

Banco de Dados Relacional e Não Relacional

Aula - Banco de Dados Orientado a Grafo Neo4j

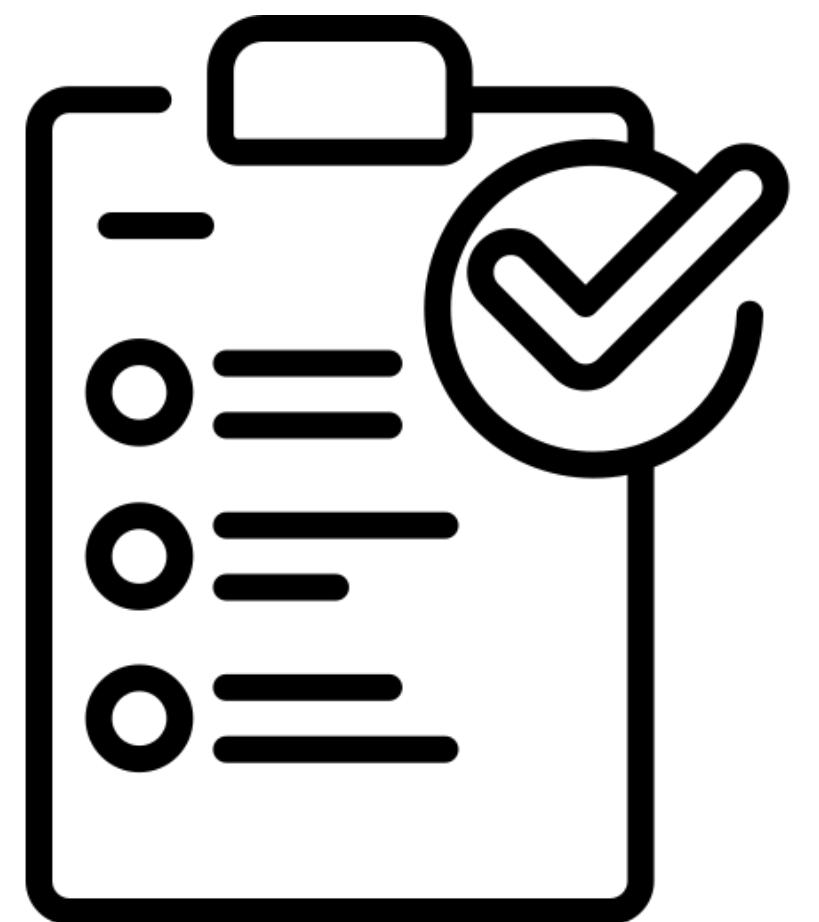
Prof. Anderson Theobaldo





Nesta Aula

- Definição de banco de dados NoSQL tipo grafo
- Características
- Casos de Uso
- Bancos mais populares
- Banco de dados Neo4j
- Demonstração



Definição

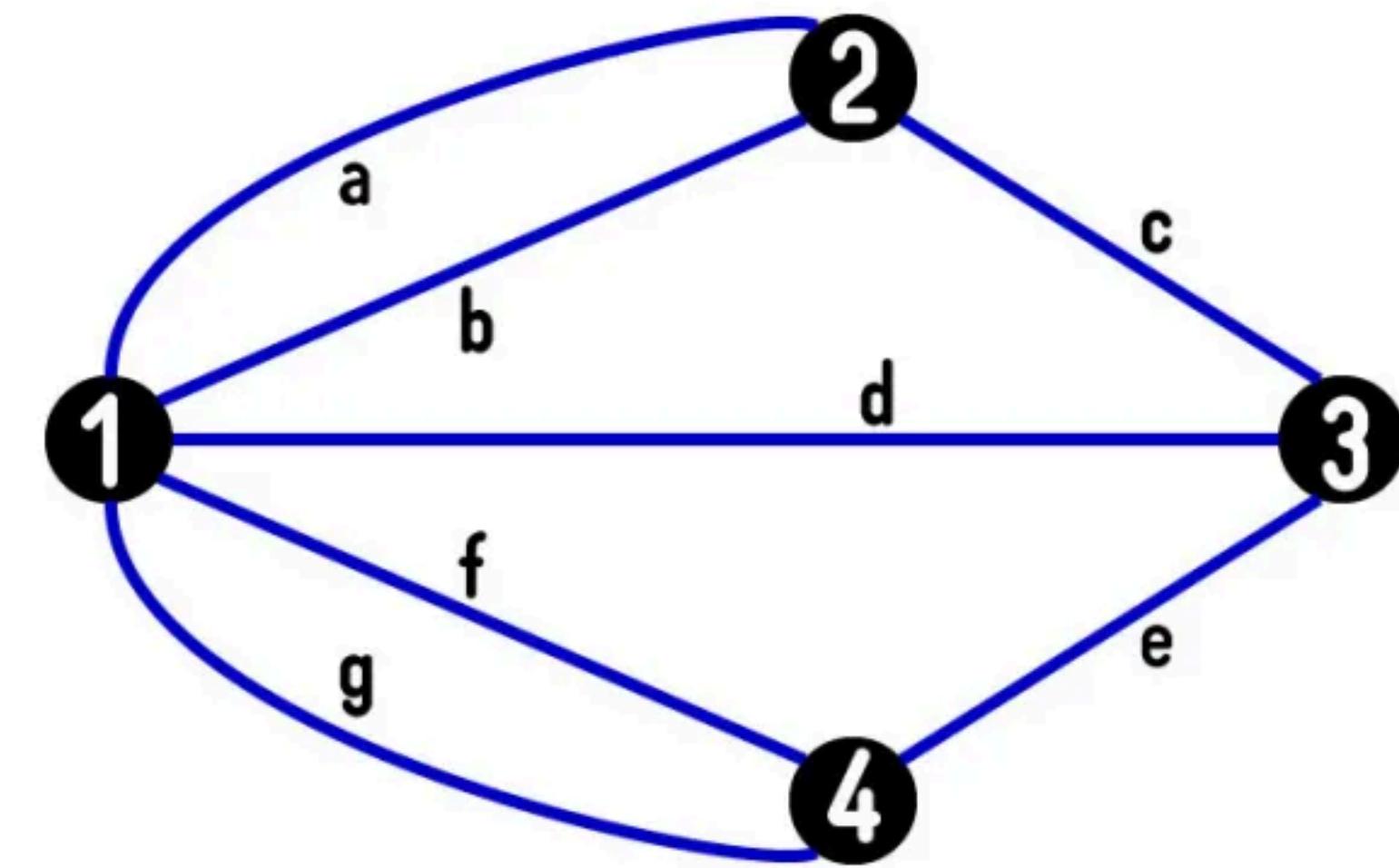
- Os grafos são poderosas estruturas de dados que modelam relações e conexões entre entidades.
 - São frequentemente utilizados para modelar relações em uma variedade de domínios, incluindo redes sociais, sistemas de transporte, circuitos elétricos, entre outros.





Definição

- Grafo é uma estrutura composta por um conjunto de vértices ou nós, e um conjunto de arestas que conectam esses vértices.
- Os vértices representam entidades distintas, enquanto as arestas representam as relações ou conexões entre essas entidades.
- Dependendo das características das arestas, os grafos podem ser direcionados (com arestas que têm uma direção) ou não direcionados (com arestas sem uma direção específica).





Características

- Banco de dados NoSQL do tipo grafo (graph database) é baseado na teoria de grafo sendo útil em contextos que sejam ricos em relacionamentos.
- O foco é modelar e armazenar dados sobre relacionamentos.
- Fornece uma fonte rica para algoritmos e aplicativos que precisam manipular dados com essa estrutura.
- Situações em que os relacionamentos se tornam tão importantes quanto os próprios dados são os campos em que esses tipos de bancos se destacam.



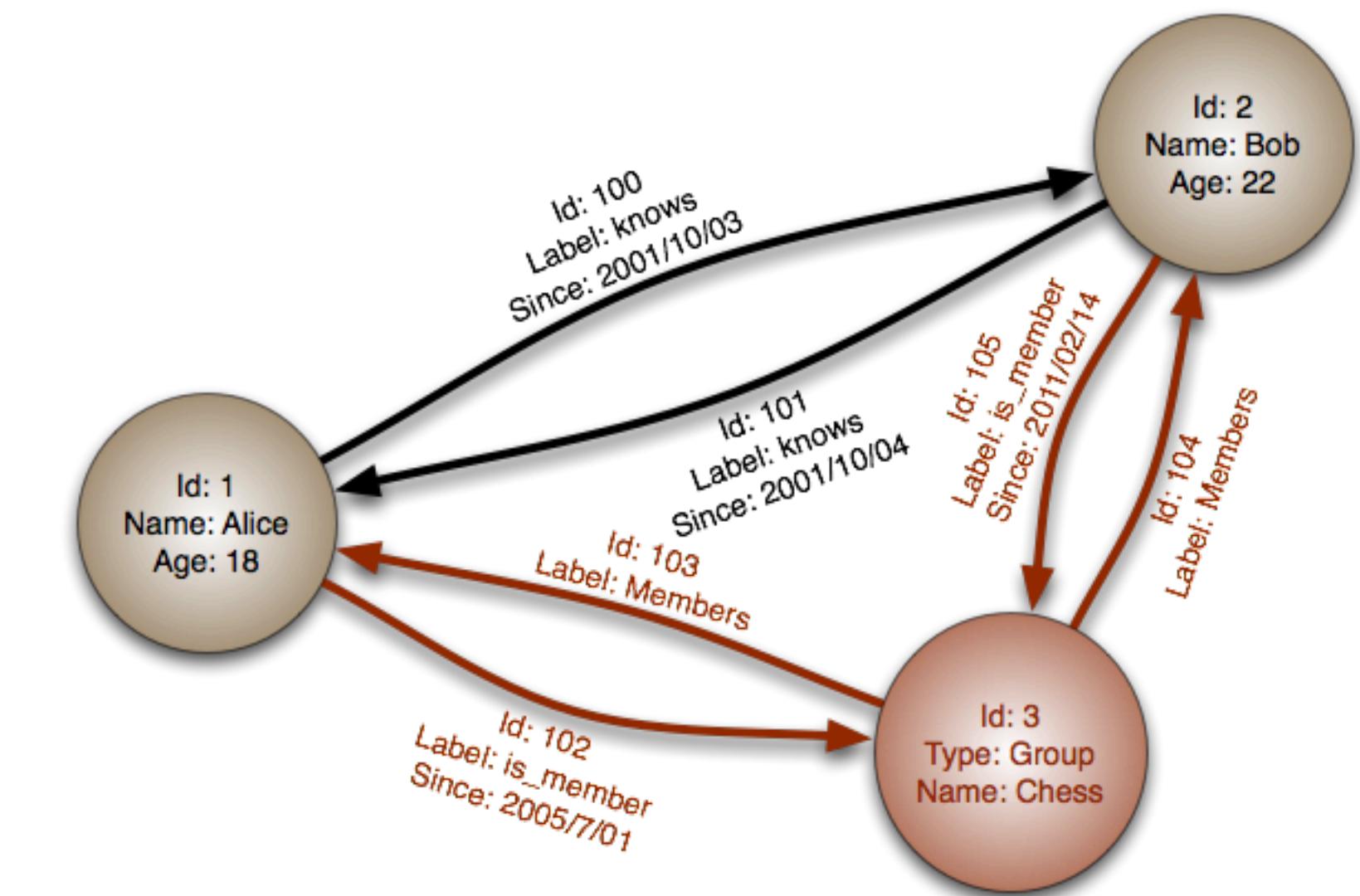
Características

- Não requerem estruturas de dados predefinidas.
- Não suportam a linguagem SQL.
- Otimizados para fornecer velocidade de processamento para dados intensivos em relacionamento.
- Não apresentam escalabilidade tão aprimorada quanto os outros tipos de bancos de dados NoSQL.



Principais Componentes

- Os componentes primários dos bancos de dados de grafo são nós, arestas e propriedades.
- Um nó corresponde à ideia de uma instância de entidade relacional. O nó é uma instância de algo sobre o qual queremos manter os dados.
- As propriedades são como atributos no qual desejamos armazenar sobre o nó. Todos os nós podem ter propriedades, mas nem todos os nós precisam ter as mesmas propriedades.
- Uma aresta é um relacionamento entre nós e podem estar em uma direção ou ser bidirecionais.



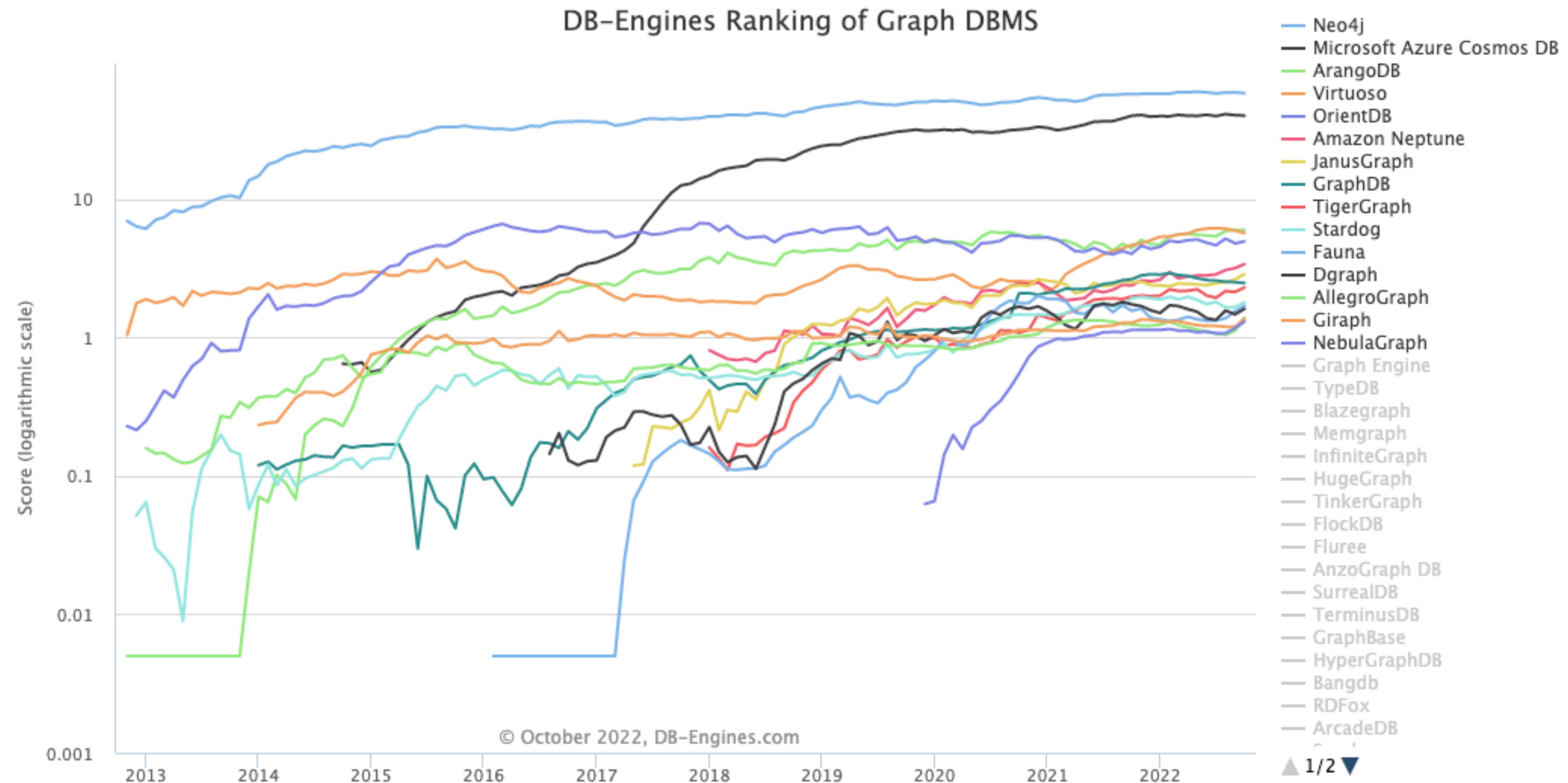


Casos de Uso

- O interesse pelas bases de dados grafo teve origem na área das redes sociais. Porém, vão além disso, abrangendo questões que dependem do rastreamento de relacionamentos complexos entre objetos:
- Casos de uso comuns:
 - Gerenciamento de recomendações
 - Logística e roteamento
 - Detecção de fraudes
 - Gerenciamento de identidade e acesso



Bancos Mais Utilizados





PUC Minas





- Criado em 2007, mantido pela Neo Technologies
- Software livre
- Usa como modelo grafos de propriedades rotulados
- Tem modelo de armazenamento e processamento nativos para grafos
- Usa uma linguagem de consulta chamada Cypher



Anatomia

Nós

- É algo sobre o que queremos manter os dados.

Label

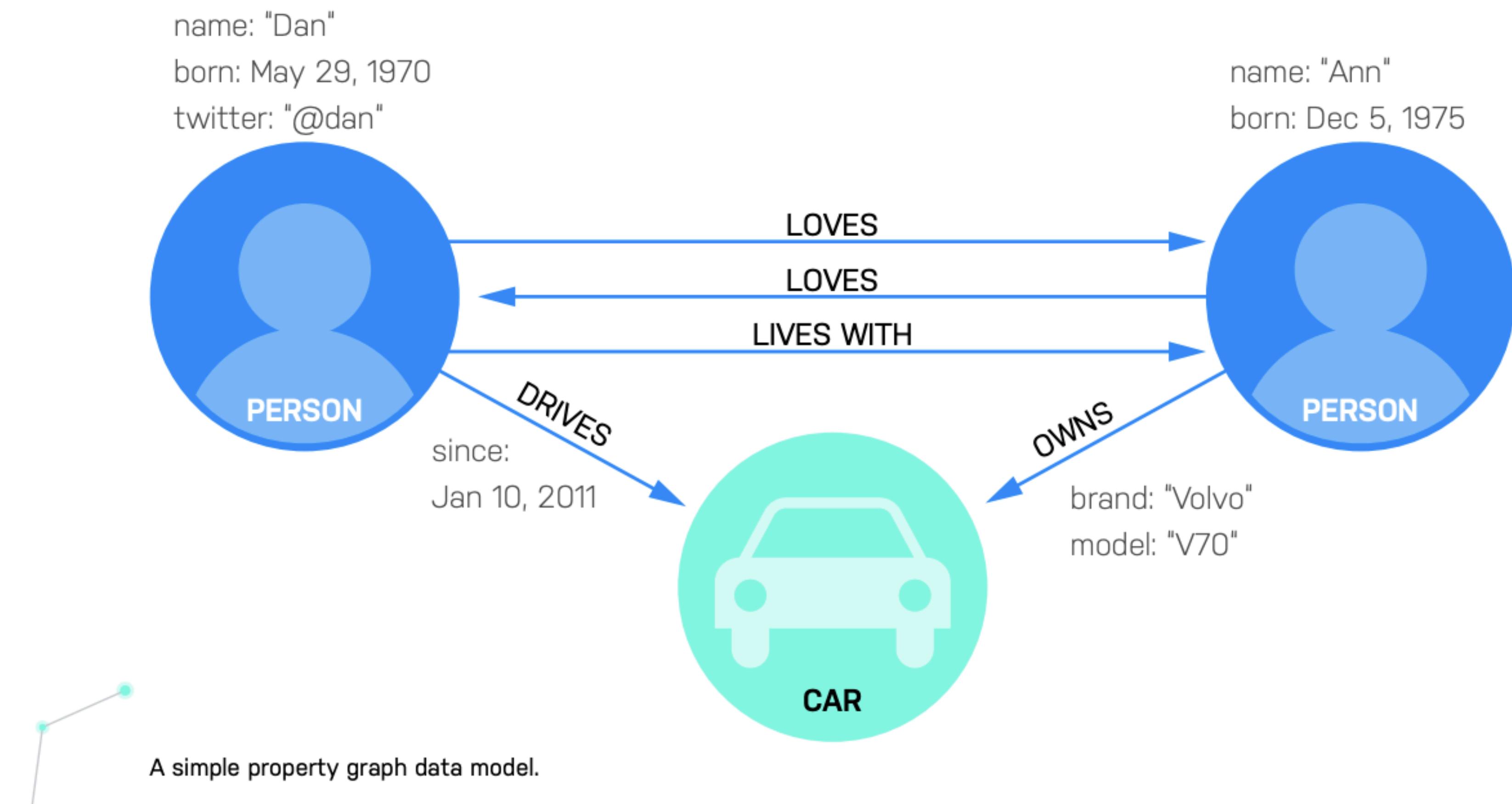
- Agrupam nós do mesmo tipo

Relacionamento

- Relacionam nós e podem estar em uma direção ou ser bidirecionais

Propriedades

- São representados como pares de chave/valor que podem ser associados ao nó ou relacionamento



Linguagem Cypher

- É uma linguagem de consulta de grafos declarativa que permite consultas, atualizações e administração expressivas e eficientes do grafo.
- Consultas de banco de dados altamente complicadas podem ser facilmente expressas, permitindo que se concentre em seu domínio.
- É usada para encontrar padrões nos relacionamentos de um grafo.
- Tem comandos inspirados em outras linguagens de consulta conhecidas, como a SQL e a SPARQL
- Sua especificação é aberta.



Instalação

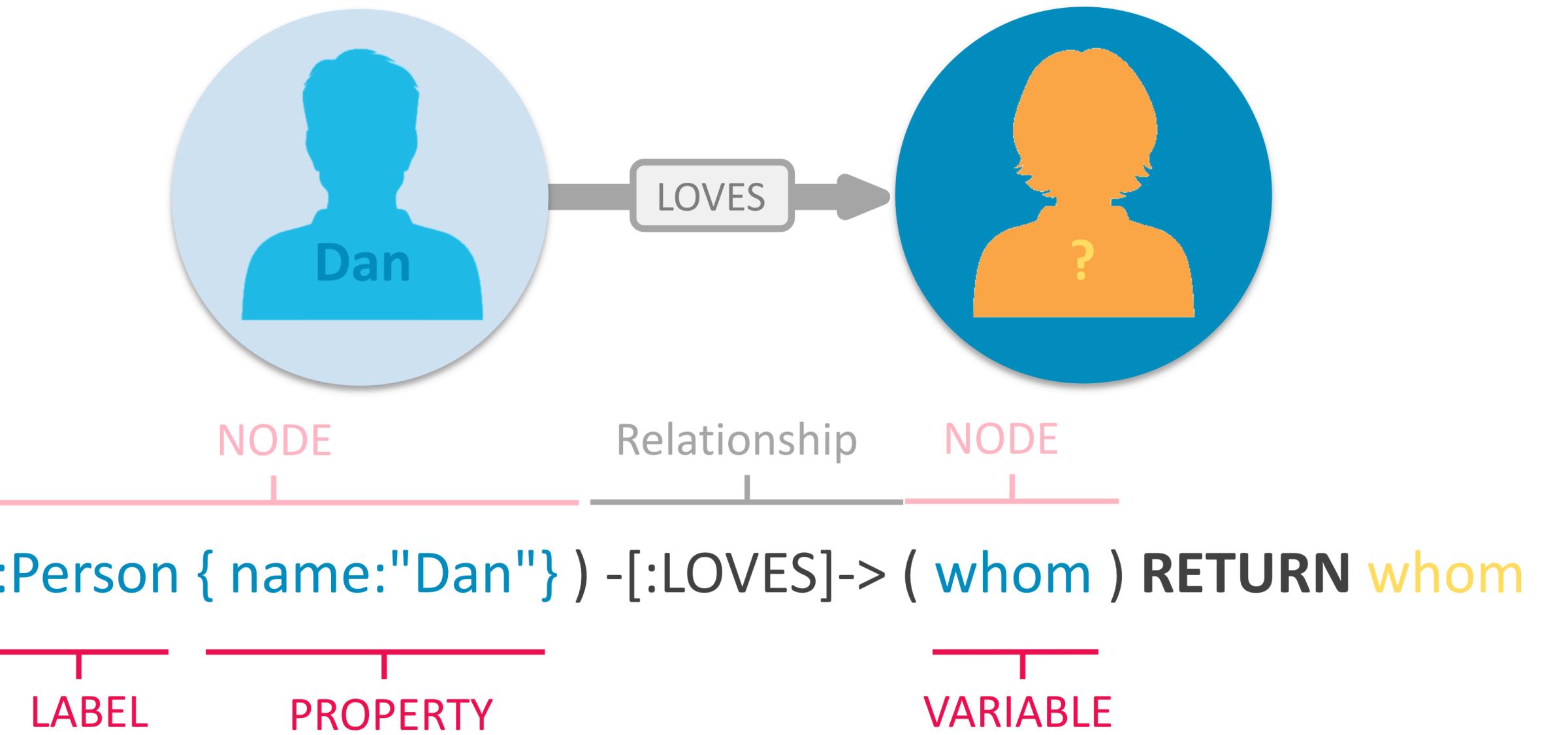
- Instale o Neo4j a partir:
 - <https://neo4j.com/download-center/#community>
- Neo4j as a service
 - <https://neo4j.com/cloud/platform/aura-graph-database/?ref=neo4j-home-hero>

Obs: a partir da versão 4.0 do Neo4j, a criação de um novo banco de dados usando o comando CREATE DATABASE só está disponível nas edições Enterprise e Developer. Se estiver usando a edição Community, essa funcionalidade não estará disponível e precisará criar apenas um novo grafo no banco de dados padrão.



Sintaxe Básica do Cypher

- Nós são denotados por um par de parênteses.
- Uma variável de nó pode ser usada em outros lugares dentro do mesmo comando em que foi definida.
- O rótulo (Label) funciona como a definição de um tipo ou papel para o nó.
 - Possibilita agrupar os nós.
 - Bastante usado como filtro nas consultas.
- Um nó pode ter mais de um rótulo (Label).
- Com exceção dos parênteses, todos os demais elementos de definição de nó são opcionais.

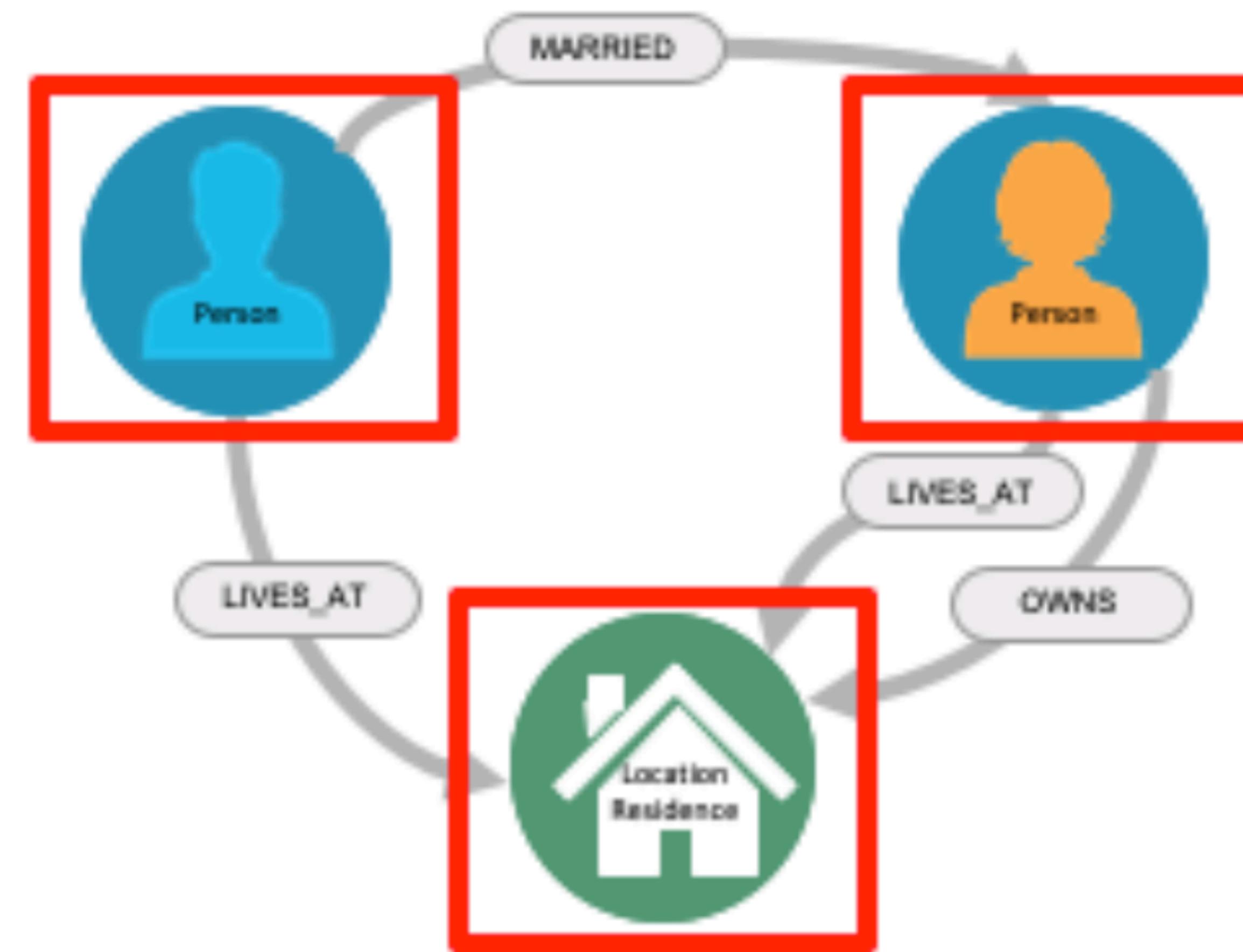




Sintaxe Básica do Cypher

Sintaxe Node

(
(p)

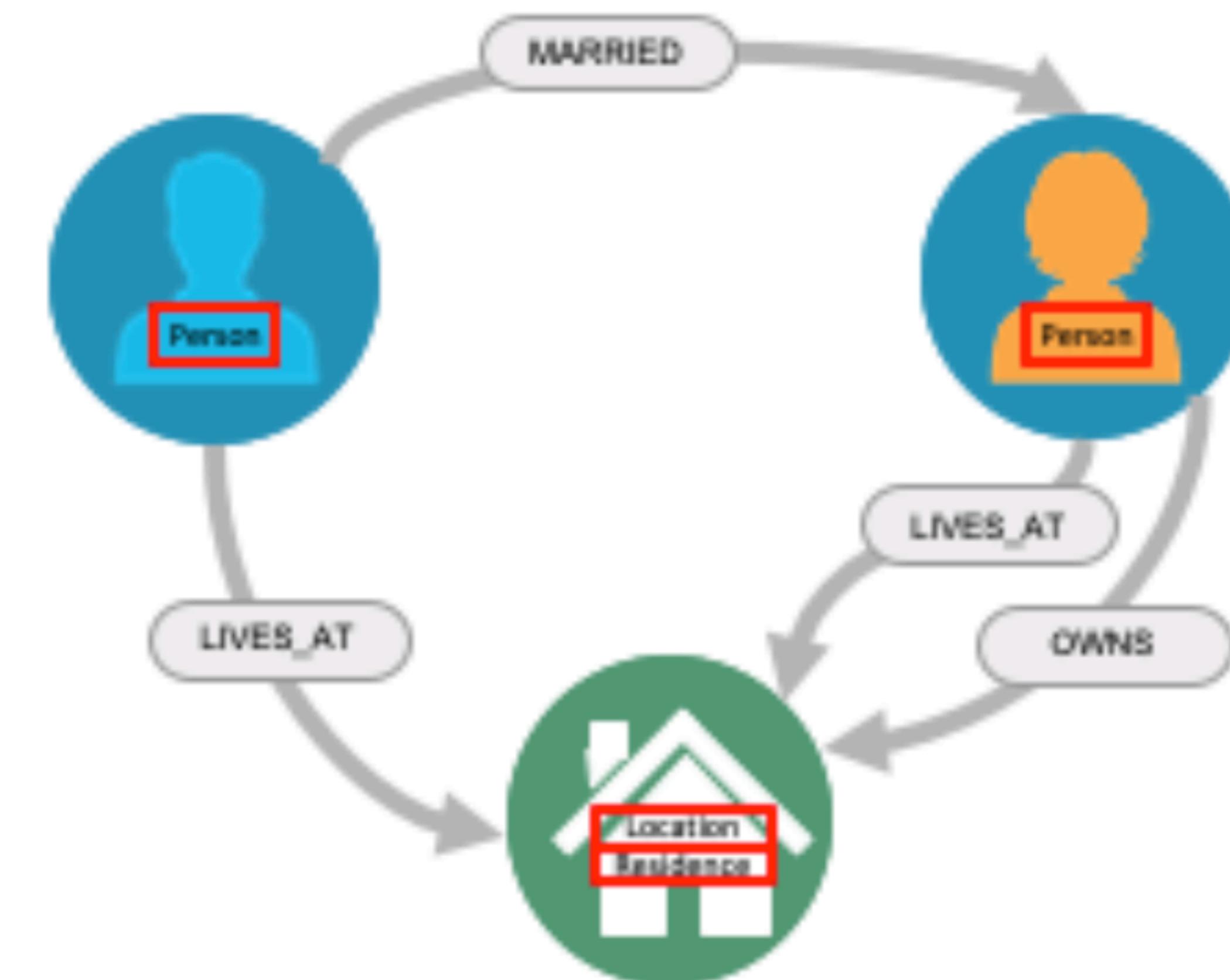




Sintaxe Básica do Cypher

Sintaxe Label(Rótulo)

```
( :Person)  
(p:Person)  
( :Location)  
(l:Location)  
(x:Residence)  
(x:Location:Residence)
```





Sintaxe Básica do Cypher

Comentários:

```
// anonymous node not be referenced later in the query
()

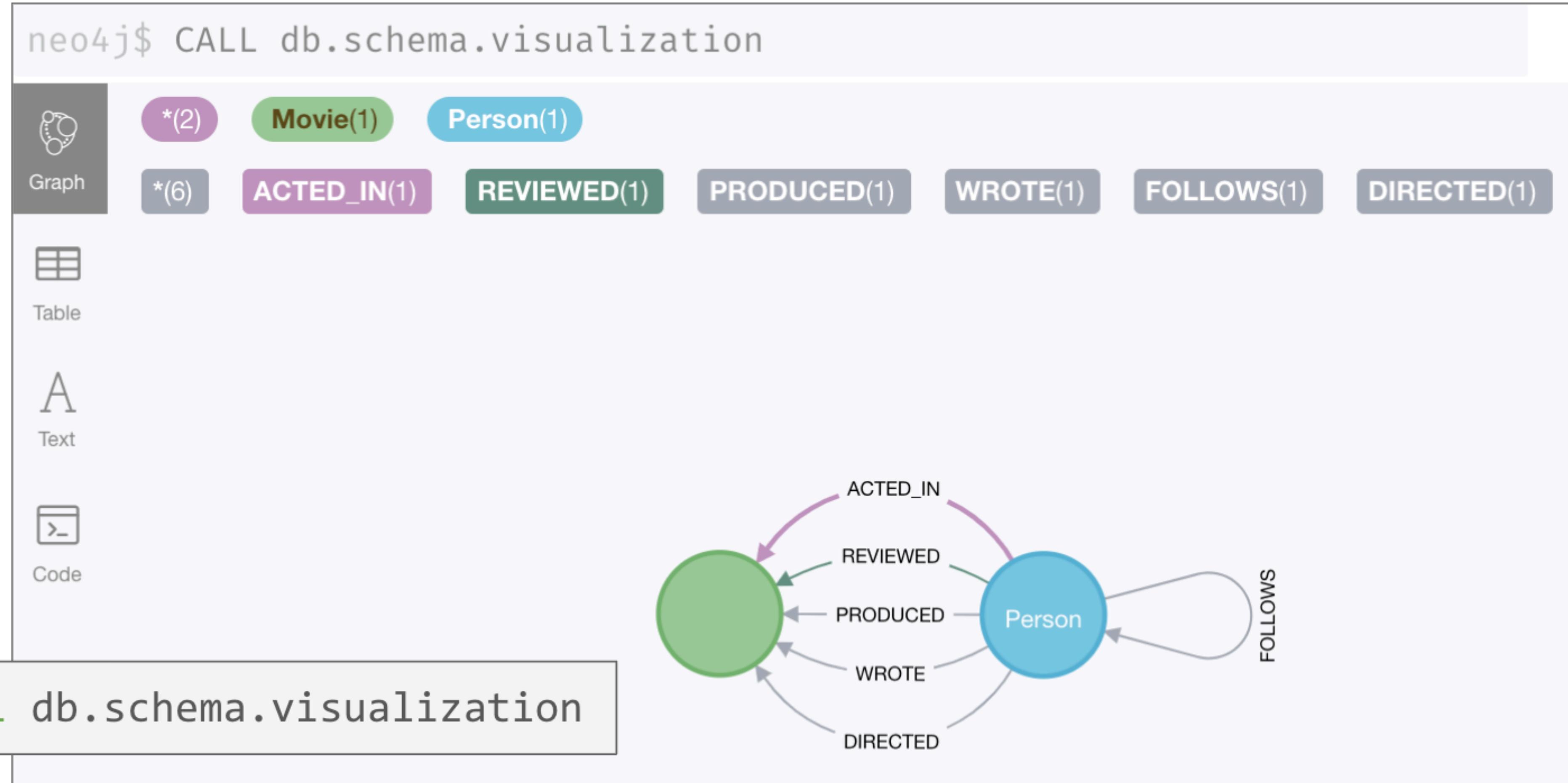
(p) // variable p, a reference to a node used later

(:Person)      // anonymous node of type Person
(p:Person)     // p, a reference to a node of type Person
                // p, a reference to a node of types Actor and Director
(p:Actor:Director)
```



Sintaxe Básica do Cypher

Visualizando o modelo de dados:





Sintaxe Básica do Cypher

Match e Return: utilizado para consulta

Sintaxe:

```
MATCH (variável)  
RETURN variável
```

```
MATCH (variável: Rótulo)  
RETURN variável
```

Recuperando todos os nodes:

```
MATCH (n)  
RETURN n
```

Recuperando todos os nodes

Person:

```
MATCH (p:Person)  
RETURN p
```

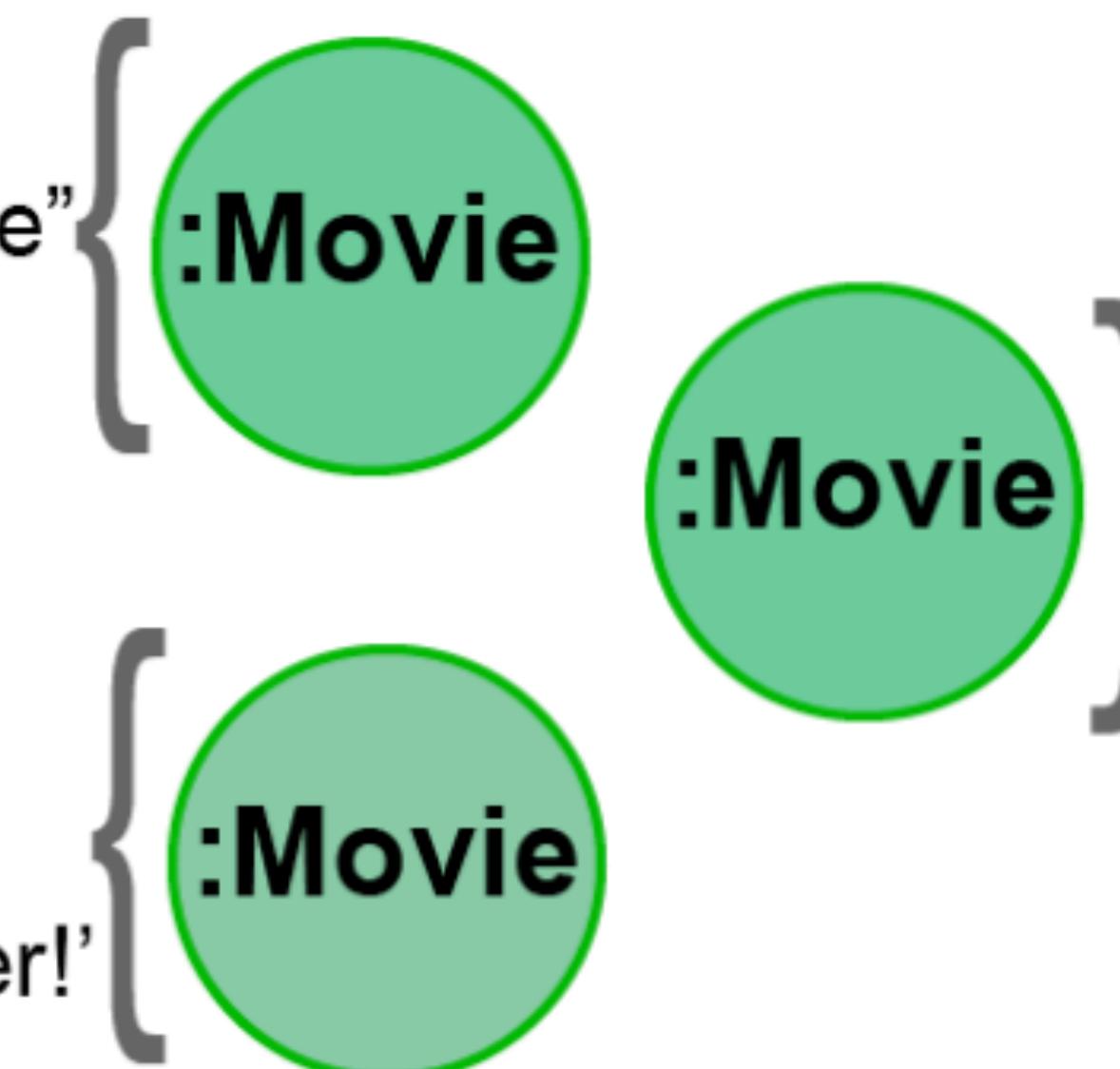


Sintaxe Básica do Cypher

Sintaxe de propriedade:

title: "Something's Gotta Give"
released: 2003

title: 'V for Vendetta'
released: 2006
tagline: 'Freedom! Forever!'



title: 'The Matrix Reloaded'
released: 2003
tagline: 'Free your mind'



Sintaxe Básica do Cypher

Filtrando através de propriedade:

```
MATCH (n:Person {born: 1970})  
RETURN n
```



Propriedade

Obs: para utilizar várias propriedades no filtro, basta separá-las por vírgula.

neo4j\$ MATCH (p:Person {born: 1970}) RETURN p

Graph *(4) Person(4)

Table

A Text

Code

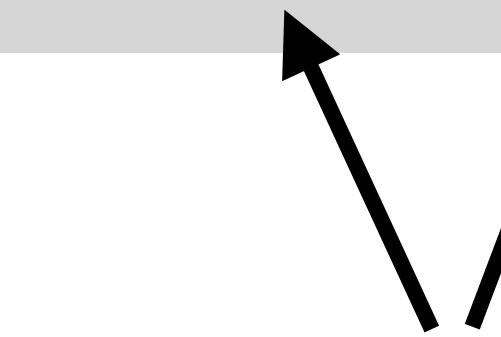
Person <id>: 236 born: 1970 name: Ethan Hawke



Sintaxe Básica do Cypher

Retornando propriedades específicas

```
MATCH (n:Person {born: 1970})  
RETURN n.name, p.born
```



Propriedades retornadas

Table

	p.name	p.born
1	"Lana Wachowski"	1965
2	"Tom Tykwer"	1965
3	"John C. Reilly"	1965

A

Text

Code

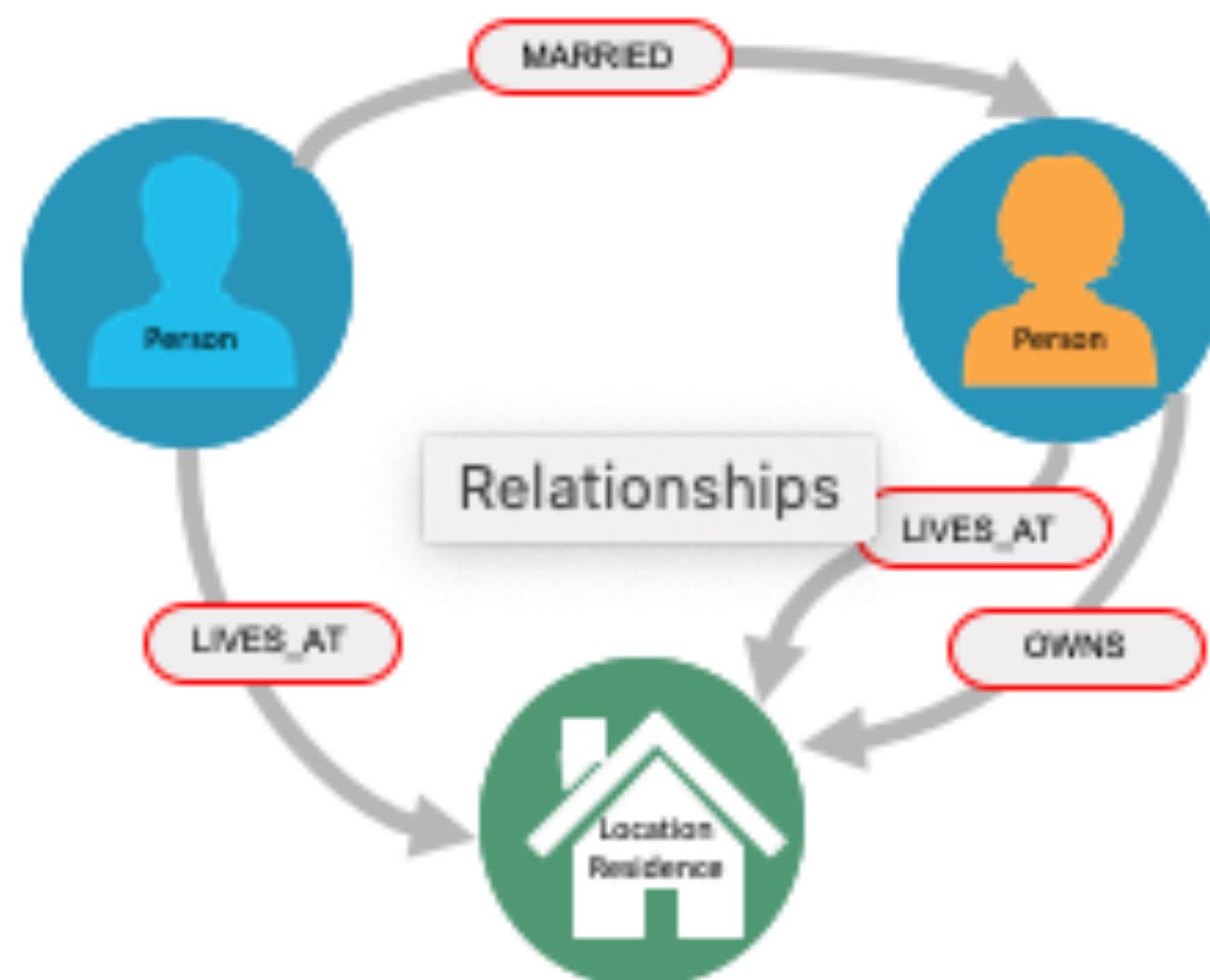
Started streaming 3 records after 1 ms and completed after 3 ms.



Sintaxe Básica do Cypher

Relacionamentos

- Conexão direcionada entre dois nodes
- Relacionamento tem um tipo
- Relacionamentos podem ter propriedades





Sintaxe Básica do Cypher

Sintaxe Relacionamento:

```
( )          // retorna um node  
( ) - - ()    // retorna nodes que tem algum tipo de relacionamento  
( ) - -> ()   // retorna nodes do lado esquerdo que tem algum tipo de relacionamento com nodes do lado direito  
( ) <- - ()   // retorna nodes do lado direito que tem algum tipo de relacionamento com nodes do lado esquerdo
```



Sintaxe Básica do Cypher

Usando relacionamento na consulta:

```
MATCH (n:Person) - [rel:ACTED_IN] -> (m:Movie {title: 'The Matrix'})  
RETURN p, rel, m
```

Recupera todas as pessoas (Person) que atuaram no filme ‘The Matrix’, retornando os nodes e relacionamentos.



Demonstração

- Criar uma nova conta para Neo4j AuraDB através do link:
<https://neo4j.com/cloud/platform/aura-graph-database/?ref=nav-get-started-cta>

The screenshot shows the Neo4j AuraDB landing page. At the top, there's a banner for the NODES 2023 conference. Below it, the Neo4j logo is on the left, followed by navigation links: Products, Use Cases, Developers & Data Scientists, Pricing, Learn, Contact Us, and Get Started Free. A large graphic in the center features a grid of mobile devices connected by a network of lines, with the text "THE FASTEST PATH TO GRAPH". The Neo4j aura DB logo is prominently displayed. Below the logo, a description reads: "The world's leading graph database as a fully-managed cloud service — zero-admin, globally available and always-on." A red-bordered "Start Free" button is located at the bottom left. The URL in the browser bar is https://neo4j.com/cloud/platform/aura-graph-database/?ref=nav-get-started-cta.

The screenshot shows the Neo4j login page. It features a "Get Started" heading with a sub-instruction "Log in to Neo4j to continue". There is an "Email address" input field with a placeholder "Email" and a "Login" button below it. A "Sign up" link is located just below the login button. A "Continue with Google" button is at the bottom. On the right side, there is a decorative graphic of nodes connected by dashed lines.



Demonstração

Criar uma nova conta para Neo4j AuraDB através do link:

[https://login.neo4j.com/u/login?](https://login.neo4j.com/u/login?state=hKFo2SA2THZTYWxhcUNYcGM4cmstV3gxVzItVIM1LXZid19PdaFur3VuaXZlcnNhbC1sb2dpbqNoaWTZIFJ4eVJqbVZ1UjB2NWtCcGU2bDM4WGVRLVzmOVI5em1Qo2NpZNkgV1NMczYwNDdrT2pwVVNXODNnRFooSnlZaElrNXpZVG8)

[state=hKFo2SA2THZTYWxhcUNYcGM4cmstV3gxVzItVIM1LXZid19PdaFur3VuaXZlcnNhbC1sb2dpbqNoaWTZIFJ4eVJqbVZ1UjB2NWtCcGU2bDM4WGVRLVzmOVI5em1Qo2NpZNkgV1NMczYwNDdrT2pwVVNXODNnRFooSnlZaElrNXpZVG8](https://login.neo4j.com/u/login?state=hKFo2SA2THZTYWxhcUNYcGM4cmstV3gxVzItVIM1LXZid19PdaFur3VuaXZlcnNhbC1sb2dpbqNoaWTZIFJ4eVJqbVZ1UjB2NWtCcGU2bDM4WGVRLVzmOVI5em1Qo2NpZNkgV1NMczYwNDdrT2pwVVNXODNnRFooSnlZaElrNXpZVG8)

The screenshot shows the Neo4j Aura login interface. On the left, the landing page for Neo4j Aura is visible, highlighting its features: Lightning-fast query performance, Fully-managed updates and patches, Easy scalability, on-demand, Robust security, reliability and ACID compliance, and Built-in tools to learn, build, and visualize. On the right, the login form is displayed with fields for 'Email address' and 'Password'. Below the password field is a 'Forgot password?' link. A large blue 'Login' button is centered. To the right of the login form is a 'Continue with Google' button. At the bottom of the login form, there's a link to 'Sign up' if you don't have an account. The URL in the browser bar is <https://login.neo4j.com/u/login?state=hKFo2SA2THZTYWxhcUNYcGM4cmstV3gxVzItVIM1LXZid19PdaFur3VuaXZlcnNhbC1sb2dpbqNoaWTZIFJ4eVJqbVZ1UjB2NWtCcGU2bDM4WGVRLVzmOVI5em1Qo2NpZNkgV1NMczYwNDdrT2pwVVNXODNnRFooSnlZaElrNXpZVG8>.



Demonstração

- Ao conectar no Neo4j, criar uma na instância selecionando no botão **New Instance**.
- Como nosso propósito será realizar somente algumas demonstrações, selecione o tipo de instância **Free**.

The screenshot shows the Neo4j Aura web interface. On the left, there's a sidebar with links for 'Instances' (highlighted with a red box), 'Connect', 'AuraDS', 'Instances', and 'Getting Started'. The main area has tabs for 'Instances' (highlighted with a red box) and 'New Instance'. The URL in the browser bar is <https://console.neo4j.io/?product=aura-db>.

The screenshot shows the 'Create an AuraDB instance' configuration page. On the left, there's a sidebar with links for 'Instances', 'Connect', 'AuraDS', 'Instances', and 'Getting Started'. The main area has a title 'Create an AuraDB instance' and a section 'Instance type:' with two options: 'Free' (selected) and 'Professional'. The 'Free' section includes details: 'For small development projects, learning, experimentation, and prototyping', a 'Create Free instance' button (highlighted with a red box), memory ('1 GB / instance'), graph size ('200K nodes, 400K relationships'), graph tools (checkbox checked), snapshots ('One on-demand only'), pause ('Automatic after 3 days of inactivity'), resume ('On-demand'), clone ('-'), API ('-'), and cloud provider options ('GCP'). The 'Professional' section includes details: 'For medium-scale applications in advanced development or production environments', a 'Select Professional instance' button, memory ('up to 64 GB / instance'), graph size ('Unlimited'), graph tools (checkbox checked), snapshots ('Daily scheduled and on-demand'), pause ('On-demand'), resume ('On-demand'), clone ('-'), API ('-'), and cloud provider options ('GCP, AWS, Azure'). A 'Private Beta' button is also visible.



Demonstração

- Copie ou faça o download da senha gerada, conforme é apresentado na tela abaixo.
- Obs.: não será possível obter a senha novamente a partir deste ponto.

The screenshot shows the neo4j aura web interface. The main navigation bar includes the neo4j aura logo, a 'New tenant' dropdown, and three icons. On the left, there are sections for 'AuraDB' (with 'Instances' selected) and 'AuraDS' (with 'Instances' and 'Getting Started' options). A central modal window is open, titled 'Credentials for Instance01'. It contains a red-bordered box with 'Username: neo4j' and a 'Generated password' field containing the value '0aoP3gmtj6yjbc [REDACTED]'. Below this is a yellow warning box with an exclamation mark and the text 'Note that the password will not be available after this point.' At the bottom of the modal are 'Close' and 'Download and continue' buttons.

neo4j aura

New tenant

AuraDB

Instances

Connect

AuraDS

Instances

Getting Started

Credentials for Instance01

Username: neo4j

Generated password

0aoP3gmtj6yjbc [REDACTED]

Note that the password will not be available after this point.

Close

Download and continue



Demonstração

- Para renomear a nova instância selecione a opção conforme demonstrado na figura.
- Selecionar o botão *Query* para iniciar o módulo de execução de instruções.

Instances [New Instance](#)

Instance01 Free
● Running
Neo4j version 5
Nodes 0 / 200000 (0%)
Relationships 0 / 400000 (0%)
Region Iowa, USA (us-central1)
Connection URI <code>neo4j+s://dd956517.databases.neo4j.io</code>

[Explore](#)
[Query](#)
[Import](#)

Rename

Reset to blank
Upgrade to Professional
Clone To New >
Clone To Existing >

Rename Lab_Neo4j

Instance Name

[Cancel](#) [Rename](#)



Demonstração

- Utilizar as credenciais criada na etapa anterior, conectar a nova instância.

The screenshot shows the Neo4j Browser interface. On the left, a sidebar titled "Database Information" displays the following sections:

- Node labels**: There are no labels in database.
- Relationship types**: No relationships in database.
- Property keys**: There are no properties in database.
- DBMS**: Information: [:sysinfo](#), Query List: [:queries](#).

The main area of the browser shows a terminal-like interface with the prompt "\$:server connect". Below it, a "Connect to Neo4j" dialog is open, containing the following fields:

- Connect URL**: neo4j+s://9ebc9ef1.databases.neo4j.io:7687
- Authentication type**: Username / Password
- Username**: neo4j
- Password**: (redacted)
- Connect** button

A blue banner at the top of the main area states: "Database access not available. Please use `:server connect` to establish connection. There's a graph waiting for you."

Demonstração

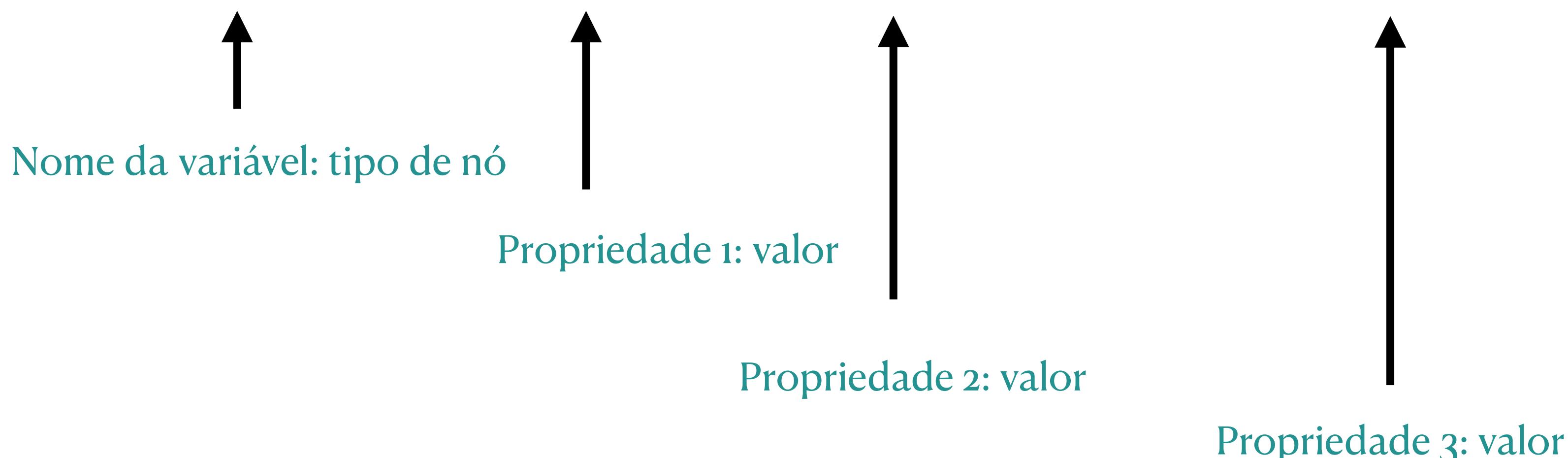
- Só há dois tipos de dados no Neo4J: o nó (semelhante ao documento no MongoDB) e o relacionamento (aresta).
- Um nós pode ter um tipo e possuir diversos atributos.

Demonstração - Criando nó

- Vamos criar 2 nós:
 - Um para o tipo Person(Pessoa) com os atributos name(nome) e born(nascimento)
 - E outro para o tipo Movie(Filme) com os atributos título(title), lançamento(released) e slogan(tagline).
- Execute o seguinte comando na interface:

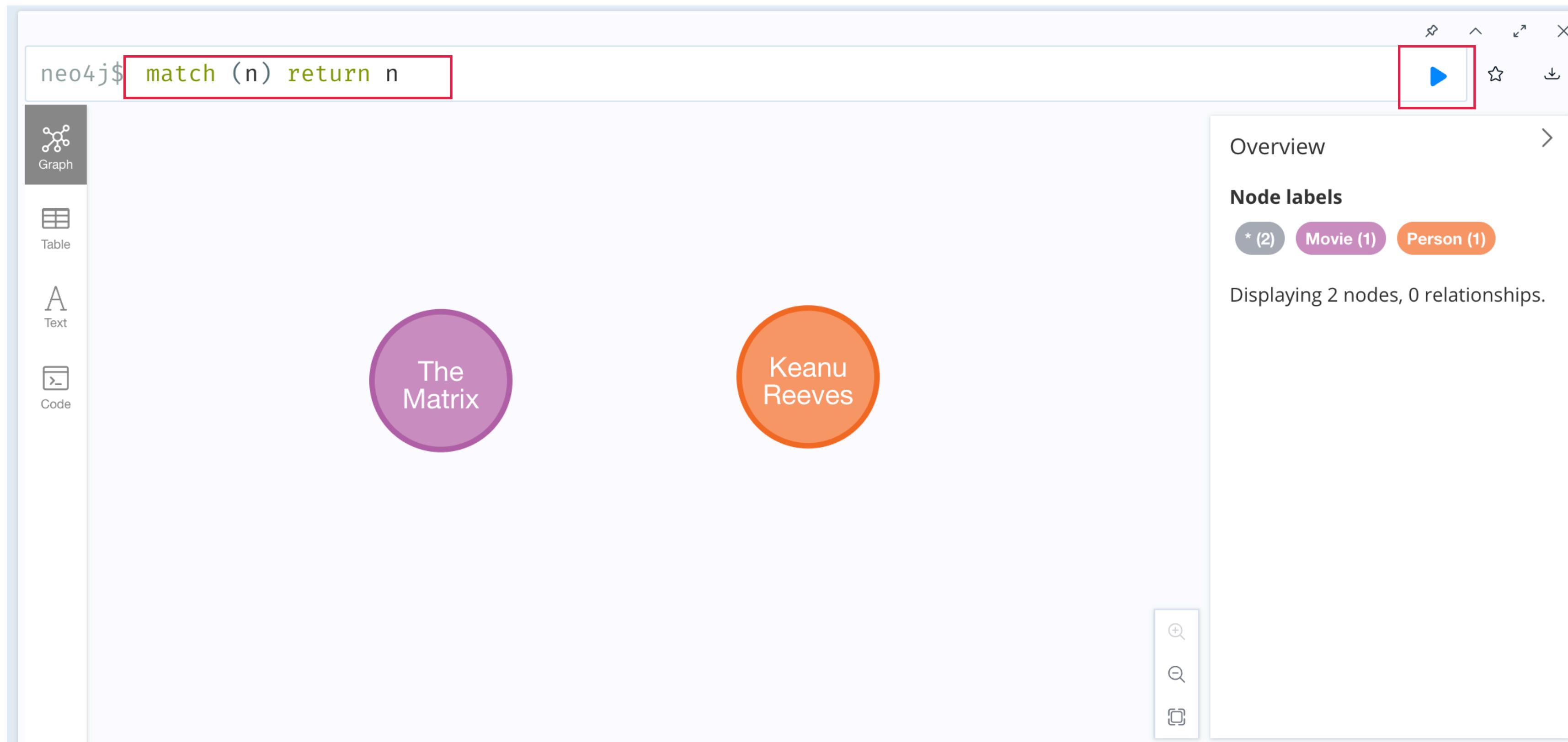
```
CREATE (:Person {name:'Keanu Reeves', born:1964})
```

```
CREATE ( TheMatrix:Movie { title: 'The Matrix', released: 1999, tagline: 'Welcome to the Real World' } )
```



Demonstração - Consultando os nós

- Para recuperar os nós criados, basta executar o seguinte comando na interface:

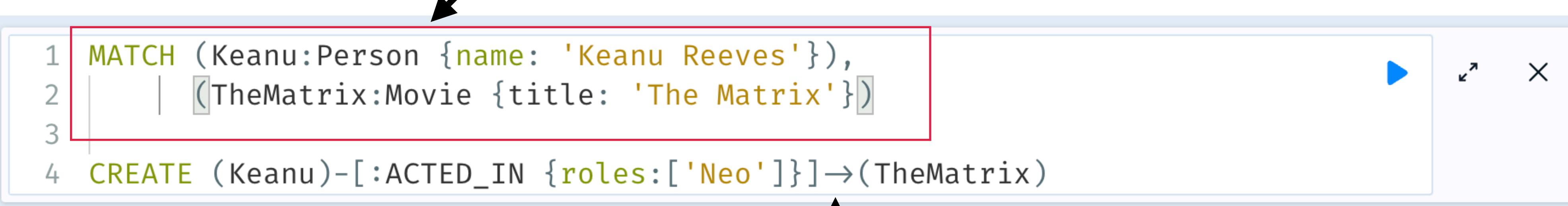


The screenshot shows the Neo4j browser interface. On the left, there's a sidebar with four tabs: Graph (selected), Table, Text, and Code. The Code tab contains the query `neo4j$ match (n) return n`. To the right of the query is a red rectangular box highlighting the play button icon. The main area displays two nodes: "The Matrix" (a purple circle) and "Keanu Reeves" (an orange circle). On the far right, there's an "Overview" section with the following details:

- Node labels:** * (2) (grey), Movie (1) (purple), Person (1) (orange).
- Displaying 2 nodes, 0 relationships.

Demonstração - Criando um relacionamento

- Para criar um relacionamento entre os dois nós (Movie e Person), execute o seguinte comando:



The screenshot shows a Cypher query in a database browser:

```
1 MATCH (Keanu:Person {name: 'Keanu Reeves'}),  
2     | (TheMatrix:Movie {title: 'The Matrix'})  
3  
4 CREATE (Keanu)-[:ACTED_IN {roles:['Neo']}]->(TheMatrix)
```

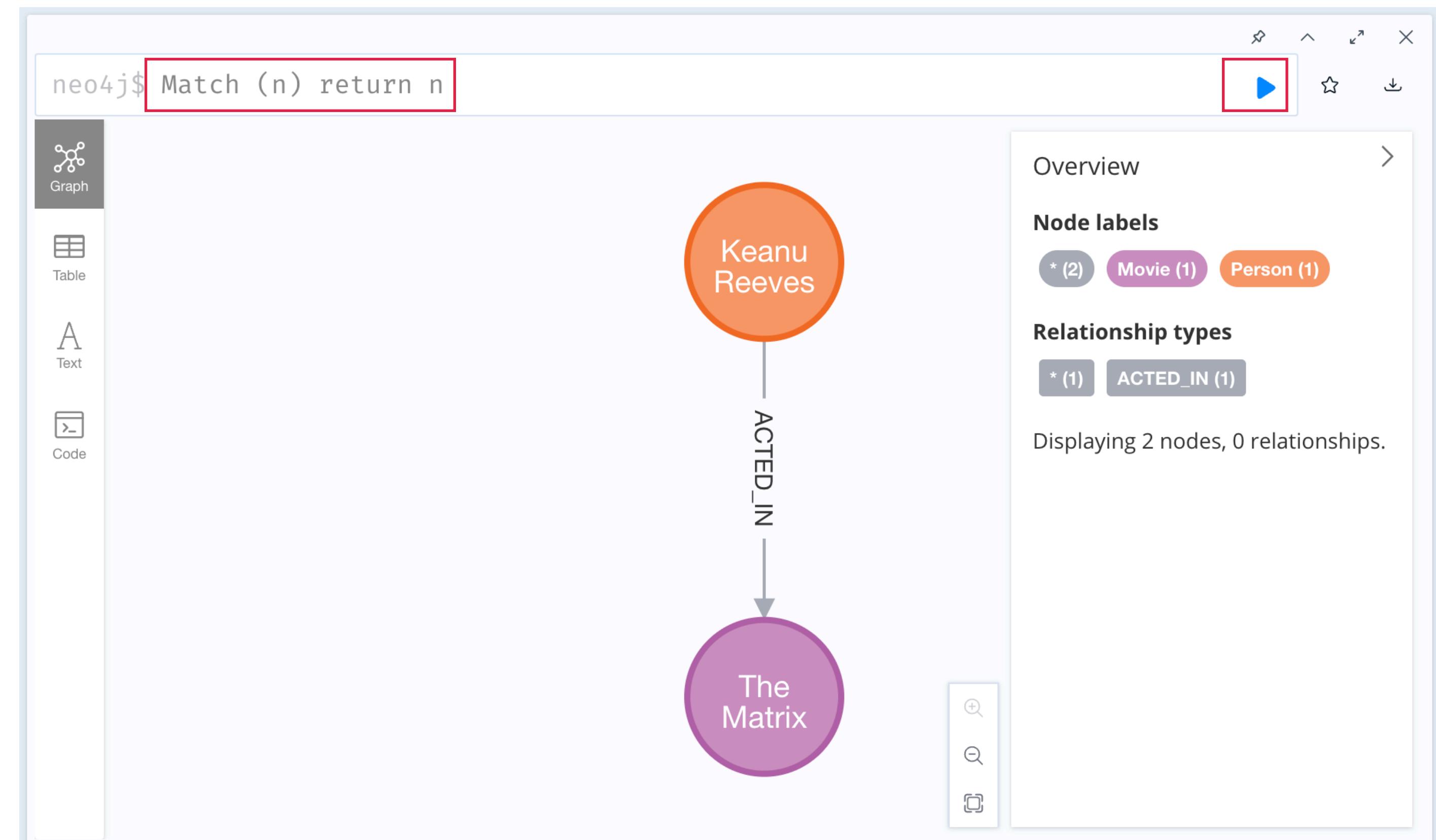
Annotations explain the components of the query:

- Nó 1**: Points to the node 'Keanu' in the first line of the query.
- Tipo do relacionamento**: Points to the relationship type ':ACTED_IN' in the fourth line.
- Propriedade do relacionamento**: Points to the property '{roles:['Neo']}' in the fourth line.
- Nó 2**: Points to the node 'TheMatrix' in the second line of the query.
- Direção do Relacionamento**: Points to the direction arrowhead in the relationship line of the query.
- Especifica a estrutura do grafo pesquisado**: Points to the first line of the query, explaining its purpose.



Demonstração - Consultando os nós relacionados

- Para consultar os nós relacionados, execute o seguinte comando na interface:





- Nesta etapa será possível criar os dados que irão popular a base de dados de filmes (movies).
- Será criado os nós atores (Person) e filmes (Movie) e diversos relacionamentos (ACTED_IN, DIRECTED, WROTE, entre outros) e propriedades (name, born e outros).

The screenshot shows the Neo4j Browser interface. On the left, there's a sidebar with icons for save, star, play, help, and settings. The main area has a title bar with a play button and an 'x' icon. Below the title bar is a code editor with the following Cypher query:

```
1 CREATE (TheMatrix:Movie {title:'The Matrix', released:1999, tagline:'Welcome to the Real World'})
2 CREATE (Keanu:Person {name:'Keanu Reeves', born:1964})
3 CREATE (Carrie:Person {name:'Carrie-Anne Moss', born:1967})
4 CREATE (Laurence:Person {name:'Laurence Fishburne', born:1961})
5 CREATE (Hugo:Person {name:'Hugo Weaving', born:1960})
6 CREATE (LillyW:Person {name:'Lilly Wachowski', born:1967})
7 CREATE (LanaW:Person {name:'Lana Wachowski', born:1965})
8 CREATE (JoelS:Person {name:'Joel Silver', born:1952})
9 CREATE
10 (Keanu)-[:ACTED_IN {roles:['Neo']}]→(TheMatrix),
11 (Carrie)-[:ACTED_IN {roles:['Trinity']}]→(TheMatrix),
```

Below the code editor is a command line interface with the prompt '\$:play movie-graph'. To the right of the command line is a section titled 'The Movie Graph' with the heading 'Create'. It contains the following text:

To the right is a giant code block containing a single Cypher query statement composed of multiple CREATE clauses. This will create the movie graph.

Below this text is a list of instructions:

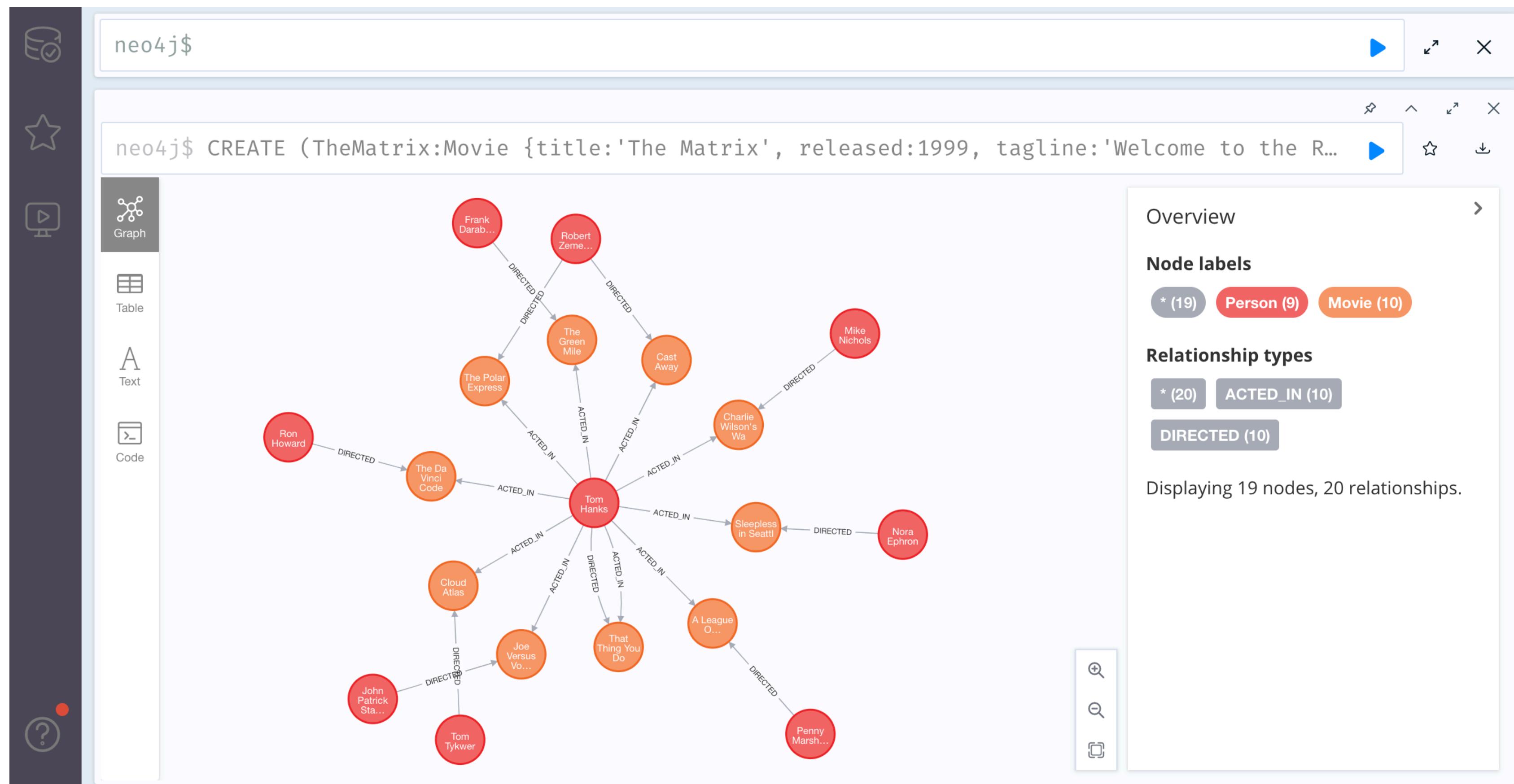
- Click on the code block
- Notice it gets copied to the editor above ↑
- Click the editor's play button to execute
- Wait for the query to finish
- WARNING: This adds data to the current database, each time it is run!

The bottom of the interface shows navigation buttons for back, forward, and search, along with a page number '2/8'.



Demonstração

- Base de Filmes (Movies) criada.





Demonstração

- Consultas para recuperar nós individuais com ou sem critério de filtro.

- As consultas podem ser escritas de forma alternativa:

```
MATCH (m:Movie)
```

```
WHERE m.title = "Cloud Atlas"
```

```
RETURN m
```

```
MATCH (p:Person)
```

```
WHERE p.name = "Tom Hanks"
```

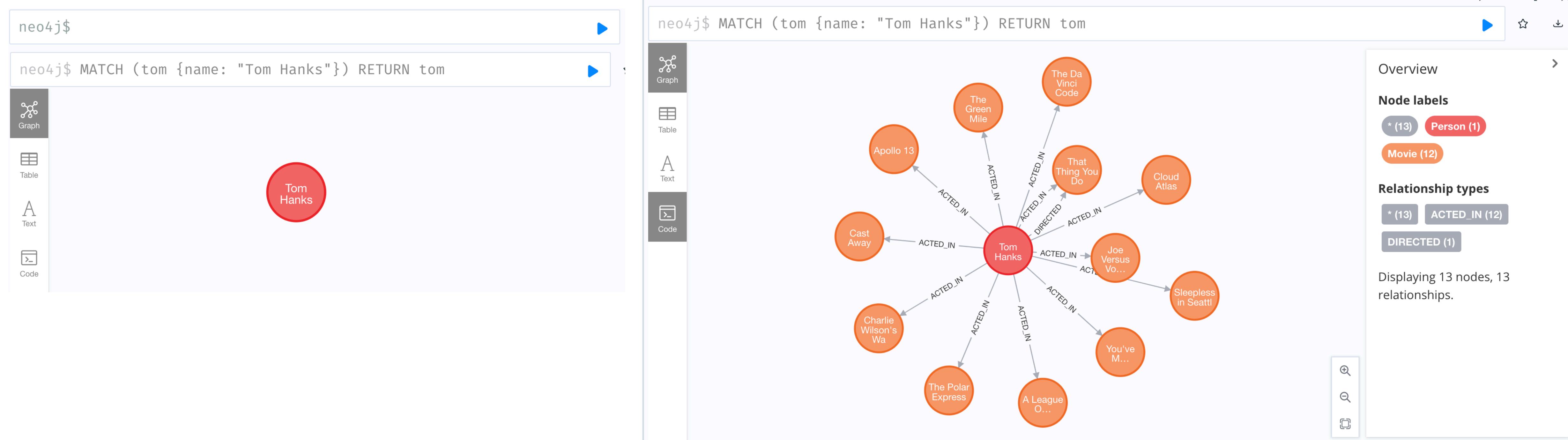
```
RETURN p
```

The screenshot shows the Neo4j browser interface. The top bar has a play button and a stop button. The query editor at the top left contains the command `$:play movie-graph`. The main area is titled "The Movie Graph". On the left, there's a sidebar with a "Find" section containing examples for finding individual nodes, movies, people, and movies from the 1990s. The first example in the "Find" section is highlighted with a green border and shows the query `MATCH (tom {name: "Tom Hanks"}) RETURN tom`. Below it are other examples: `MATCH (cloudAtlas {title: "Cloud Atlas"}) RETURN cloudAtlas`, `MATCH (people:Person) RETURN people.name LIMIT 10`, and `MATCH (nineties:Movie) WHERE nineties.released >= 1990 AND nineties.released < 2000 RETURN nineties.title`. The bottom right corner of the interface shows a navigation bar with icons for back, forward, and search.



Demonstração

- Um duplo clique no nó Tom Hanks irá recuperar todos os nós com relacionamento de primeiro nível.





Demonstração

- Consultas para recuperar nós com padrões estabelecidos.

The screenshot shows a Neo4j browser window with the following details:

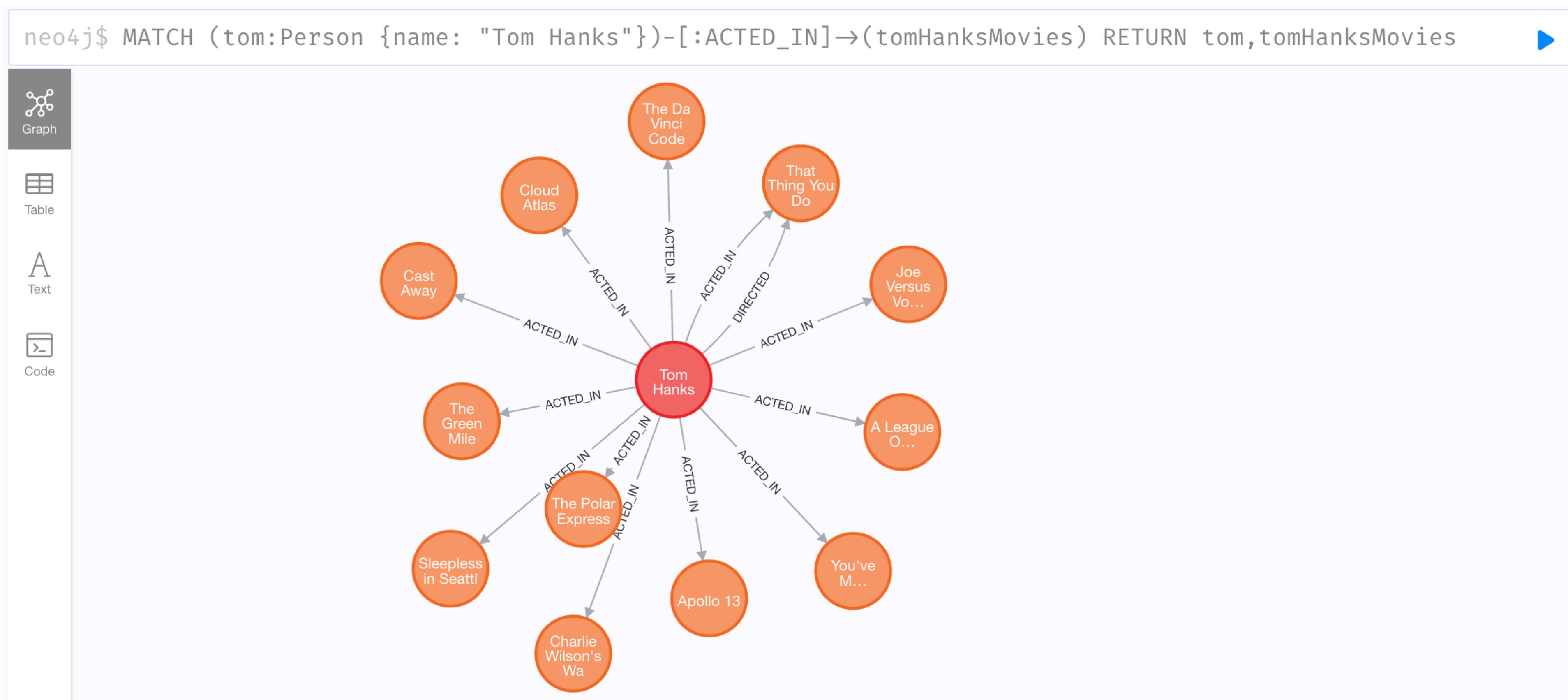
- Query Bar:** \$:play movie-graph
- Left Panel (Query):**
 - The Movie Graph
 - Finding patterns within the graph.
 - Actors are people who acted in movies
 - Directors are people who directed a movie
 - What other relationships exist?
- Right Panel (Results):**
 - List all Tom Hanks movies...
Cypher: `MATCH (tom:Person {name: "Tom Hanks"})-[:ACTED_IN]→(tomHanksMovies) RETURN tom, tomHanksMovies`
 - Who directed "Cloud Atlas"?
Cypher: `MATCH (cloudAtlas {title: "Cloud Atlas"})←[:DIRECTED]-(directors) RETURN directors.name`
 - Tom Hanks' co-actors...
Cypher: `MATCH (tom:Person {name:"Tom Hanks"})-[:ACTED_IN]→(m)←[:ACTED_IN]-(coActors) RETURN coActors.name`
 - How people are related to "Cloud Atlas"...
Cypher: `MATCH (people:Person)-[relatedTo]-(:Movie {title: "Cloud Atlas"}) RETURN people.name, Type(relatedTo), relatedTo`

At the bottom, there is a navigation bar with 4/8, a left arrow, a series of dots, and a right arrow.



Demonstração

- Lista todos os filmes que Tom Hanks atuou.





Demonstração

- Lista todos os diretores do filme Cloud Atlas

```
neo4j$ MATCH (cloudAtlas {title: "Cloud Atlas"})-[:DIRECTED]-(directors) RETURN directors.name
```

directors.name

1	"Tom Tykwer"
2	"Lilly Wachowski"
3	"Lana Wachowski"

Started streaming 3 records after 1 ms and completed after 3 ms.



Demonstração

- Lista todos os co-atores que atuaram nos filmes com Tom Hanks

```
neo4j$ MATCH (tom:Person {name:"Tom Hanks"})-[:ACTED_IN]→(m)←[:ACTED_IN]-(coActors) RETURN coActors.name
```

	coActors.name
1	"Ed Harris"
2	"Gary Sinise"
3	"Kevin Bacon"
4	"Bill Paxton"
5	"Parker Posey"
6	"Greg Kinnear"
7	

Started streaming 39 records after 1 ms and completed after 4 ms.



Demonstração

- Lista todos as pessoas que trabalharam no filme Cloud Atlas

```
neo4j$ MATCH (people:Person)-[relatedTo]-(:Movie {title: "Cloud Atlas"}) RETURN people.name, Type(relatedTo), relatedTo
```

The screenshot shows a Neo4j browser interface. On the left, there are three tabs: 'Table' (selected), 'Text' (disabled), and 'Code' (disabled). The main area displays a table with the following data:

people.name	Type(relatedTo)	relatedTo
"Tom Hanks"	"ACTED_IN"	{"roles":["Zachry", "Dr. Henry Goose", "Isaac Sachs", "Dermot Hoggins"]}
"Jim Broadbent"	"ACTED_IN"	{"roles":["Vyvyan Ayrs", "Captain Molyneux", "Timothy Cavendish"]}
"David Mitchell"	"WROTE"	{}
"Tom Tykwer"	"DIRECTED"	{}
"Lana Wachowski"	"DIRECTED"	{}
"Stefan Arndt"	"PRODUCED"	{}
"Jessica Thompson"	"REVIEWED"	{"summary": "An amazing journey", "rating": 95}
"Halle Berry"	"ACTED_IN"	{"roles": ["Luisa Rey", "Jocasta Ayrs", "Ovid", "Meronym"]}
"Hugo Weaving"	"ACTED_IN"	{"roles": ["Bill Smoke", "Haskell Moore", "Tadeusz Kesselring", "Nurse Noakes", "Boardman Mephi", "Old Georgie"]}
"Lilly Wachowski"	"DIRECTED"	{}



Demonstração

- Consultas baseadas em níveis de relacionamento e menor caminho

The screenshot shows a Neo4j browser interface. At the top, there is a command line with the text '\$:play movie-graph'. To the right of the command line are several small icons: a play button, a star, and some navigation symbols.

The main area of the browser has two columns. On the left, under the heading 'The Movie Graph', there is a section titled 'Solve' containing the following text:

You've heard of the classic "Six Degrees of Kevin Bacon"? That is simply a shortest path query called the "Bacon Path".

Below this text are three links: 'Variable length patterns', 'Built-in shortestPath()', and 'algorithm'.

On the right, under the heading 'Movies and actors up to 4 "hops" away from Kevin Bacon', there are two code snippets in a grey box:

```
④ MATCH (bacon:Person {name:"Kevin Bacon"})-[*1..4]-(hollywood)
RETURN DISTINCT hollywood
```

Bacon path, the shortest path of any relationships to Meg Ryan

```
④ MATCH p=shortestPath(
(bacon:Person {name:"Kevin Bacon"})-[*]-(meg:Person {name:"Meg Ryan"})
)
RETURN p
```

Note you only need to compare property values like this when first creating relationships

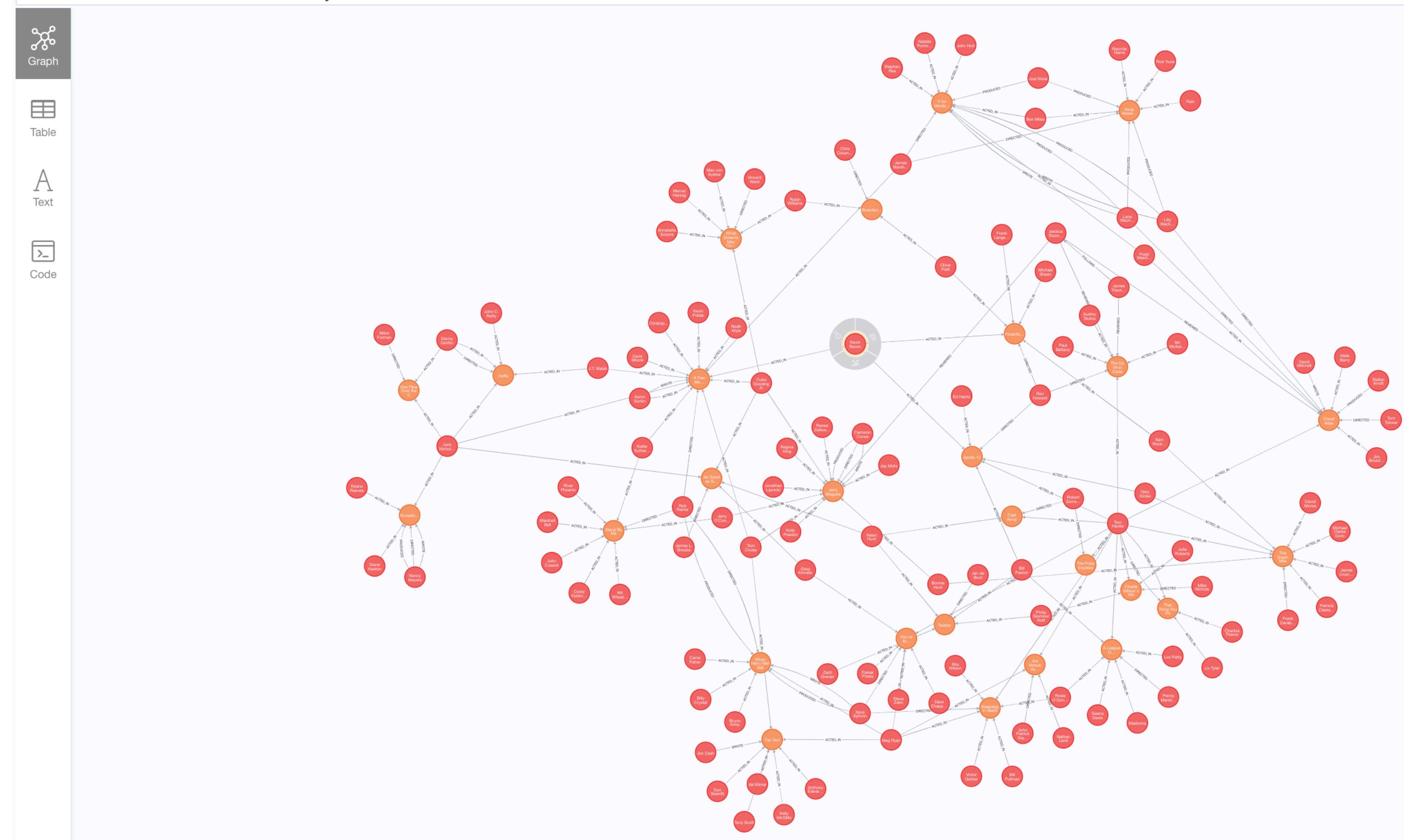
At the bottom of the browser window, there is a navigation bar with the text '5 / 8' followed by a series of small dots and arrows.



Demonstração

- Recupera todos os filmes e atores com até 4 níveis de relacionamento do ator Kevin Bacon

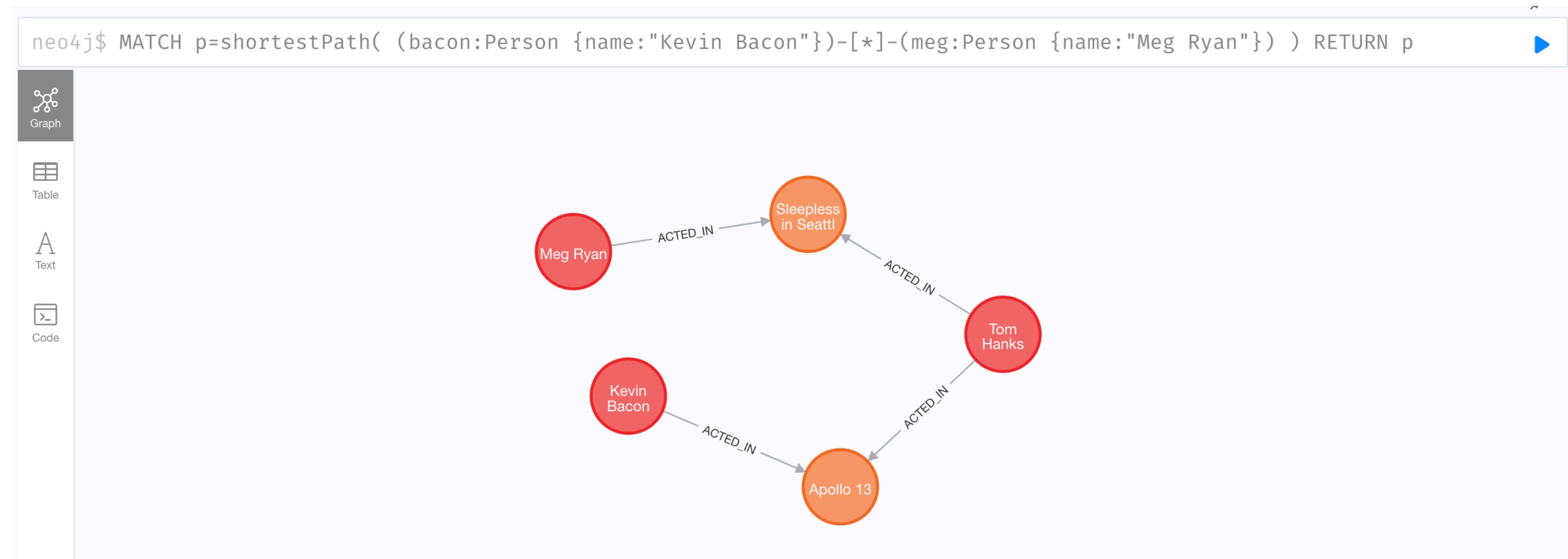
```
1 MATCH (bacon:Person {name:"Kevin Bacon"})-[*1..4]-(hollywood)
2 RETURN DISTINCT hollywood
```





Demonstração

- Recupera o caminho mais curto entre os nós Kevin Bacon e Meg Ryan.





Demonstração

- Abordagem de recomendação básica para encontrar conexões além de uma vizinhança imediata que estejam bem conectadas.

The Movie Graph

Extend Tom Hanks co-actors, to find co-co-actors who haven't worked with Tom Hanks...

```
④ MATCH (tom:Person {name:"Tom Hanks"})-[:ACTED_IN]→(m)←[:ACTED_IN]-(coActors),  
    (coActors)-[:ACTED_IN]→(m2)←[:ACTED_IN]-(cocoActors)  
    WHERE NOT (tom)-[:ACTED_IN]→()←[:ACTED_IN]-(cocoActors) AND tom ◇ cocoActors  
    RETURN cocoActors.name AS Recommended, count(*) AS Strength ORDER BY Strength DESC
```

Find someone to introduce Tom Hanks to Tom Cruise

```
④ MATCH (tom:Person {name:"Tom Hanks"})-[:ACTED_IN]→(m)←[:ACTED_IN]-(coActors),  
    (coActors)-[:ACTED_IN]→(m2)←[:ACTED_IN]-(cruise:Person {name:"Tom Cruise"})  
    RETURN tom, m, coActors, m2, cruise
```

Recommend

Let's recommend new co-actors for Tom Hanks. A basic recommendation approach is to find connections past an immediate neighborhood which are themselves well connected.

For Tom Hanks, that means:

Find actors that Tom Hanks hasn't yet worked with, but his co-actors have.
Find someone who can introduce Tom to his potential co-actor.

\$:play movie-graph

6 / 8 < • • • • • • >



Demonstração

- Lista os co-atores de Tom Hanks, para encontrar co-co-atores que não trabalharam com Tom Hanks.

```
1 MATCH (tom:Person {name:"Tom Hanks"})-[:ACTED_IN]→(m)←[:ACTED_IN]-(coActors),  
2 | (coActors)-[:ACTED_IN]→(m2)←[:ACTED_IN]-(cocoActors)  
3 WHERE NOT (tom)-[:ACTED_IN]→()←[:ACTED_IN]-(cocoActors) AND tom ◇ cocoActors  
4 RETURN cocoActors.name AS Recommended, count(*) AS Strength ORDER BY Strength DESC
```

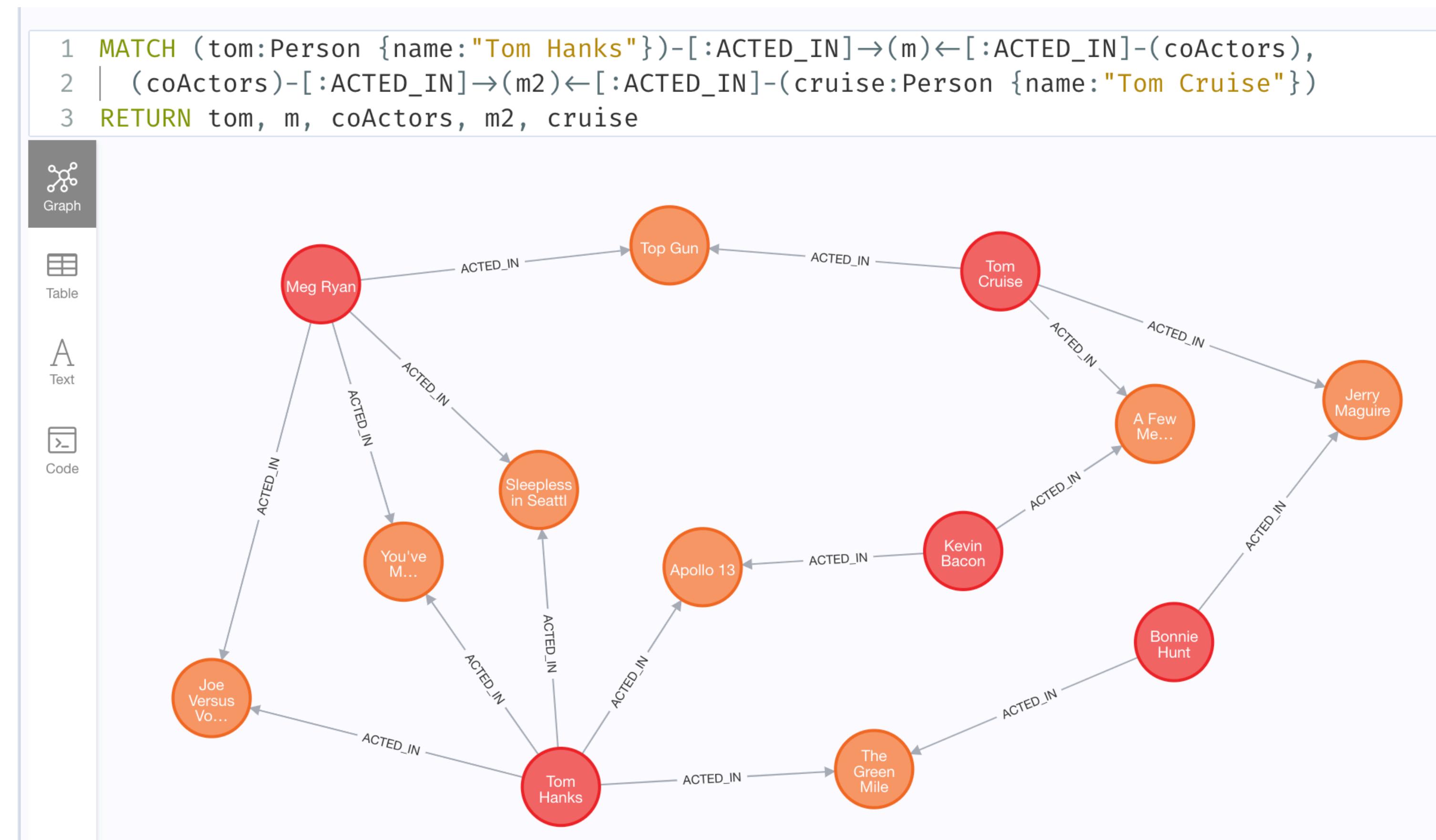
Table	Text
	"Tom Cruise" 5
	"Zach Grenier" 5
	"Cuba Gooding Jr." 4
	"Keanu Reeves" 4
	"Jack Nicholson" 3
	"Val Kilmer" 3
	"Tom Skerritt" 3
	"Kelly McGillis" 3
	"Anthony Edwards" 3
	"Carrie Fisher" 3
	"Billy Crystal" 3
	"Bruno Kirby" 3
	"Laurence Fishburne" 3
	"Carrie-Anne Moss" 3
	"Michael Sheen" 2
	"Frank Langella" 2
	"Oliver Platt" 2
	"James Marshall" 1
	"Kevin Pollak" 1
	"J.T. Walsh" 1
	"Aaron Sorkin" 1
	"Christopher Guest" 1
	"Noah Wyle" 1



Demonstração

- Encontrar alguém que possa apresentar Tom Hanks à Tom Cruise

```
1 MATCH (tom:Person {name:"Tom Hanks"})-[:ACTED_IN]→(m)←[:ACTED_IN]-(coActors),  
2 | (coActors)-[:ACTED_IN]→(m2)←[:ACTED_IN]-(cruise:Person {name:"Tom Cruise"})  
3 RETURN tom, m, coActors, m2, cruise
```





Demonstração

- Remover todos os nós e seus relacionamentos

The screenshot shows the Neo4j browser interface with the following content:

```
$ :play movie-graph
```

The Movie Graph

Clean up

When you're done experimenting, you can remove the movie data set.

Note:

< Nodes can't be deleted if relationships exist
Delete both nodes and relationships together
WARNING: This will remove all Person and Movie nodes!

>Delete all Movie and Person nodes, and their relationships

```
① MATCH (n) DETACH DELETE n
```

Prove that the Movie Graph is gone

```
② MATCH (n) RETURN n
```

:help ③ DELETE

7 / 8 < >

Demonstração

- ## • Próximos passos



Referências

- Documentação do Neo4J
 - <Https://neo4j.com/developer/>
- Documentação
 - <https://neo4j.com/developer/cypher/>
- Manual
 - <http://neo4j.com/docs/developer-manual/current/#cypher-query-lang>
- Treinamento Oficial
 - <https://neo4j.com/graphacademy/>