

Aula 15
Introdução a Bancos de Dados de Grafos
Usando o Neo4J

5 de outubro de 2016

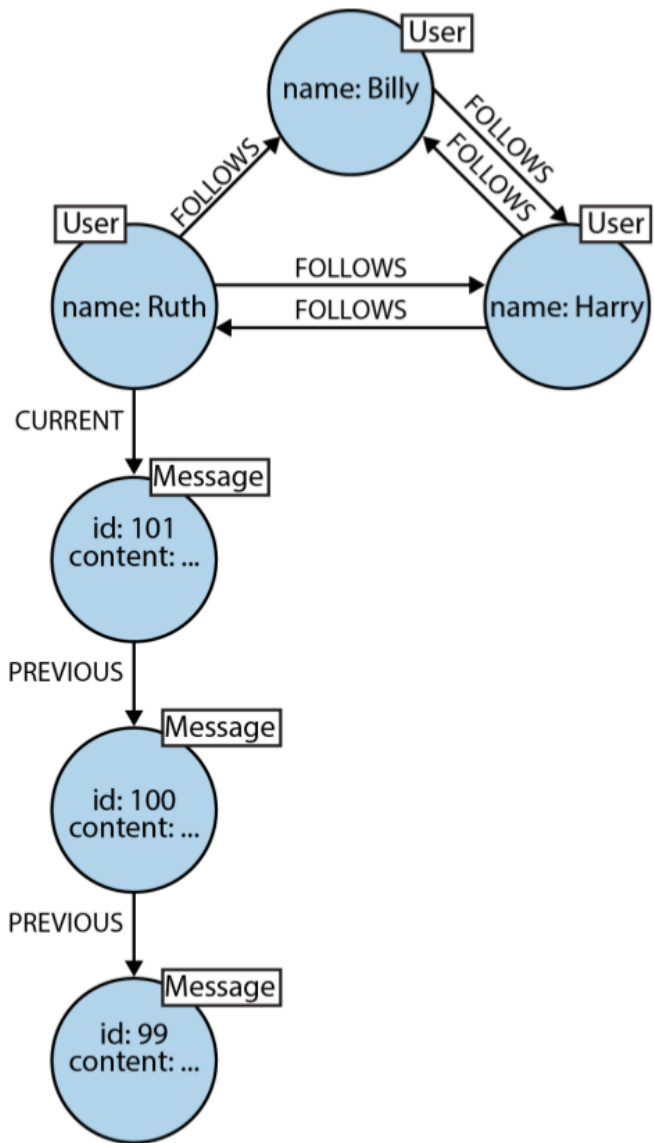
Profa. Kelly Rosa Braghetto

Banco de Dados de Grafo

- **Grafo:** coleção de nós e arestas que os conectam
- Em um BD de grafo, os nós representam entidades de um domínio e as arestas representam os relacionamentos que existem entre as entidades
- Uma das formas mais populares de modelos de grafos é o chamado de **grafo de propriedades rotulado**
 - Simples e fácil de ser entendido
 - Pode ser usado na maioria dos domínios para os quais grafos são uma boa representação para os dados

Grafo de Propriedades Rotulado

- Características:
 - Contém nós e relacionamentos (arestas)
 - Nós contêm propriedades (pares do tipo chave-valor)
 - Nós podem ser rotulados com um ou mais rótulos
 - Relacionamentos têm nome e são direcionados; sempre possuem um nó de origem e um nó de destino
 - Relacionamentos também podem conter propriedades



- Exemplo de grafo de propriedade rotulado de uma rede social

Sistema Gerenciador de Bancos de Dados de Grafos

- É um sistema de gerenciamento com suporte a operações de criação, leitura, alteração e remoção de dados, aplicadas a dados armazenados num modelo de grafo
- Esse tipo de sistema é geralmente desenvolvido para ser usado com sistemas transacionais (OLTP – online transaction processing)
 - Oferecem garantias transacionais (seguem o modelo ACID), mantendo consistência “forte” dos dados
 - São otimizados para processar grandes volumes de transações tradicionais
- Aspectos importantes desse tipo de sistema
 - Modelo de armazenamento subjacente
 - Modelo de processamento de consultas

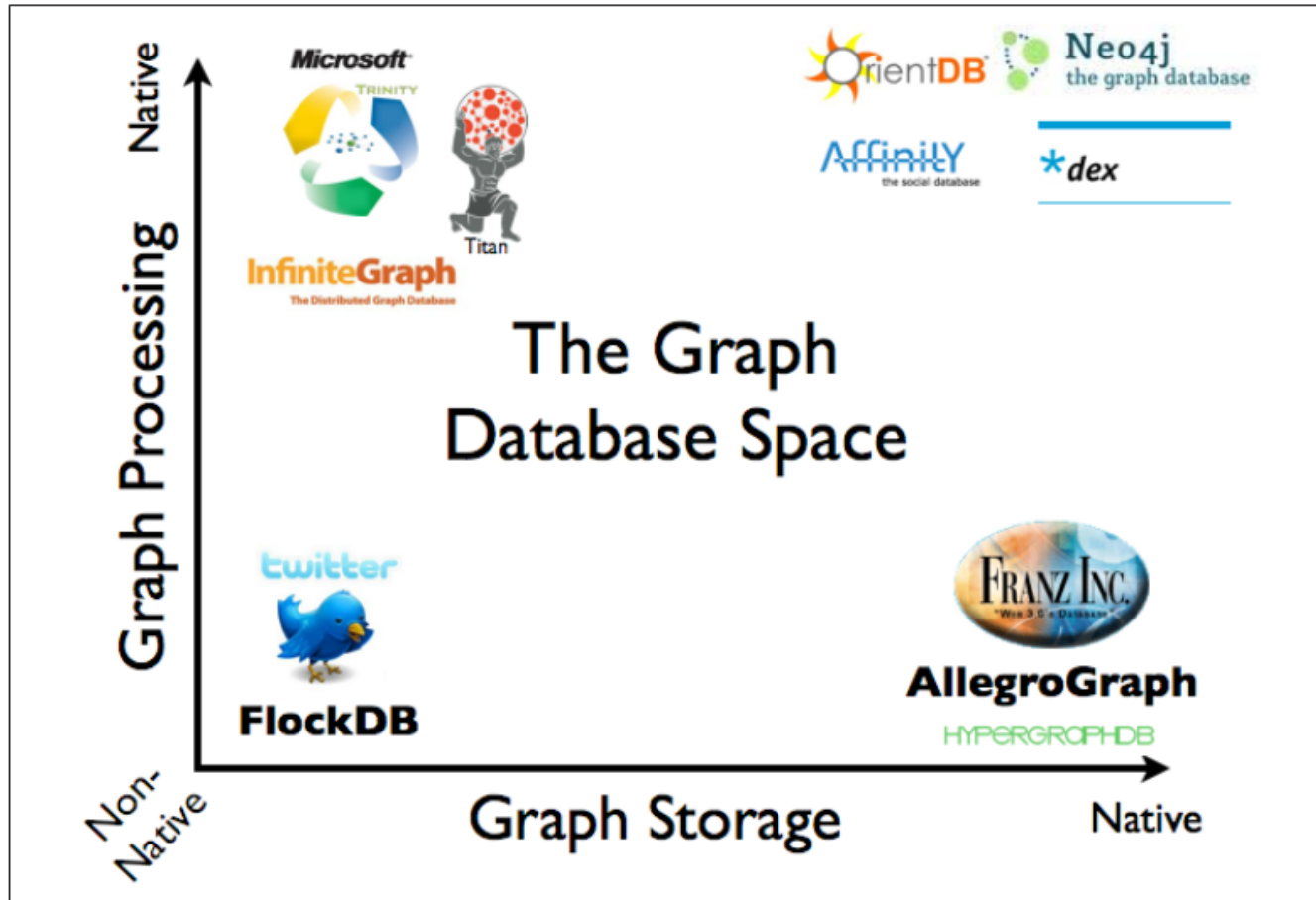
SGBD de Grafos – Modelo de de Armazenamento

- Alguns SGBDs de grafos têm **armazenamento nativo para grafos**, ou seja, um tipo de armazenamento que é projetado e otimizado para armazenar e gerenciar grafos
 - Benefícios: bom desempenho e escalabilidade
- Mas existem sistemas gerenciadores que seriam (= transformam) os grafos em outros formatos de dados, para que possam ser mantidos em outros sistemas, como em SGBDs relacionais, SGBDs orientados a objetos ou em outros sistemas de armazenamento de dados de propósito geral
 - Benefícios: usam *backends* maduros, cujas características em produção já são bastante conhecidas pelos desenvolvedores de software

SGBD de Grafos – Modelo de Processamento

- Diz-se que um SGBD faz **processamento de grafos nativo** se nele os nós conectados do grafo fisicamente apontam uns para outros banco de dados, ou seja, as adjacências dos nós não são representadas por meio de chaves estrangeiras e índices (o que é conhecido como *index-free-adjacency*).
- Mas qualquer banco de dados que, sob a perspectiva de um usuário, se comporta como um grafo (com operações de inserção, remoção, consulta, etc.) geralmente é considerado um BD de grafo
- SGBDs com **processamento de grafos nativo** têm como benefício um bom desempenho em travessias (consultas que percorrem os nós por meio de seus relacionamentos)
 - Entretanto, a execução de consultas que não envolvem travessias fica mais difícil ou mais custosa em termos de consumo de memória

Caracterização de Alguns SGBDs de Grafos



Ferramentas para Processamento de Grafos

















- São ferramentas que possibilitam a execução de algoritmos sobre grafos de grande escala
- Esse tipo de software é projetado para realizar operações tais como:
 - Identificar padrões nos dados (agrupar)
 - Reponder questões do tipo: “Quantos relacionamentos em média um pessoa tem em uma rede social?”
- Diferentemente dos SGBDs de Grafos, essas ferramentas são otimizadas para varrer e processar grandes quantidades de dados em *batch* (como ocorre em ferramentas OLAP e de mineração de dados)
- A maioria dessas ferramentas não possui uma camada de armazenamento de dados, apenas processam dados que recebem de alguma fonte externa e devolvem os resultados para serem armazenados em algum outro lugar
- Exemplos:
 - Apache Giraph: <http://giraph.apache.org/>
 - Apache Spark GraphX: <http://spark.apache.org/graphx/>



Ranking de Popularidade de Sistemas Gerenciadores de BDs de Grafos

Multi-model =
Key-Value,
Document Store,
Graph DBMS

21 systems in ranking, October 2016

| Rank | | | DBMS | Database Model | Score | | |
|----------|---|---|--|---|----------|----------|----------|
| Oct 2016 | Sep 2016 | Oct 2015 | | | Oct 2016 | Sep 2016 | Oct 2015 |
| 1. | 1. | 1. | Neo4j  | Graph DBMS | 36.45 | +0.08 | +3.04 |
| 2. | 2. |  3. | OrientDB | Multi-model  | 6.25 | -0.15 | +1.32 |
| 3. | 3. |  2. | Titan | Graph DBMS | 5.12 | +0.06 | -0.40 |
| 4. | 4. | 4. | Virtuoso | Multi-model  | 2.69 | +0.19 | -0.65 |
| 5. | 5. | 5. | ArangoDB | Multi-model  | 2.15 | +0.02 | +0.68 |
| 6. | 6. | 6. | Giraph | Graph DBMS | 0.95 | -0.17 | +0.03 |
| 7. |  8. | 7. | AllegroGraph  | Multi-model  | 0.46 | -0.05 | -0.43 |
| 8. |  7. | 8. | Stardog | Multi-model  | 0.43 | -0.17 | -0.10 |
| 9. | 9. |  10. | Sqrrl | Multi-model  | 0.26 | -0.10 | -0.17 |
| 10. |  12. |  18. | GraphDB | Multi-model  | 0.21 | +0.05 | +0.17 |

Sobre o Neo4J

- SGBD para grafos 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

A Linguagem Cypher

- Linguagem declarativa, como a SQL
 - Comandos definem o que deve ser recuperado, não como
- Muito legível e expressiva
- 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
 - Foi desenvolvida como uma proposta de linguagem de consulta padrão para BDs de grafos

Instruções de Acesso ao Neo4J

- Para se conectar ao seu BD no servidor Neo4J instalado na rede Linux, acesse a interface web disponível em:

<https://neo4j.linux.ime.usp.br:<porta>>

onde **<porta>** é o número da porta que você recebeu por e-mail (quando o seu BD foi criado no Neo4J da rede Linux)

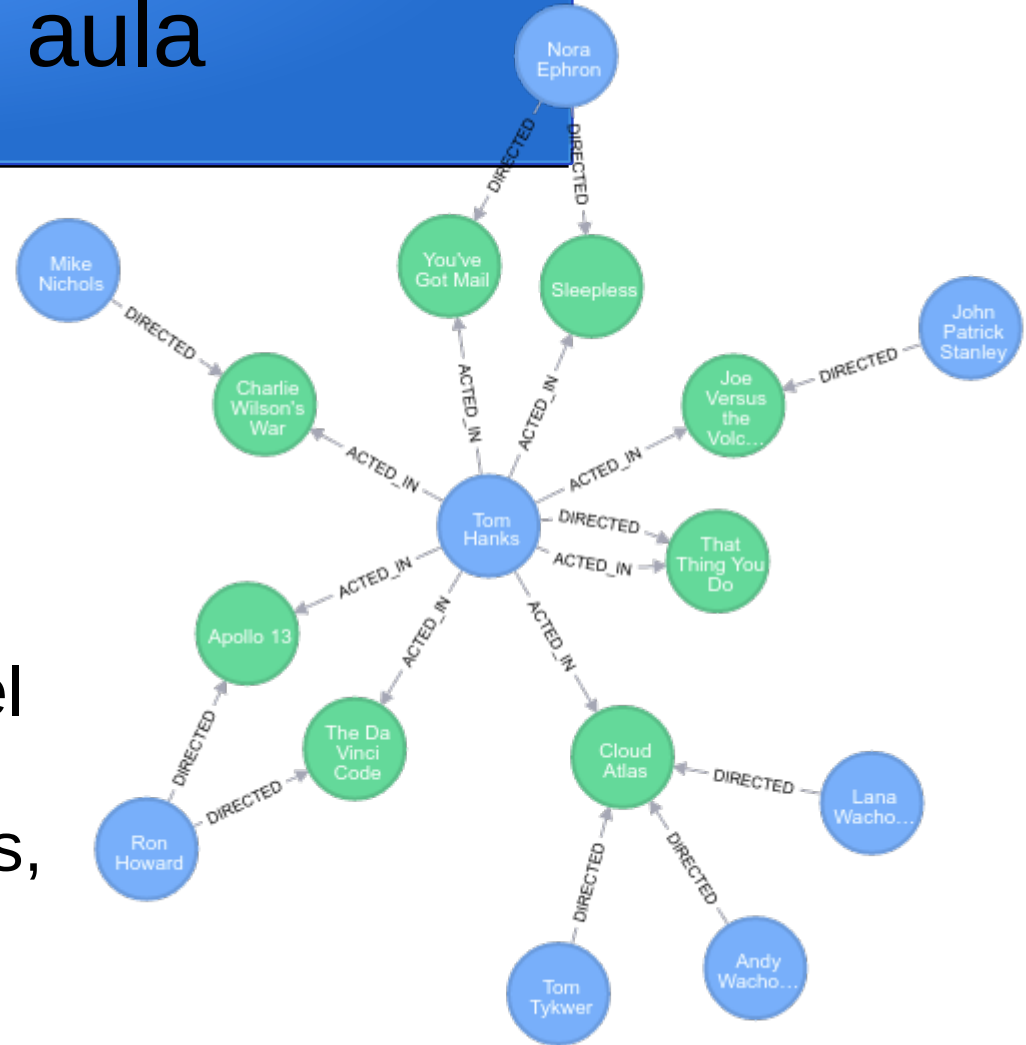
- Informe como nome de usuário para a conexão o seu login da rede Linux
- No seu primeiro acesso, será solicitado que você cadastre uma senha para o seu usuário
- Essa interface web está acessível mesmo de fora da rede do IME

BD de Exemplo para a aula

- Para criar o banco de dados que será usado nos exemplos desta aula
 - Abra o arquivo “criacao_bd_filmes.txt” (disponível no Paca)
 - Copie o conteúdo
 - Cole no prompt do cliente web de conexão com o Neo4J
 - Aberte o botão para executar os comandos colados

BD de Exemplo para a aula

- A imagem mostra parte do BD criado
- O BD contém dados sobre filmes, atores e diretores
- Na interface web, é possível interagir com os nós, consultar suas propriedades, etc.



Cypher – Nós

- São denotados por um par de parênteses
- Exemplo: um nó com nome de variável `matrix`, rótulo `Movie` e duas propriedades (`title` e `released`)

```
(matrix:Movie {title: "The Matrix", released: 1999})
```

- A variável de nó pode ser usada em outros lugares dentro do mesmo comando em que foi definida
- O rótulo 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
- Com exceção dos parênteses, todos os demais elementos de definição de nó mostrados acima são opcionais

Cypher – Outros Exemplos de Nós

```
()
```

```
(matrix)
```

```
(:Movie)
```

```
(matrix:Movie)
```

```
(matrix:Movie {title: "The Matrix"})
```

```
(matrix:Movie {title: "The Matrix", released: 1999})
```

```
(matrix:Movie:Promoted)
```

Cypher – Busca por Nós

- A consulta a seguir devolve todos os nós de filme (ou seja, os nós que têm `Movie` como rótulo)

```
MATCH (movie:Movie)
```

```
RETURN movie
```

Cypher – Busca por Nós

- A consulta a seguir devolve todos os nós de filme que têm a propriedade title = "The Matrix"

```
MATCH (movie:Movie {title:"The Matrix"})
```

```
RETURN movie
```

- Embora o uso deles na especificação de uma consulta não seja obrigatório, o Cypher usa os rótulos na otimização das consultas, portanto o uso deles é fundamental para garantir um bom desempenho nas operações sobre o grafo

Cypher – Busca por Relacionamentos

- Formato geral:

```
MATCH (node1:Label1)-[rel:REL_TYPE]->(node2:Label2)
RETURN node1, node2
```

- Exemplo: Devolva todos os personagens dos filmes no BD

```
MATCH (actor:Person)-[rel:ACTED_IN]->(movie:Movie)
RETURN rel.roles
```

Obs.: os personagens aparecem na propriedade `roles` dos relacionamentos com rótulo `ACTED_IN`.

Cypher – Busca por Padrões

- Combinando buscas por nós com buscas por relacionamentos, é possível especificar buscas por padrões
- Exemplo de padrão simples: verifique se o Keanu Reeves fez o personagem Neo no filme The Matrix

```
MATCH (matrix:Movie {title:"The Matrix" }
      <-[role:ACTED_IN {roles:["Neo"]}]-
      (keanu:Person {name:"Keanu Reeves"})
RETURN matrix, role, keanu
```

Cypher – Busca por padrões

- Busca os filmes nos quais o Keanu Reeves atuou fazendo um personagem chamado Neo:

```
MATCH (keanu:Person) - [r:ACTED_IN] -> (movie)
WHERE keanu.name="Keanu Reeves"
AND "Neo" IN r.roles
RETURN movie.title
```

Cypher – Buscas

- Para devolver todos os nós que possuem relacionamentos com outros nós:

```
MATCH (n) --> (m)
```

```
RETURN n, m;
```

- O resultado de uma busca pode ser armazenado em uma variável:

```
MATCH cast = (:Person)-[:ACTED_IN]->(:Movie)
```

```
RETURN cast
```

Cypher – Criação de Nós

- Exemplo: criação de um novo nó de pessoa

```
CREATE (me:Person {name: "Kelly R B"})  
RETURN me
```

- Consulta para verificar se o nó criado está no grafo

```
MATCH (me:Person {name:"Kelly R B"})  
RETURN me.name
```

ou

```
MATCH (me:Person)  
WHERE me.name="Kelly R B"  
RETURN me.name
```


Cypher – Criação de Relacionamentos

- Primeiro, vamos criar um novo filme:

```
CREATE (movie:Movie {title: "Mystic River",  
released:1993})
```

- Agora, vamos incluir no BD o fato do Kevin Bacon ter atuado como Sean em Mystic River:

```
MATCH (kevin:Person) WHERE kevin.name = "Kevin Bacon"  
MATCH (mystic:Movie) WHERE mystic.title = "Mystic River"  
CREATE (kevin)-[r:ACTED_IN {roles:["Sean"]}]->(mystic)  
RETURN mystic, r, kevin
```

Cypher – Criação de Relacionamentos

- Vamos incluir mais um novo relacionamento: uma avaliação para o filme `Mystic River`

```
MATCH (me:Person), (movie:Movie)
WHERE me.name="Kelly R B"
      AND movie.title="Mystic River"
CREATE (me)-[r:REVIEWED{rating:80,
      summary:"tragic character movie"}]->(movie)
RETURN me, r, movie
```

Cypher – Criação de Relacionamentos

- O comando abaixo tem resultado igual ao do slide anterior:

```
MATCH (me:Person {name:"Kelly R B"}),  
      (movie:Movie {title:"Mystic River"})  
CREATE (me)-[r:REVIEWED {rating:80,  
      summary:"tragic character movie"}]->(movie)  
RETURN me, r, movie
```

Cypher – Merge de Relacionamentos

- Se executamos um comando de criação de relacionamentos mais de uma vez, mais de uma aresta é criada entre o mesmo par de nós
- Para evitar esse problema, podemos usar o comando MERGE no lugar do CREATE. O MERGE tem semântica *get-or-create*, ou seja, primeiro ele busca pela aresta. Se a encontrar no grafo, devolve-a como resposta. Senão, ele cria a aresta.
- Exemplo:

```
MATCH (me:Person {name:"Kelly R B"}),  
      (movie:Movie {title:"Mystic River"})  
MERGE (me)-[r:REVIEWED {rating:80,  
  summary:"tragic character movie"}]->(movie)  
RETURN me, r, movie
```

Cypher – Adição de novas propriedades

- O operador **set** pode ser usado para incluir propriedades
- Exemplo: adiciona uma *tagline* ao filme `Mystic River`

```
MATCH (movie:Movie)
WHERE movie.title = "Mystic River"
SET movie.tagline = "We bury our sins here, Dave. We
wash them clean."
RETURN movie.title AS title,
        movie.tagline AS tagline
```

Cypher – Modificação de propriedades

- O operador **set** também pode ser usado para mudar o valor de propriedades que já existem
- Exemplo: muda o nome do papel do Kevin Bacon em Mystic River para Sean Devine:

```
MATCH (kevin)-[r:ACTED_IN]->(mystic)
WHERE kevin.name = "Kevin Bacon"
      AND mystic.title = "Mystic River"
SET r.roles = ["Sean Devine"]
RETURN mystic, r, kevin
```

Cypher – Remoção de Nós

- O comando a seguir remove o nó da `Kelly R B` e todos os seus relacionamentos (caso ele tenha algum)

```
MATCH (me:Person {name:"Kelly R B"})
```

```
OPTIONAL MATCH (me) - [r] - ()
```

```
DELETE me, r
```

Cypher – Remoção de Nós

- Como a operação de remover um nó juntamente com seus relacionamentos é bastante comum, foi incluída no Cypher uma variação do `DELETE`, o `DETACH DELETE`, que faz a remoção completa:

```
MATCH (me:Person {name:"Kelly R B"})
```

```
DETACH DELETE me
```


Cypher – Ordenação dos resultados

- Mostra os anos de nascimento de todas as pessoas em ordem decrescente

```
MATCH (person:Person)
RETURN person.name, person.born
ORDER BY person.born DESC
```

Cypher – Limit e Skip

- A consulta abaixo devolve a “segunda página” (cada página com 10 atores) dos dados de atores:

```
MATCH (actor:Person) -[:ACTED_IN]->(movie:Movie)
RETURN actor.name AS Actor,
        movie.title AS Movie
SKIP 10 LIMIT 10;
```

Cypher – Eliminando repetições

- Usa-se o operador **DISTINCT** para eliminar repetições na resposta de uma consulta:

```
MATCH (actor:Person) -[:ACTED_IN] -> ()  
RETURN DISTINCT actor  
ORDER BY actor.born  
LIMIT 5
```

Cypher – Buscas mais avançadas

- Encontra os nomes dos atores que atuaram em filmes em que o Tom Hanks atuou e que são mais velhos que o Tom Hanks:

```
MATCH (tom:Person)
      -[:ACTED_IN]->()-[:ACTED_IN]-
      (actor:Person)
WHERE tom.name="Tom Hanks"
AND actor.born < tom.born
RETURN distinct actor.name AS Name
```

Cypher – Buscas mais avançadas

- Encontra os nomes e a diferença de idade dos atores que atuaram em filmes em que o Tom Hanks atuou e que são mais velhos que o Tom Hanks:

```
MATCH (tom:Person {name:"Tom Hanks"})
      -[:ACTED_IN]->(movie:Movie),
      (movie)<-[:ACTED_IN]-(actor:Person)
WHERE actor.born < tom.born
RETURN DISTINCT actor.name AS Name,
               (tom.born - actor.born) AS diff
```

Cypher – Buscas mais avançadas

- Encontra os nomes dos atores que são também diretores e que atuaram em filmes em que o Gene Hackman atuou :

```
MATCH (gene:Person)
      -[:ACTED_IN]->()<-[:ACTED_IN]-
      (other:Person)
WHERE gene.name="Gene Hackman"
AND exists( (other)-[:DIRECTED]->() )
RETURN DISTINCT other
```

Cypher – Buscas mais avançadas

- Encontra os nomes dos atores que atuaram em filmes em que o Gene Hackman atuou, mas não quando o Robin Williams estava no filme :

```
MATCH (gene:Person {name:"Gene Hackman"})
      -[:ACTED_IN]->(movie:Movie),
      (other:Person)
      -[:ACTED_IN]->(movie),
      (robin:Person {name:"Robin Williams"})
WHERE NOT exists( (robin)-[:ACTED_IN]->(movie) )
RETURN DISTINCT other
```

Cypher – Busca por Padrões com caminhos de tamanho variável

- Devolve todos os pares de nós que estão conectados, mesmo que seja por meio de relacionamentos indiretos de qualquer distância:

```
MATCH (node1) -[*]- (node2)
```

```
RETURN node1, node2
```


Cypher – Busca por Padrões com caminhos de tamanho variável

Variações

- Encontra pares de nós que estão relacionados por meio de um caminho de tamanho **4**:

```
MATCH (a) - [*4] -> (b)
```

- Encontra pares de nós que estão relacionados por meio de um caminho de tamanho **entre 1 e 4**:

```
MATCH (a) - [*1..4] -> (b)
```

- Encontra pares de nós que estão relacionados (mesmo que indiretamente) por meio de relacionamentos do tipo `LIKE`

```
MATCH (a) - [:LIKE*] -> (b)
```

- Encontra pares de nós que estão relacionados por meio de relacionamentos do tipo `LIKE` em segundo grau **ou** relacionamentos diretos do tipo `KNOWS` em primeiro grau

```
MATCH (a) - [:LIKE*2 | :KNOWS] -> (b)
```

Cypher – Extrair dados de caminhos

- Exibe somente os nós dos caminhos (subgrafos) devolvidos na consulta (MATCH):

```
MATCH p = (actor:Person)
        -[:ACTED_IN]→(movie:Movie)←[:DIRECTED]-
        (director:Person)

RETURN nodes (p) ;
```

- Além da função `nodes(p)`, temos também:
 - `length (p)` – devolve o tamanho do caminho
 - `rels (p)` – devolve apenas os relacionamentos

Cypher – Agrupamento/Agregação

- Funções de agregação:
 - **count(x)**: conta o número de ocorrências
 - **min(x)**: obtém o menor valor
 - **max(x)**: obtém o maior valor
 - **avg(x)**: obtém a média de valores numéricos
 - **sum(x)**: soma valores
 - **collect(x)**: coloca todos os valores em uma collection

Cypher – Agrupamento/Agregação

- Mostra todos os títulos dos filmes dos quais um ator participou

```
MATCH (person:Person)
      -[:ACTED_IN]->(movie:Movie)
RETURN person.name, collect(movie.title);
```

- Mostra o nome de todos os diretores com os quais um ator trabalhou

```
MATCH (person:Person) -[:ACTED_IN]->
      -[:ACTED_IN]->(movie:Movie) <-[:DIRECTED]-
      (director:Person)
RETURN person.name,
       collect(DISTINCT director.name) as directors;
```

Cypher – Agrupamento/Agregação

- Devolve o número de filmes em que um ator trabalhou junto com um mesmo diretor:

```
MATCH (actor:Person)
      -[:ACTED_IN]->(movie:Movie)<-[:DIRECTED]-
      (director:Person)
RETURN actor.name, director.name, count(movie);
```

- Devolve os 10 atores que mais atuaram em filmes

```
MATCH (person:Person)-[:ACTED_IN]->(movie:Movie)
RETURN person.name, count(movie)
ORDER BY count(movie) DESC
LIMIT 10;
```

Referências Sobre o Neo4J

- Documentação do Neo4J
 - <https://neo4j.com/developer/>
 - “Treinamentos” oficiais:
 - <https://neo4j.com/graphacademy/>
 - Material da aula veio do tutorial a seguir:
<https://neo4j.com/graphacademy/online-training/introduction-graph-databases/>
 - Livros (disponíveis para download gratuito!) :
 - **“Graph Databases - The Definitive Book on Graph Databases“**
<http://graphdatabases.com/>
 - **“The Definitive Guide to Graph Databases – for the RDBMS developers“**
<https://neo4j.com/resources/rdbms-developer-graph-white-paper/>
- Obs.: compara BDs de grafos com BDs relacionais

Referências sobre a Cypher

- Documentação
 - <https://neo4j.com/developer/cypher/>
- Manual
 - <http://neo4j.com/docs/developer-manual/current/#cypher-query-lang>
- Cartão de referência
 - <http://neo4j.com/docs/cypher-refcard/current/>
-