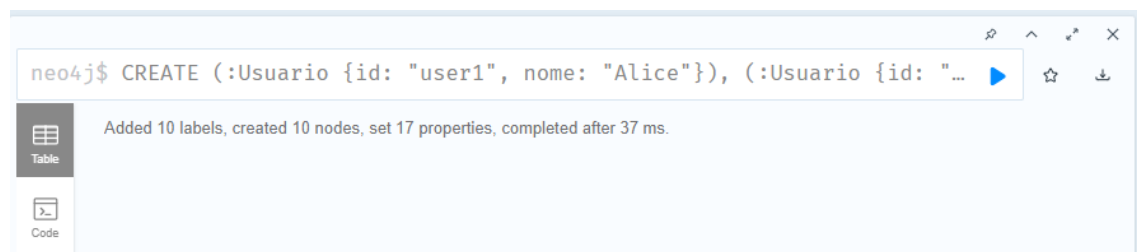
Prática 03 - Sistema de Detecção de Fraudes em Cartões de Crédito

Simulando que estamos criando um sistema de detecção de fraudes em cartões de crédito, os requisitos do cliente foram especificados no blackboard, vamos começar criando alguns nós de usuário, transação e endereço IP

```

1 CREATE
2
3 (:Usuario {id: "user1", nome: "Alice"}),
4
5 (:Usuario {id: "user2", nome: "Bob"}),
6
7 (:Usuario {id: "user3", nome: "Charlie"}),
8
9 (:EnderecoIP {endereco: "192.168.0.1"}),
10
11 (:EnderecoIP {endereco: "192.168.0.2"}),
12

```



No resultado final foram gerados os valores que especificamos, e adicionado em labels e nós

Agora vamos criar alguns relacionamentos entre usuários, transações e endereço IP

```

1 MATCH (u:Usuario), (t:Transacao), (ip:EnderecoIP)
2 WHERE u.id = "user1" AND t.id = "tx1" AND ip.endereco =
  "192.168.0.1"
3 CREATE (u)-[:REALIZOU]→(t), (u)-[:USOU_IP]→(ip)

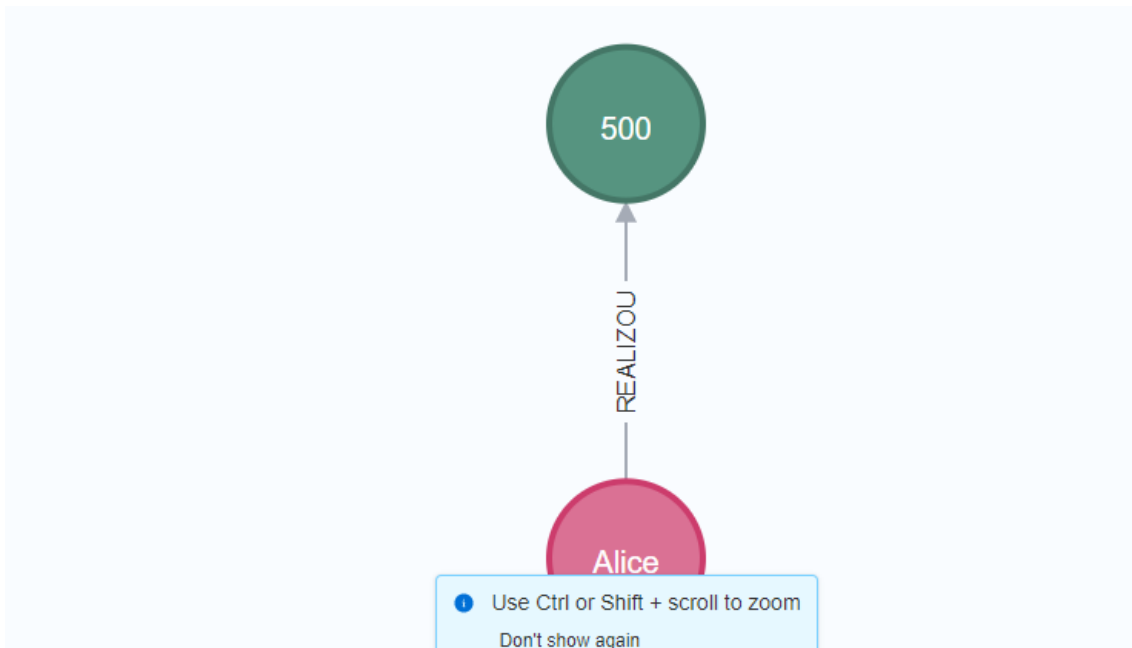
```

neo4j\$ MATCH (u:Usuario), (t:Transacao), (ip:EnderecoIP) WHERE u.id = ...

Created 2 relationships, completed after 30 ms.

Table

Warn



```
1 MATCH (u:Usuario), (t:Transacao), (ip:EnderecoIP)
2
3 WHERE u.id = "user2" AND t.id = "tx2" AND ip.endereco =
4 "192.168.0.2"
5 CREATE (u)-[:REALIZOU]->(t), (u)-[:USOU_IP]->(ip)
```

neo4j\$ MATCH (u:Usuario), (t:Transacao), (ip:EnderecoIP) WHERE u.id = "..."

Created 2 relationships, completed after 8 ms.

Table

Warn

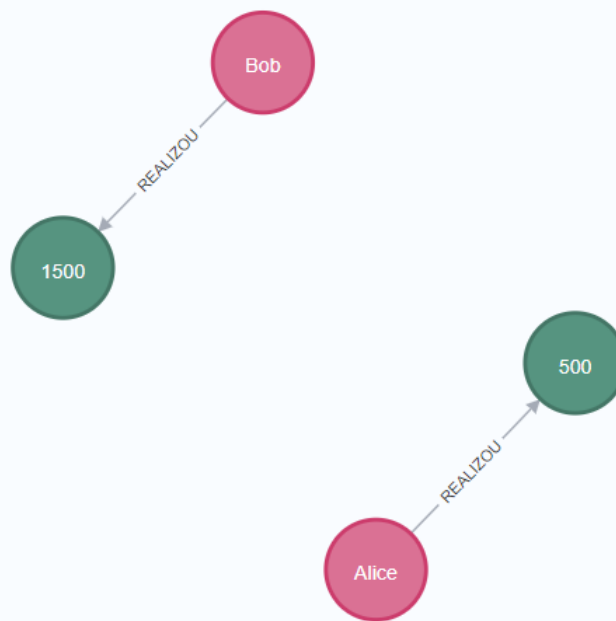
```
neo4j$ MATCH p=()-[r:REALIZOU]→() RETURN p LIMIT 25
```

Graph

Table

Text

Code



```
1 MATCH (u:Usuario), (t:Transacao), (ip:EnderecoIP)
2
3 WHERE u.id = "user3" AND t.id = "tx3" AND ip.endereco =
  "192.168.0.3"
4
5 CREATE (u)-[:REALIZOU]→(t), (u)-[:USOU_IP]→(ip)
```

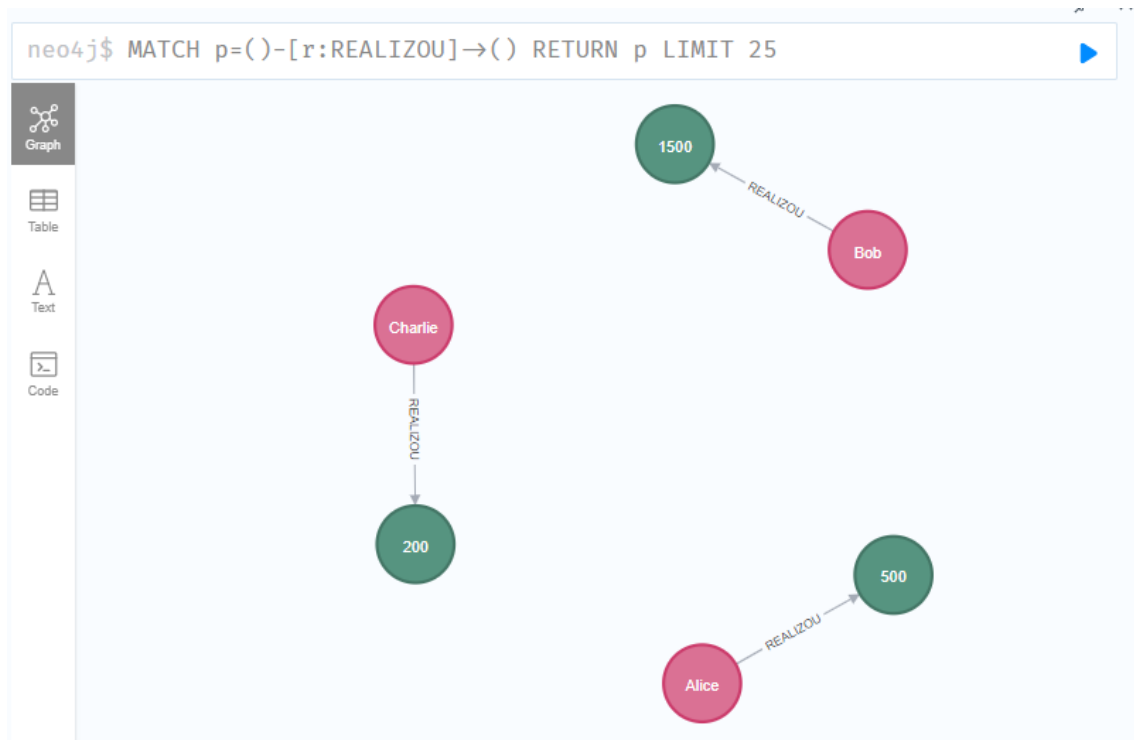
```
neo4j$ MATCH (u:Usuario), (t:Transacao), (ip:EnderecoIP) WHERE u.id = "..."
```

Table

Warn

Code

Created 2 relationships, completed after 8 ms.



Agora iremos fazer algumas buscar cypher dos nós e relacionamnetos criados acima:

```

1 MATCH (u:Usuario), (t:Transacao), (ip:EnderecoIP)
2
3 WHERE u.id = "user3" AND t.id = "tx4" AND ip.endereco =
4   "192.168.0.3"
5 CREATE (u)-[:REALIZOU]→(t), (u)-[:USOU_IP]→(ip)

```

Created 2 relationships, completed after 1 ms.

```

1 MATCH (u:Usuario)-[:REALIZOU]→(t:Transacao)-[:USOU_IP]→
2   (ip:EnderecoIP)
3
4 WHERE t.valor > 1000
5
6 AND NOT EXISTS((u)-[:REALIZOU]→(:Transacao {valor: 499}))
7
8 RETURN u.nome AS Usuario, t.id AS Transacao, t.valor AS Valor,
9   ip.endereco AS EnderecoIP

```

Esta consulta busca por transações com valores acima de 1000, mas que não foram precedidas por uma transação de exatamente 1000 para o mesmo usuário.

Isso pode indicar uma atividade fraudulenta.

