UniRitter
Laureate International Universities®

Banco de Dados NoSQL

(NS0401)

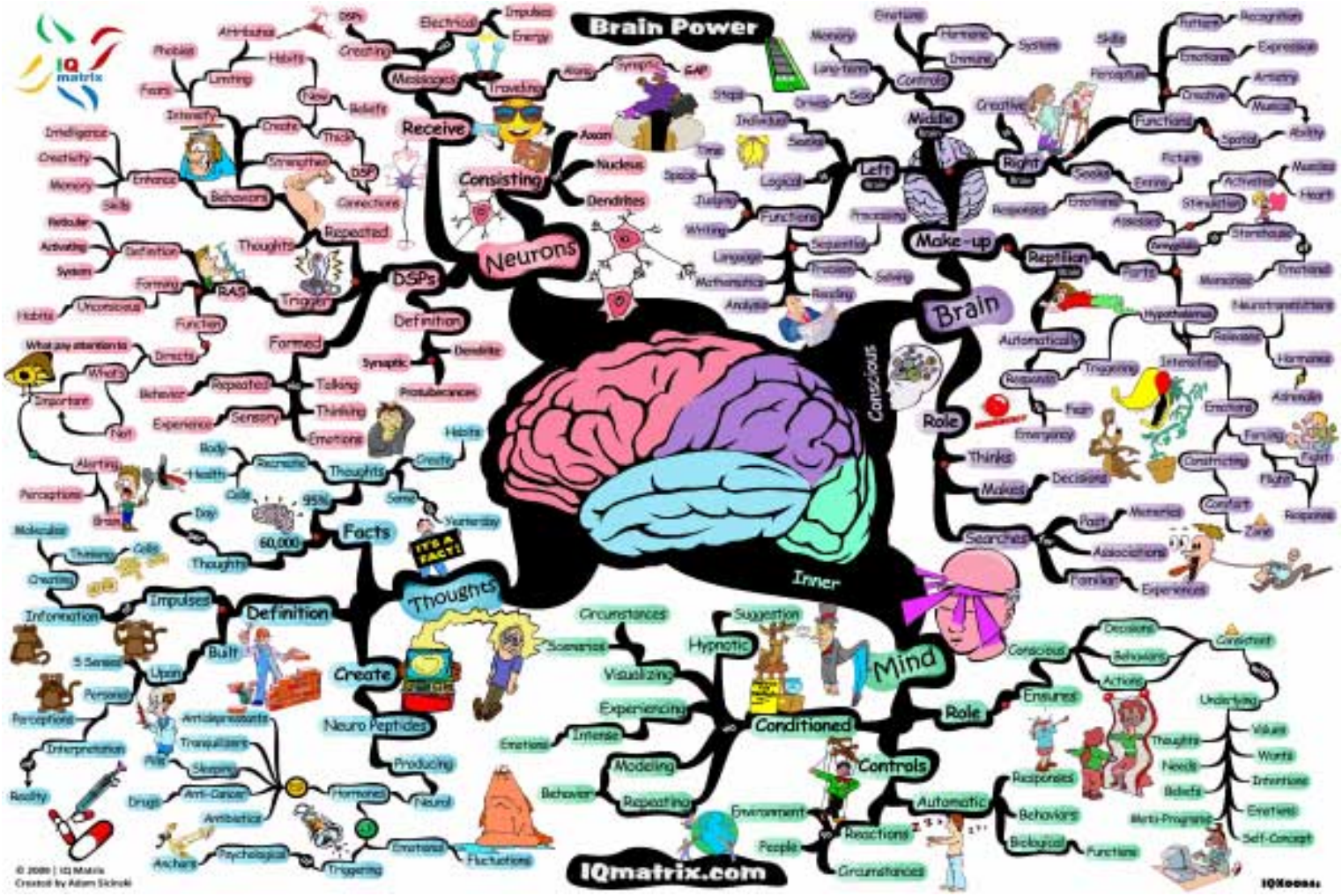
Prof. Giovane Barcelos
giovane_barcelos@uniritter.edu.br

O que vamos aprender?

- Importância e características chaves do OrientDB
- Instalação
- Pesquisas (queries) com Cypher
- CRUD básico
- Agregação / Map Reduce
- Índices (Index)
- Laboratório Prático
- Técnicas de Modelagem

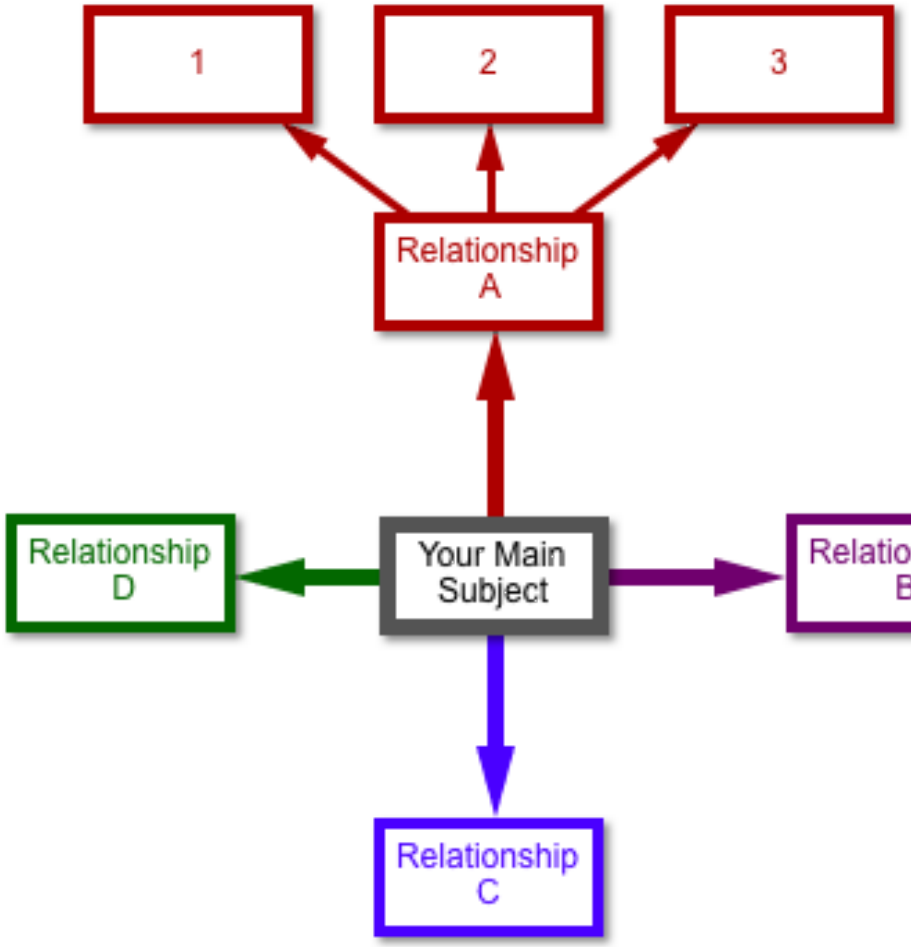
Banco de Dados em Grafo
Afinal de contas, porque precisamos de Grafos?

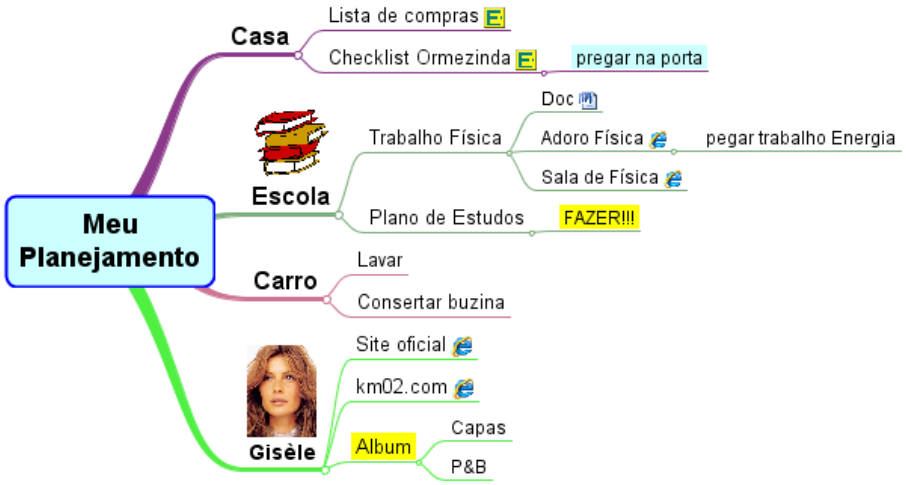
- Porque nosso **cérebro** é um grande **Grafo**

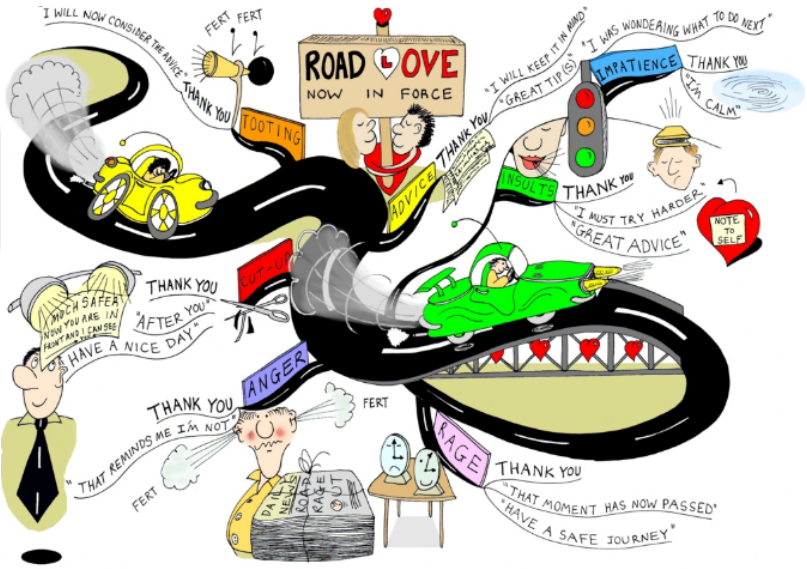


Banco de Dados em Grafo
Afinal de contas, porque precisamos de Grafos?

- **Porque aprendemos e somos seres de relacionamentos**

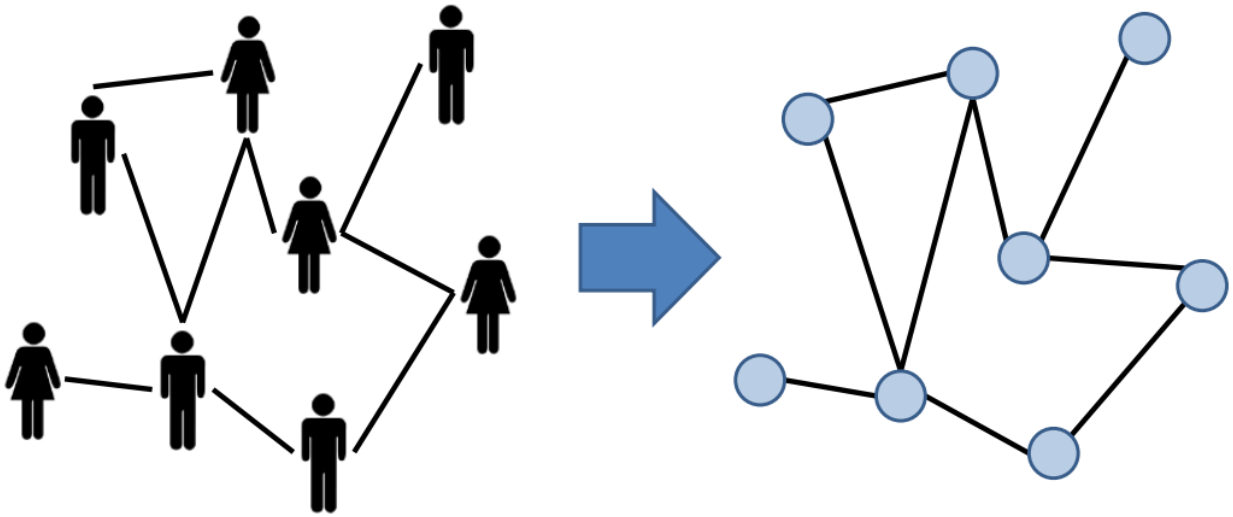






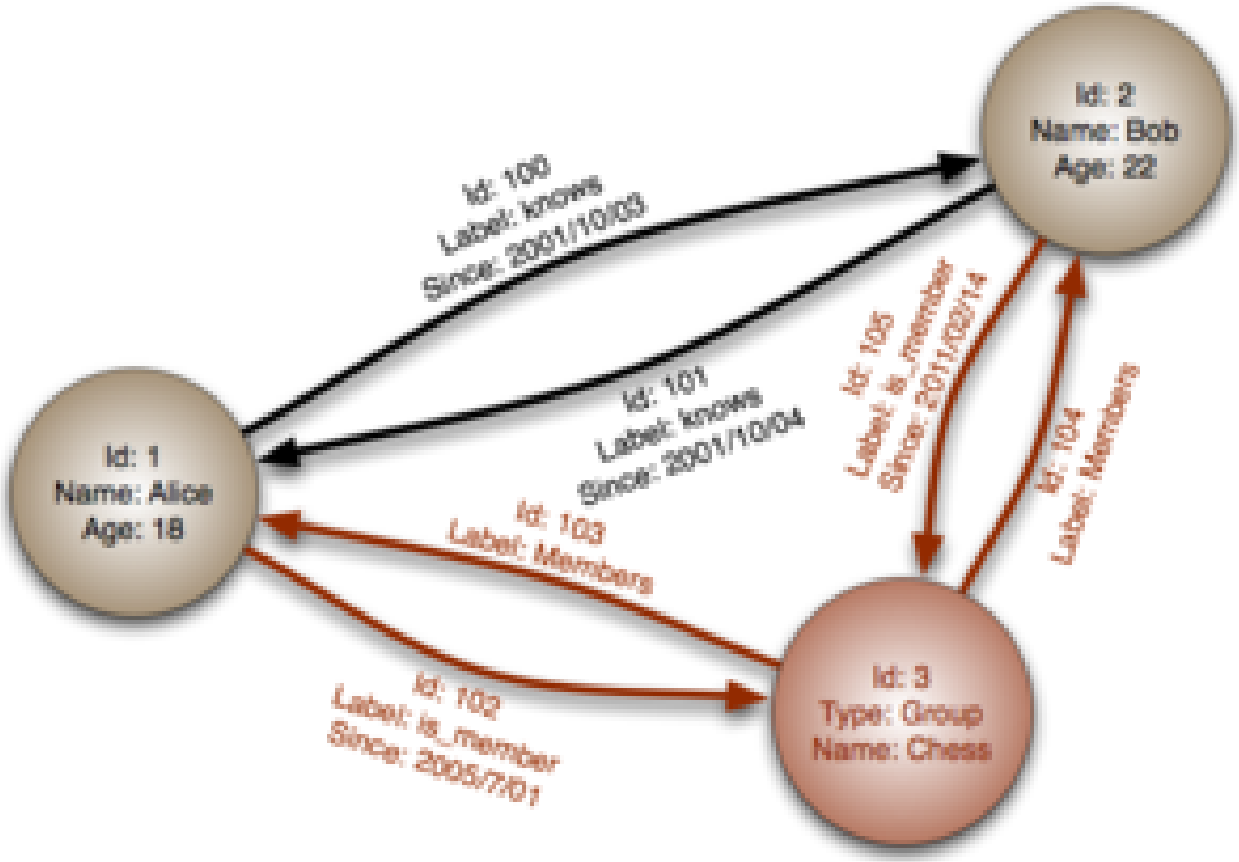
Banco de Dados em Grafo
Perguntas típicas para Grafos?

- Qual a melhor rota da minha casa até a faculdade?
- Duas pessoas tem amigos em comum?
- Quais são as rotas alternativas entre um ponto A e B no mapa?
- Quais os clientes que mais compram e que tem fornecedores que também são nossos clientes e cujos clientes destes fornecedores possuem funcionários que um dia compraram nossos produtos?



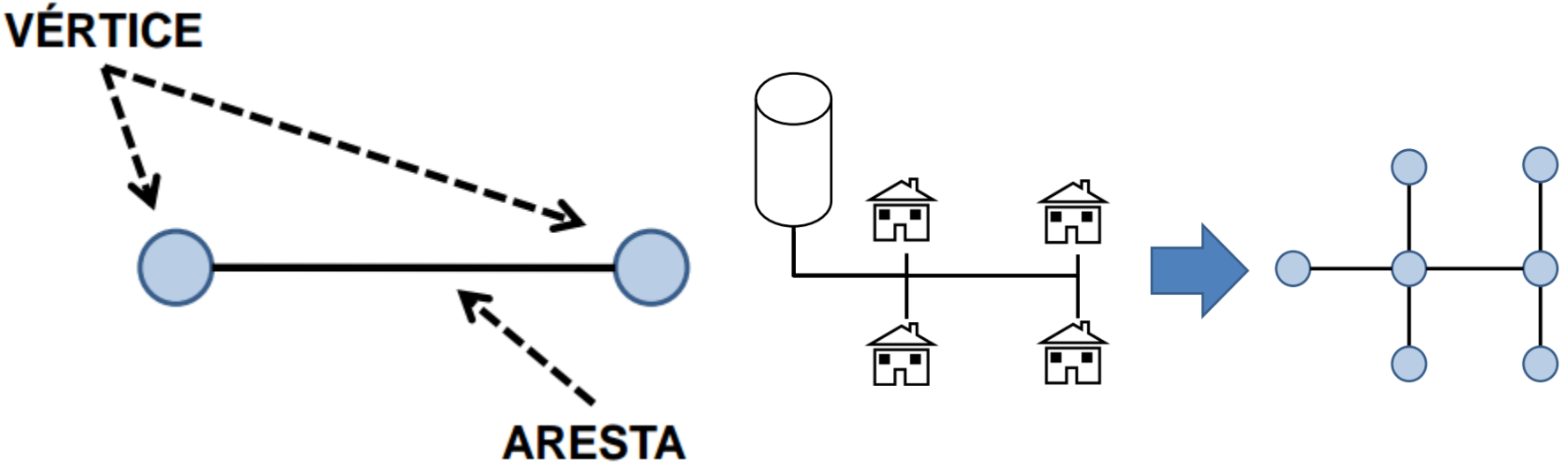
Banco de Dados em Grafo
O que são os bancos de dados em grafo?

- Banco de dados que usam **estruturas em grafos** para **pesquisas semânticas** com **vértices** (**nodos**), **arestas** e **propriedades** para armazenar os dados



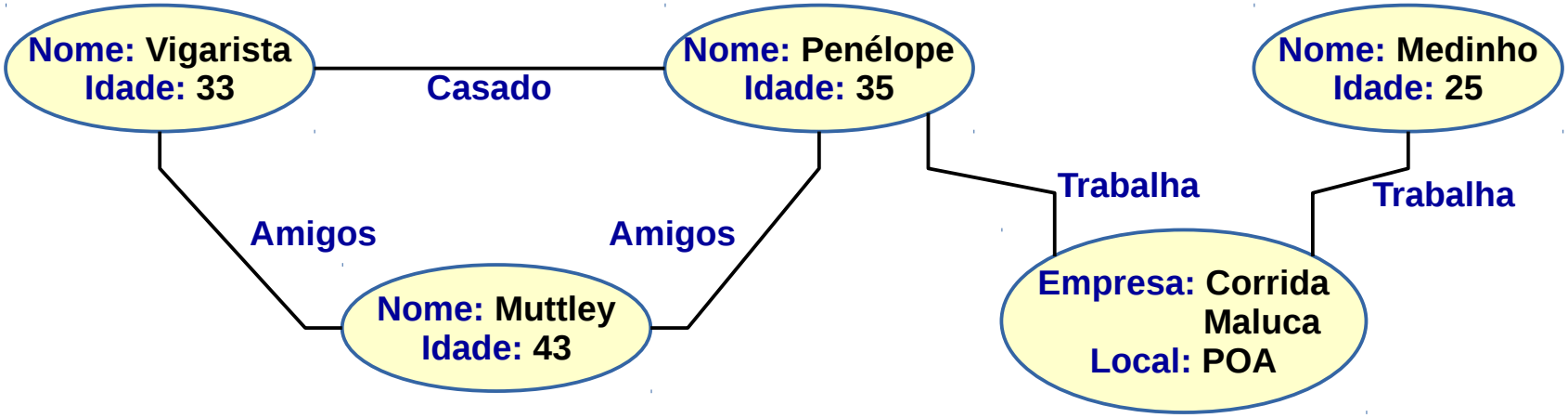
Banco de Dados em Grafo
Quais os elementos dos grafos?

- **Vértices ou Nodos**: representam entidades ou itens (Usuários, Clientes, Produtos, etc)
- **Arestas**: linhas que conectam os nodos com outros nodos ou nodos para propriedades
- **Propriedades**: informação pertinente ou relacionada ao nodo



Banco de Dados em Grafo
Exemplo de Grafo e Bancos em Grafo

Exemplo Grafo:



Alguns Bancos:

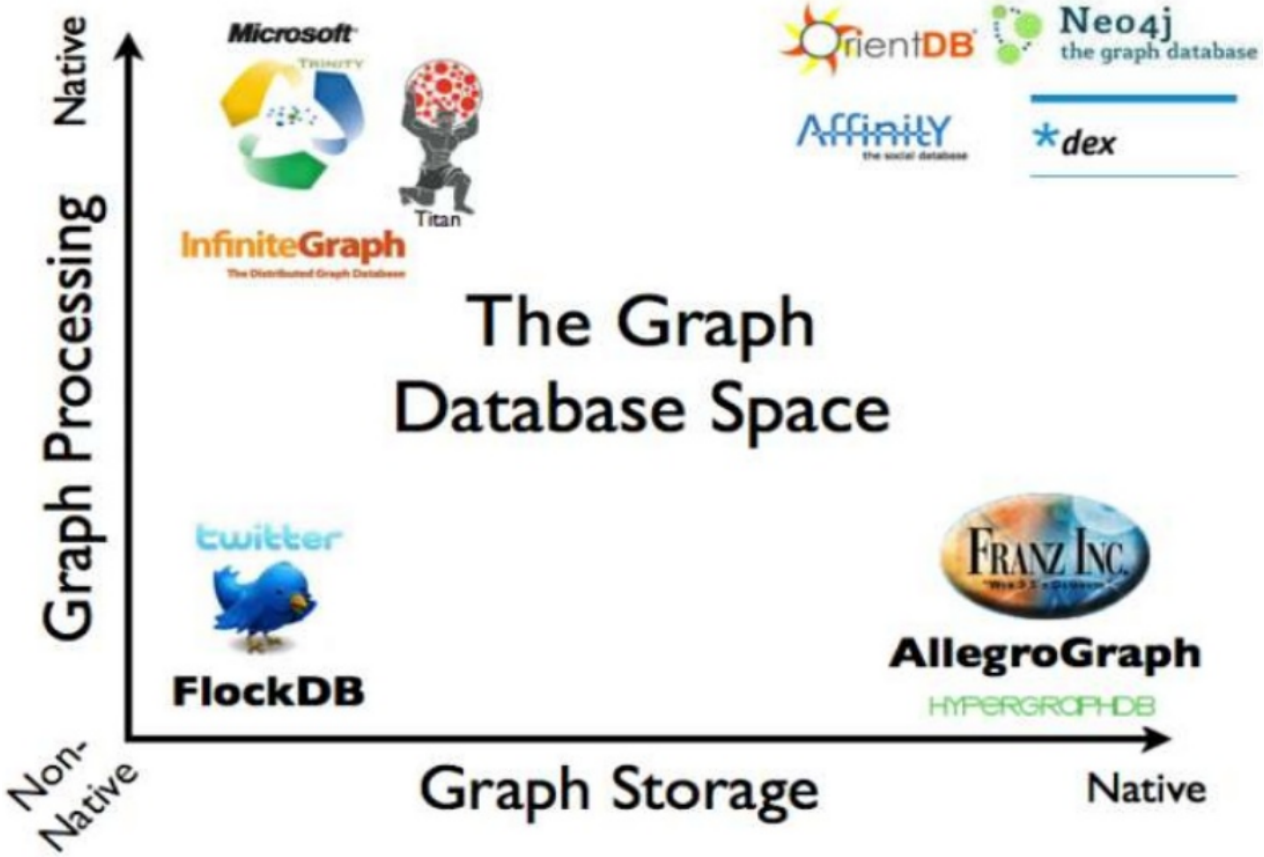
- Neo4j
- ArangoDB
- DGraph
- OrientDB
- MarkLogic
- Titan
- Filament
- GraphBase
- Cayley
- InfoGrid
- Bitsy

Banco de Dados em Grafo
Quais as Vantagens?

- Bancos de dados em Grafo **geralmente são mais rápidos** (*http://bit.do/GraphFaster*) que bancos de dados relacionais por **associarem conjunto** de dados **diretamente**
- Fazem **mapeamento direto de objetos** para a **aplicação**
- Manipulam **grandes quantidades de dados** de forma muito mais eficiente que ***RDBMS***
- **Fácil para escalar**
- **Performance**, **flexibilidade** e **agilidade**

Banco de Dados em Grafo
Quais são os mecanismos de armazenamento?

- Grande **variedade** de **formas** nos produtos
- Alguns usam **armazenamento de grafo nativo**
- Outros são **grafos em tabelas**
- Formato em grafo é similar a "**database index**"



Banco de Dados em Grafo
Onde Usar Banco em Grafo?

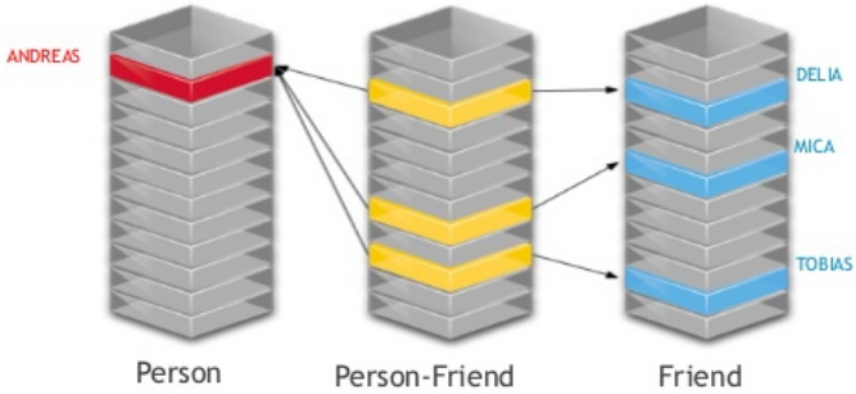
- Redes e Operações de TI
- Aplicações baseadas em transações (Ex: Bancos)
- Detecção de Fraudes
- Redes Sociais
- Pesquisas baseadas em grafos
- Quem Usa?

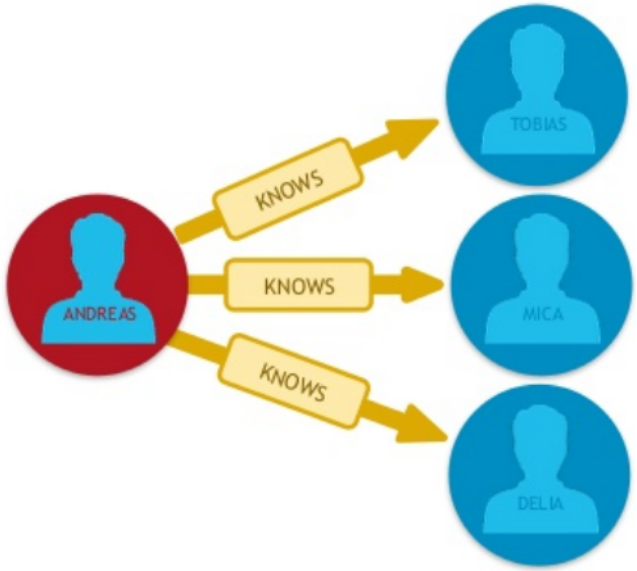
Banco de Dados em Grafo
Quando Usar Banco em Grafo?

RDBMS	*Banco em Grafo*
Complexidade Moderada	Alta Complexidade
Centenas de Relacionamentos	Bilhões de Relacionamentos
Operações JOIN moderadas	"JOINs" ao extremo
Esquema com poucas alterações	Esquema com muitas alterações
Poucas mudanças dos dados	Muitas mudanças dos dados
Dados estruturados	Dado estruturado/sem estrutura
Transações complexas	Transações simples
Forte consistência	Consistência ajustável
Inserção de dados moderada	Inserção de dados muito rápida
Alta disponibilidade (*failover*)	Disponibilidade continua (no down)
Aplicação centralizada (padrão)	Aplicação Descentralizada
Escalabilidade Vertical	Escalabilidade Horizontal

Banco de Dados em Grafo
Onde Usar Banco em Grafo?

RDBMS	*Banco em Grafo*
Banco de Dados	Banco de Dados
Tabela	Vértice (Nodo)
Linha	Documento de Chaves/Valores ou *JSON*
Coluna	Propriedade
Índice	Índice
Junção (Join)	Caminhar (*Traversing*) pelas Arestas (*Links*)
Chave Estrangeira	Aresta
Particionamento	*Cache Sharding*





Banco de Dados em Grafo
Exemplo SQL vs Cypher (*Neo4j*)

Definição SQL:

```
SELECT p.nome
  FROM Pessoa p LEFT JOIN Pessoa_Departamento pd
                    ON p.pessoa_id = pd.pessoa_id
                  LEFT JOIN Departamento d
                    ON d.departamento_id = pd.departamento_id
 WHERE d.nome = 'Departamento TI'
```

Definição Cypher:

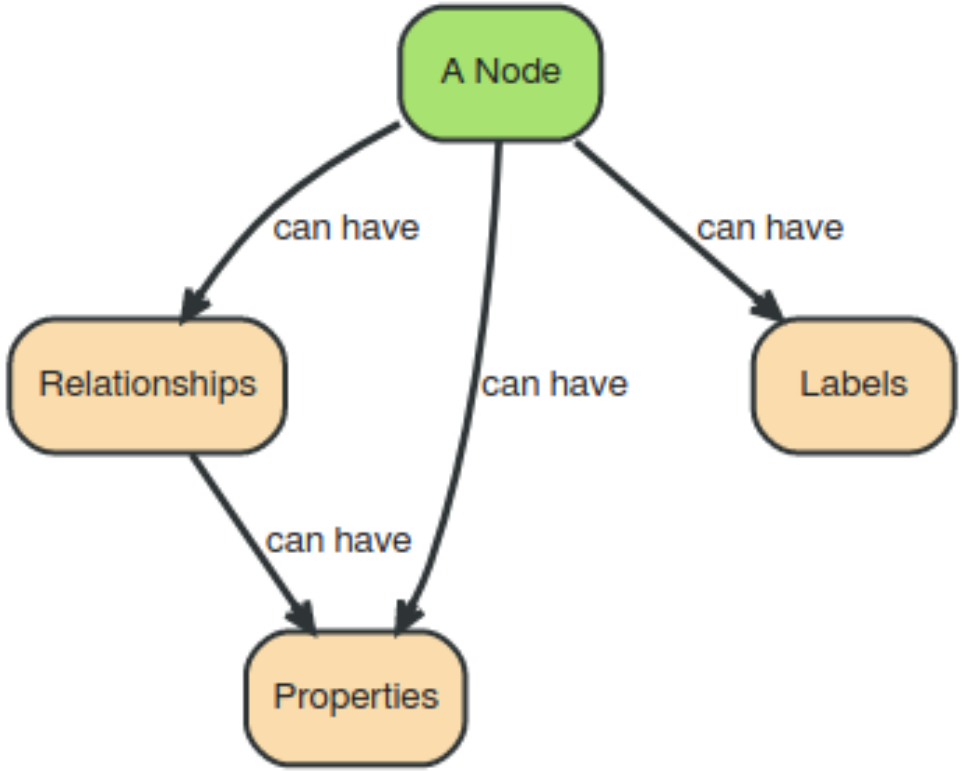
```
MATCH (p:Pessoa)<-[:Empregado]-(d:Departamento)
WHERE d.nome = 'Departamento TI'
RETURN p.nome
```

Banco de Dados em Grafo
Modelagem em Grafo é mais Fácil

- **Modelar Banco de Dados significa extrair Substantivos e Verbos de um texto ou diálogo, onde:**
  - ✔ **Substantivos: são Nodos/Vértices (Tabelas) ou Propriedades dos Nodos**
    - **Nodos/Vértices: quando são dependentes, ou seja, precisam de ao menos uma propriedade interna para fazerem sentido e tornarem-se aceitos como objeto de banco**
    - **Propriedades: quando independentes, mas para terem identidade precisam estar dentro de um Nodo/Vértice**
  - ✔ **Verbos: são Arestas (Chaves Estrangeiras) dos Nodos/Vértices (Tabelas)**

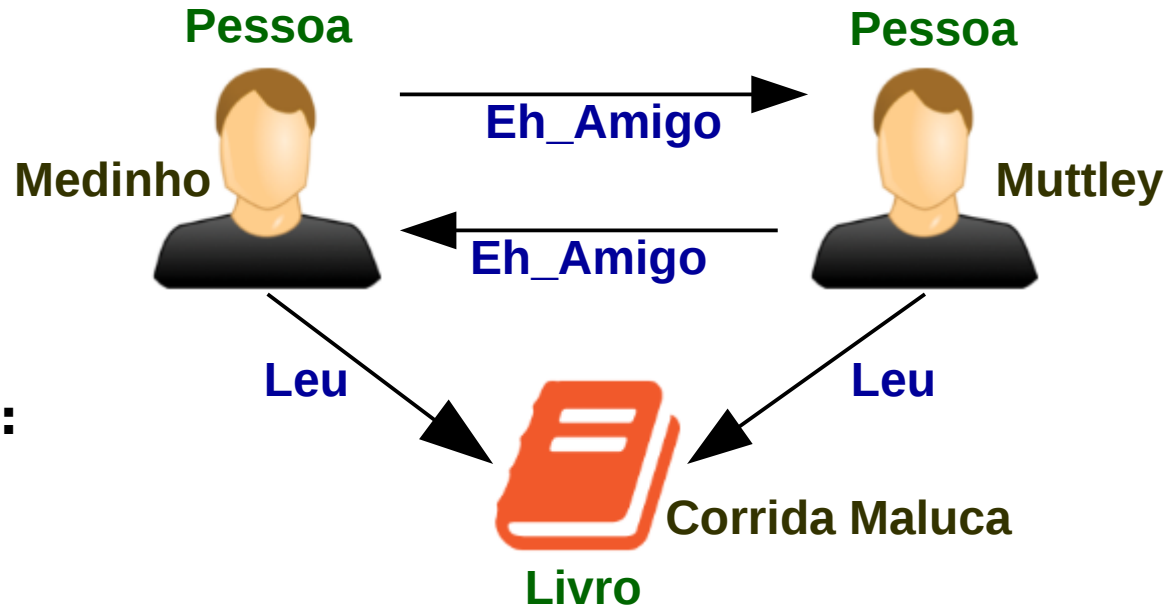
Banco de Dados em Grafo
Elementos de modelagem

- **Nodos (*Nodes*)**: entidades conectadas que podem possuir qualquer número de propriedades/atributos
- **Propriedades (*Properties*)**: pares chave/valor {nome: 'Penélope', idade: 34} ou *JSON*
- **Etiquetas (*Label*)**: os nós podem ser marcados com Rótulos, que são os agrupadores de tipos
- **Relacionamentos (*Relationships*)**: linhas para conectar nós que podem manter propriedades



Banco de Dados em Grafo
Exemplo de Texto para Modelagem

- **Medinho e Muttley são amigos. Medinho e Muttley leram o livro "Corrida Maluca"**



- **Substantivo (Nodo Rótulo):**
  - ✔ **Pessoa**
  - ✔ **Livro**
- **Substantivo (Propriedade Nome):**
  - ✔ **Medinho**
  - ✔ **Muttley**
  - ✔ **Corrida Maluca**
- **Verbo (Relacionamento):**
  - ✔ **Medinho e Muttley SÃO Amigos: Medinho é Amigo de Muttley. Muttley é Amigo de Medinho**
  - ✔ **Medinho e Muttley Leram o Livro Corrida Maluca: Medinho Leu o Livro Corrida Maluca Muttley Leu o Livro Corrida Maluca**

Neo4j
Quem é o *Neo4j*?

- **Neo4j é um Banco de Dados em Grafo Nativo altamente escalável e criado para aproveitar não apenas dados, mas também seus relacionamentos**
- **Armazenamento e processamento de Grafo nativos**
- **Usa a linguagem de pesquisa *Cypher***
- ***Full ACID* (Atomicidade, Consistência, Isolamento e Durabilidade)**
- **Implementa *MVCC* (Controle de Concorrência Multiversão)**

Neo4j
Instalação e Uso

- ***Wizards Windows***:
  http://bit.do/Neo4JWin64 e http://bit.do/Neo4JWin32
- ***Linux Debian***: http://debian.neo4j.org/

- **Senha sudo/root na VM xubuntu: xubuntu**
- ***Docker***:

```
sudo docker run \
        --publish=7474:7474 --publish=7687:7687 \
        --volume=$HOME/neo4j/data:/data \
            --env=NEO4J_AUTH=none \
              -dit  --restart=always \
            neo4j
```

- **Usuário / Senha: neo4j / neo4j**
- **Gerenciamento: http://localhost:7474/**

Neo4j
Características

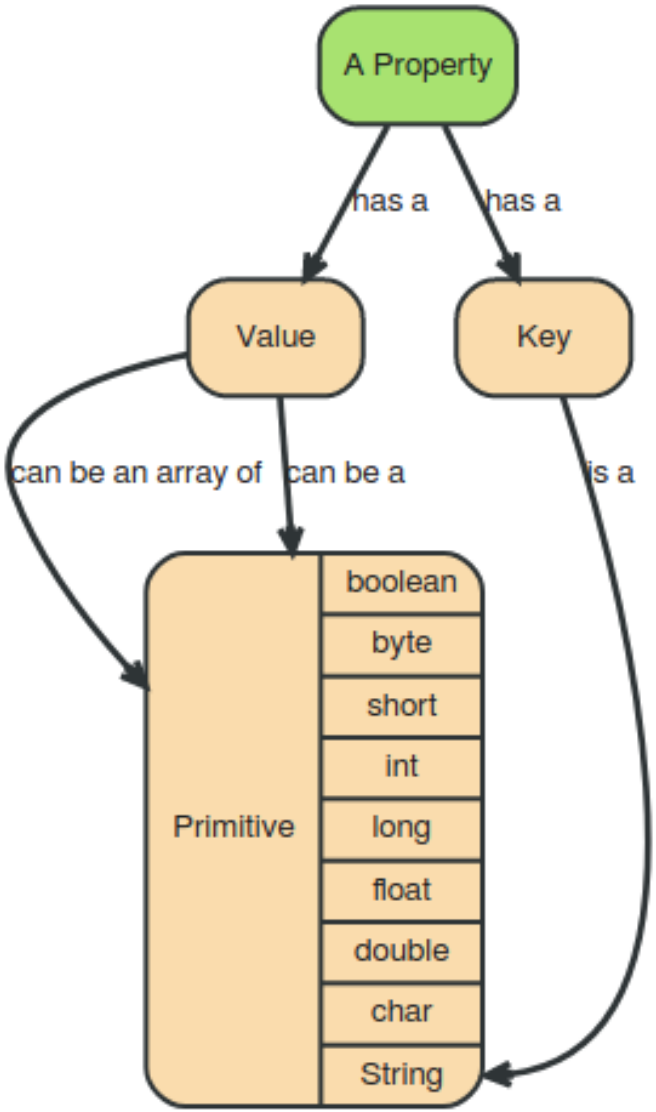
- Escalabilidade por ***Sharding***
- **Edições**: ***Community***, ***Enterprise*** e ***Government***
- Possui interface **web**: ***http://localhost:7474***
- Pode ser acessado por ***API*** ou ***Driver***
- ***Drivers***: ***.NET***, ***Java***, ***JavaScript***, ***Python***, ***Go***, ***etc***
- Integra com ***MongoDB***, ***Spark***, ***Cassandra*** e ***Elastic***
- **Não tem limites** para tamanho de grafos, espaço em disco, tamanho dos índices e escalabilidade
- **Esquema e restrições opcionais**
- Gatilhos **(Triggers)** por **eventos de transação**
- Possui **autorização** e **autenticação** por usuário e perfis: **http://bit.do/Neo4JAuth**
- **Backup** Full, Incremental e Online: **http://bit.do/Neo4JBackup**

Neo4j
O que é o *Cypher*?

- Linguagem **declarativa** de **consulta** de **Grafos** que utiliza **padrões** de **correspondência**
- **Eficiente** em **pesquisa** e **atualização** de Grafos
- **Simples**, mas **poderoso**
- Consultas complexas podem ser expressas com facilidade
- É o ***SQL*** dos **Grafos**!

```
MATCH (eu:Pessoa)-[:CONHECE]→(tu:Amigo)
```

Neo4j
Quais os tipos das propriedades?



Tipo	bit	Limite
boolean		*true / false*
byte	8	-128 até 127
short	16	-32768 até 32767
int	32	-2147483648 até 2147483647
long	64	-9223372036854775808 até 9223372036854775807
float	32	
double	64	
char	16	u0000 to uffff (0 to 65535)
String	16+	

Neo4j
Onde o Neo4J é mais utilizado?

- ***Matchmaking***
- **Gerenciamento de rede**
- **Análise de dados**
- **Pesquisa científica**
- **Gerenciamento de projetos**
- **Recomendações**
- **Redes Sociais**

Cypher
Quais as cláusulas do Cypher?

- ***MATCH***: usado para **combinar**, **encontrar** e **selecionar** padrões
- ***WHERE***: adiciona **restrições** opcionais
- ***CREATE***: **cria** nodos e relacionamentos
- ***DELETE***: **remove** nodos e relacionamentos
- ***SET & REMOVE***: **atribui** e **remove** valores de propriedades e adiciona rótulos aos nodos
- ***RETURN***: retorna nodos, propriedades e dados. *SET, REMOVE, CREATE E DELETE* não precisam retornar
- ***MERGE***: **combina** (*match*) ou **cria** novos nodos e padrões
- ***WITH***: permite que as **partes de consulta** sejam **encadeadas**, canalizando os **resultados** de **um** para serem **usados** como **pontos de início** ou **critérios** no **próximo**

Cypher
Regras Gerais

- Deve-se usar parênteses () para descrever NODO

  MATCH (pessoa) RETURN pessoa
- Rótulos agrupam Nodos e servem para distinguir Tipos de nodos. Ex: "Pessoa" e "Local"
- Pode-se criar alias para identificar rótulos, tais como, n, a, b

  MATCH (nodo:Rotulo) RETURN nodo

  MATCH (nodo:Tipo) RETURN nodo

  MATCH (p:Pessoa) RETURN p
- Relacionamentos são representados por --> e usam [:] para identificar o tipo de relacionamento

  MATCH (a)-->(b)

  MATCH (a)-->()

  MATCH (a)-[:TIPO_RELACIONAMENTO]->(b)

  MATCH (a:Pessoa)-[:Conhece]->(b:Pessoa) RETURN a, b

Cypher
Direções

- Apenas uma direção é suportada ao usar CREATE

```
CREATE (a:Pessoa {nome: 'Dick Vigarista'})
          -[r:AMIGOS]->(b:Pessoa {nome:'Muttley'})
```

- Três modos são suportados ao usar MATCH

```
MATCH (a:Pessoa)-[:AMIGOS]->
    (b:Pessoa) RETURN a,b // Esquerda para Direita
MATCH (a:Pessoa)<-[:AMIGOS]-
    (b:Pessoa) RETURN a,b // Direita para Esquerda
MATCH (a:Pessoa)-[:AMIGOS]-
    (b:Pessoa) RETURN a,b  // Ambas Direções
```

Cypher
Esquema Opcional e *Constraint*

- **Propriedade Única (PK)**

```
CREATE CONSTRAINT ON (livro:Livro)
        ASSERT livro.isbn IS UNIQUE
DROP CONSTRAINT ON (livro:Livro)
        ASSERT livro.isbn IS UNIQUE
```

- **Propriedade obrigatória no Nodo:**

```
CREATE CONSTRAINT ON (livro:Livro)
                ASSERT exists(livro.isbn)
```

- **Propriedade obrigatória no Relacionamento:**

```
CREATE CONSTRAINT ON ()-[like:LIKED]-()
                ASSERT exists(like.day)
```

Cypher
Propriedades

- Em vez de retornar todo o Nodo em si, pode-se devolver qualquer uma das propriedades

```
MATCH (a) RETURN a.propriedade
MATCH (a:Rotulo {propriedade1: ”valor”,
                 propriedade2: ”valor”})
      RETURN a
MATCH (a:Rotulo {propriedade1: “valor”,
                 propriedade2: ”valor”})
      RETURN a.propriedade1

MATCH (a:Pessoa) RETURN a.nome
MATCH (a:Pessoa {nome: ‘Foo’}) RETURN a
MATCH (a:Pessoa {nome: ‘Foo’}) RETURN a.nome
```

Cypher
Padrões dos Nodos

- (n:Pessoa) → Nodo com Rótulo Pessoa
- (n:Pessoa:Membro) → Nodo com Rótulos Pessoa e Membro
- (n:Pessoa {nome:'Penélope'}) → Nodo com Propriedade

Cypher
Padrões dos Relacionamentos

- (a)-->(b) → Relacionamento de a para b
- (a)--(b) → Relacionamento em qualquer direção entre a e b
- (a:Pessoa)-->(b) → Nodo com Rótulo Pessoa com Relacionamento para b
- (a)-[:Conhece]->(b) → Relacionamento do tipo 'Conhece' do a para o b
- (a)-[:Conhece|:Ama]->(b) → Relacionamento do tipo 'Conhece' ou 'Ama' de a para b
- (a)-[r]->(b) → Vincula a relação com a variável r
- (a)-[:Conhece]->(b {propriedade:valor}) → Relacionamento do tipo 'Conhece' com a propriedade declarada

Cypher
Padrões de Comprimento Variável

- **(a)-[*1..5]->(b)** → **Caminho de comprimento variável de entre 1 e 5 relações de a para b**
- **(a)-[*]->(b)** → **Comprimento variável de qualquer número ou relacionamento de a para b**
- **size((a)-->()-->())** → **Contar caminhos que combinam com o padrão**

Cypher
Listas e Operador IN

- Se o Cypher sabe que algo **existe** em uma **lista**, o resultado será **verdadeiro**
- Qualquer **lista** que **contenha** um ***NULL*** e **não** tenha um **elemento** correspondente **retornará *NULL***

```
MATCH (n)
WHERE id(n) IN [0,3,5]
RETURN n
```

Expression	Result
2 IN [1, 2, 3]	TRUE
2 IN [1, NULL, 3]	NULL
2 IN [1, 2, NULL]	TRUE
2 IN [1]	FALSE
2 IN []	FALSE
NULL IN [1,2,3]	NULL
NULL IN [1,NULL,3]	NULL
NULL IN []	FALSE

Cypher
MATCH Opcional

- Funciona como *MATCH*, exceto se nenhuma correspondência for encontrada, *OPTIONAL MATCH* usará NULL para partes faltantes do padrão
- Semelhante ao Outer Join do SQL

```
MATCH (a:Filme { titulo: 'Algum Filme' })
OPTIONAL MATCH (a)-->(x)
RETURN x
```

Cypher
Álias

- Pode-se renomear colunas e criar alias usando 'AS'

```
MATCH (a {nome: 'Muttley'})
RETURN a.data_nascimento AS DataNascimento
```

Cypher
Order By

- **Order By é usado para ordenar uma ou mais propriedades de retorno**

```
MATCH (n)
RETURN n
ORDER BY n.ultimo_nome DESC
```

```
MATCH (n)
RETURN n
ORDER BY n.ultimo_nome, n.primeiro_nome DESC
```

Cypher
Limitar e Saltar Resultados

- Comando **LIMIT** para limitar os resultados:

```
MATCH (n)
RETURN n
ORDER BY n.nome
LIMIT 5
```

- Comando **SKIP** para saltar resultados:

```
MATCH (n)
RETURN n
ORDER BY n.nome
SKIP 3
// Este exemplo inicia a partir do quarto nodo
```

Cypher
WITH

- **Permite que as consultas sejam encadeadas, encaminhando os resultados de um nodo, para ser usado como ponto de partida ou critérios, no próximo**

```
MATCH (joao { nome: "Joao" })
          --(outraPessoa)-->()
WITH outraPessoa, count(*) AS qtde_pessoas
WHERE qtde_pessoas > 1
RETURN outraPessoa
```

Cypher
UNWIND

- **Expande uma lista em uma seqüência de linhas**

```
UNWIND [1,2,3] AS x
RETURN x
```

Cypher
UNION

- Usado para combinar o resultado de várias pesquisas

```
MATCH (n:Ator)
RETURN n.nome AS nome
UNION ALL MATCH (n:Filme)
RETURN n.titulo AS nome
```

Cypher
Negação correspondência (*Match*) de *String*

- Pode-se usar *NOT* para excluir todas as correspondências de uma determinada *string*

```
MATCH (n)
WHERE NOT n.nome ENDS WITH `r'
RETURN n
```

Cypher
Operadores

- **Matemáticos**: +, -, *, /, %, ^
- **Comparação**: =, <>, <, >, <=, >=, IS NULL, IS NOT NULL
- **Especiais**: STARTS WITH, ENDS WITH, CONTAINS
- **Booleanos**: AND, OR, XOR, NOT

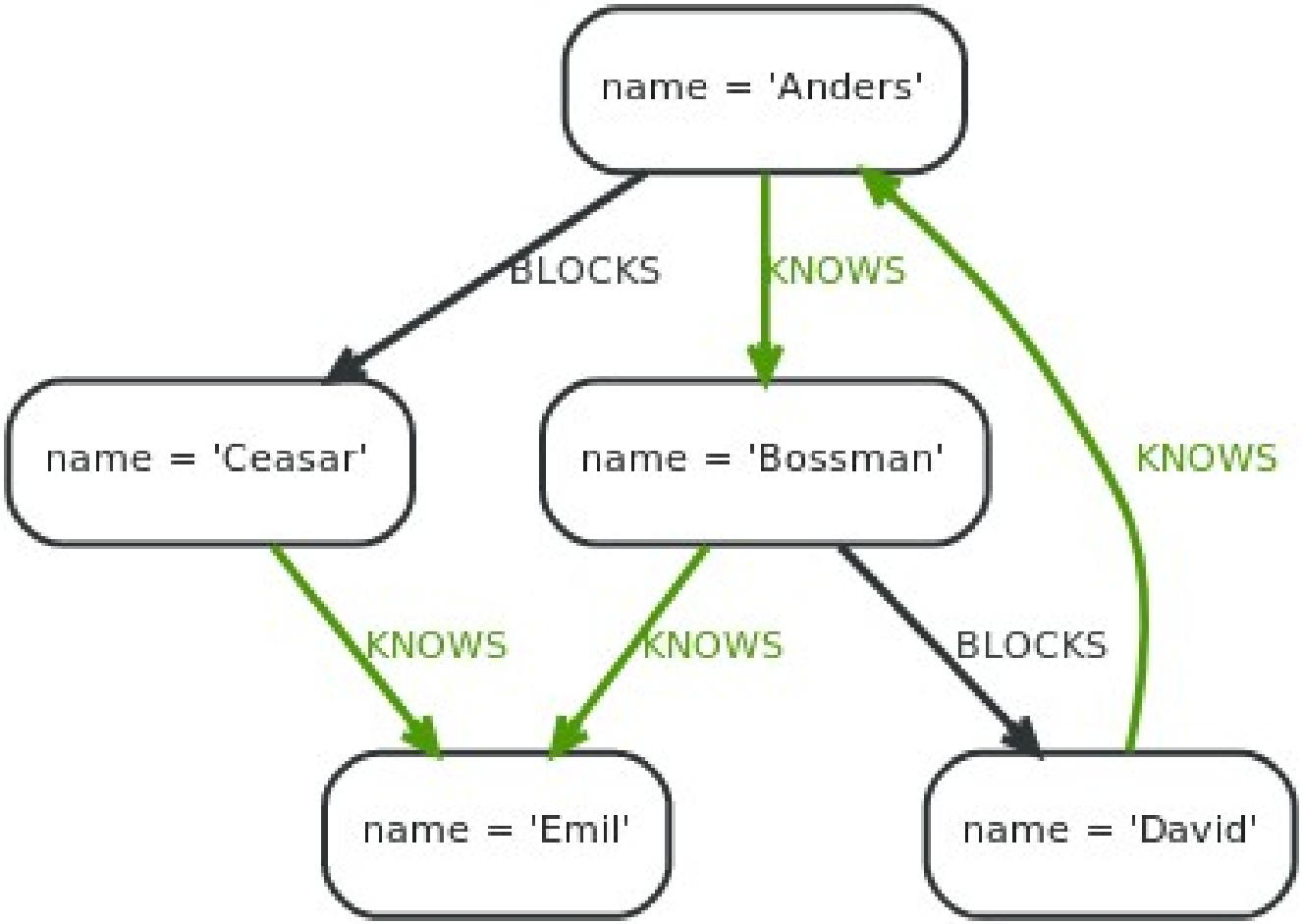
Cypher
Agregações

- ***Count, Sum, Avg, Min, Max***
- ***Funções Nativas e Stored Procedures do Usuário***

Comando	*Exemplo*
Count(*) - Número de linhas **Count(variavel)** – Número de valores não nulos **count(DISTINCT variavel)** – Remove duplicados	**MATCH** (:Pessoa) **RETURN count**(*) AS pessoas
Sum(a.propriedade) – Pesquisa e soma do total das linhas **Avg(a.propriedade)** – Média da propriedade	// Total e Média dos Salários **MATCH** (e:Empregados) **RETURN SUM**(e.sal),**AVG**(e.sal)

Cypher
Agregações com WITH

- ***WITH* é equivalente uma *sub-query* do *SQL* com recursos de agregação e filtragem**
- **Exemplo:**



Cypher
Agregações com WITH

Problema	Solução

Quais os nomes das pessoas conectadas a 'David' que possuem ao menos mais de um relacionamento	`MATCH (david { name: 'David' })--(otherPerson)-->()`
	`WITH otherPerson, count(*) AS foaf`
	`WHERE foaf > 1`
	`RETURN otherPerson.name`

Apresente a lista de nomes de pessoas na ordem inversa, limitada a 3	`MATCH (n)`
	`WITH n`
	`ORDER BY n.name DESC LIMIT 3`
	`RETURN collect(n.name)`

Começando em 'Anders', encontre todos os nós, ordene por nome descendente e encontre todos os nós conectados a esse resultado	`MATCH (n { name: 'Anders' })--(m)`
	`WITH m`
	`ORDER BY m.name DESC LIMIT 1`
	`MATCH (m)--(o)`
	`RETURN o.name`

Cypher
Exclusão (DELETE)

- **Exclusão de Nodos:**

```
MATCH (n:Inutil)
DELETE n
```

- **Exclusão de Nodos e Relacionamentos**

```
MATCH (n)
DETACH DELETE n
```

- **Exclusão de Relacionamentos**

```
MATCH (n { nome: 'Muttley' })-[r:Conhece]->()
DELETE r
```

Cypher
Alteração (SET)

- Alteração de uma propriedade de um Nodo:

```
MATCH (n { nome: 'Penélope' })
    SET n.sobrenome = 'Charmosa' RETURN n
```

- Exclusão de uma propriedade

```
MATCH (n { nome: 'Penélope' })
    SET n.nome = NULL RETURN n
```

- Cópia de propriedades entre nodos e relacionamentos

```
MATCH (at { nome: 'Mario' }), (pn { nome: 'Bros' })
    SET at = pn RETURN at, pn
```

- Adicionando mais de uma propriedade:

```
MATCH (peter { nome: 'Peter' })
    SET peter +=
            { faminto: TRUE , posicao: 'Empreendedor' }
```

- Inserir mais de um Rótulo para o mesmo Nodo

```
MATCH (n { nome: 'Peter' })
    SET n:Pessoa:Chefe RETURN n
```

Cypher
FOREACH

- Utilizado para alterar propriedades em uma lista ou agregação
- Exemplo marcar todos os Nodos em um caminho:

```
MATCH p =(inicio)-[*]->(FIM)
WHERE inicio.nome = 'A' AND FIM.nome = 'D'
FOREACH (n IN nodes(p) | SET n.marcar = TRUE )
```

Cypher
ÍNDICES

- **Índice simples:**

```
CREATE INDEX ON :Pessoa(nome)
CALL db.indexes
DROP INDEX ON :Pessoa(nome)
```

- **Índice composto:**

```
CREATE INDEX ON :Person(firstname, surname)
```

- **Definir Uso de Índices:**

```
MATCH (liskov:Scientist { name:'Liskov' })-[:KNOWS]->
      (wing:Scientist)-[:RESEARCHED]->
      (cs:Science { name:'Computer Science' })
      <-[:RESEARCHED]-(conway:Scientist { name: 'Conway' })
USING INDEX liskov:Scientist(name)
USING INDEX conway:Scientist(name)
RETURN liskov.born AS column
```

Cypher
Transações

- **Suporta transações por HTTP, *Drivers* e *cypher-shell***

- **Por Cyper-shell:**
  Executar: bin/cypher-shell -u usuario -p senha

- **Comandos Cyper-shell:**
  :begin Abre transação
  :commit Confirma (Commit) transação corrente
  :rollback Desfaz (Rollback) transação corrente

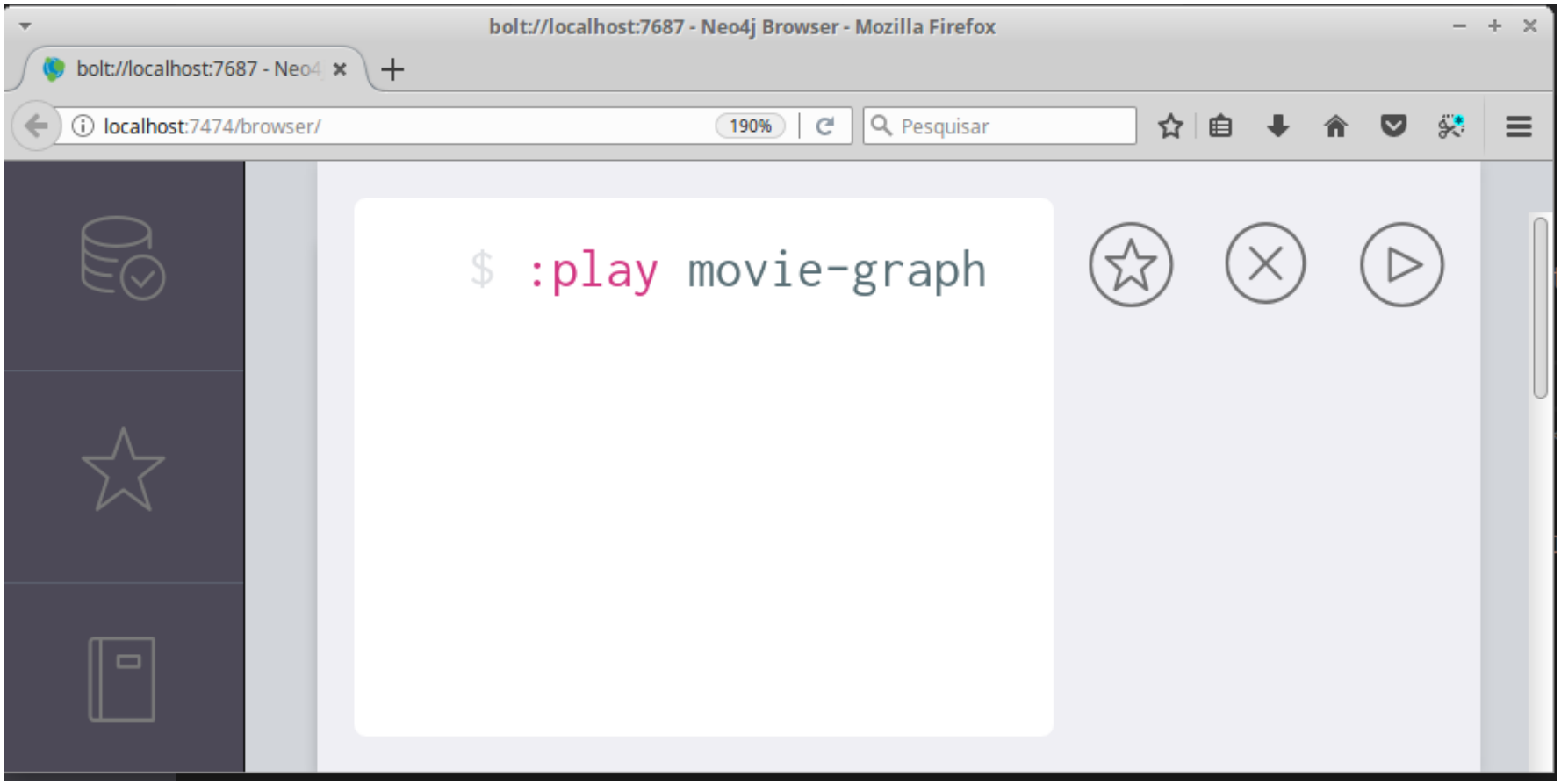
Cypher
Alta Disponibilidade

- **Basta configurar conf/neo4j.conf das máquinas/servidores que irão replicar:**

```
# Código da instância do servidor. Um código para cada máquina
ha.server_id=1

# Lista de todas as instâncias conhecidas (DNS ou IP)
ha.initial_hosts=neo4j-01.local:5001, neo4j-02.local:5001
#ha.initial_hosts=192.168.0.20:5001, 192.168.0.21:5001

# HA – Alta Disponibilidade ou SINGLE – Modo Simples, padrão
dbms.mode=HA

# Conector HTTP
dbms.connector.http.enabled=true
dbms.connector.http.listen_address=:7474
```

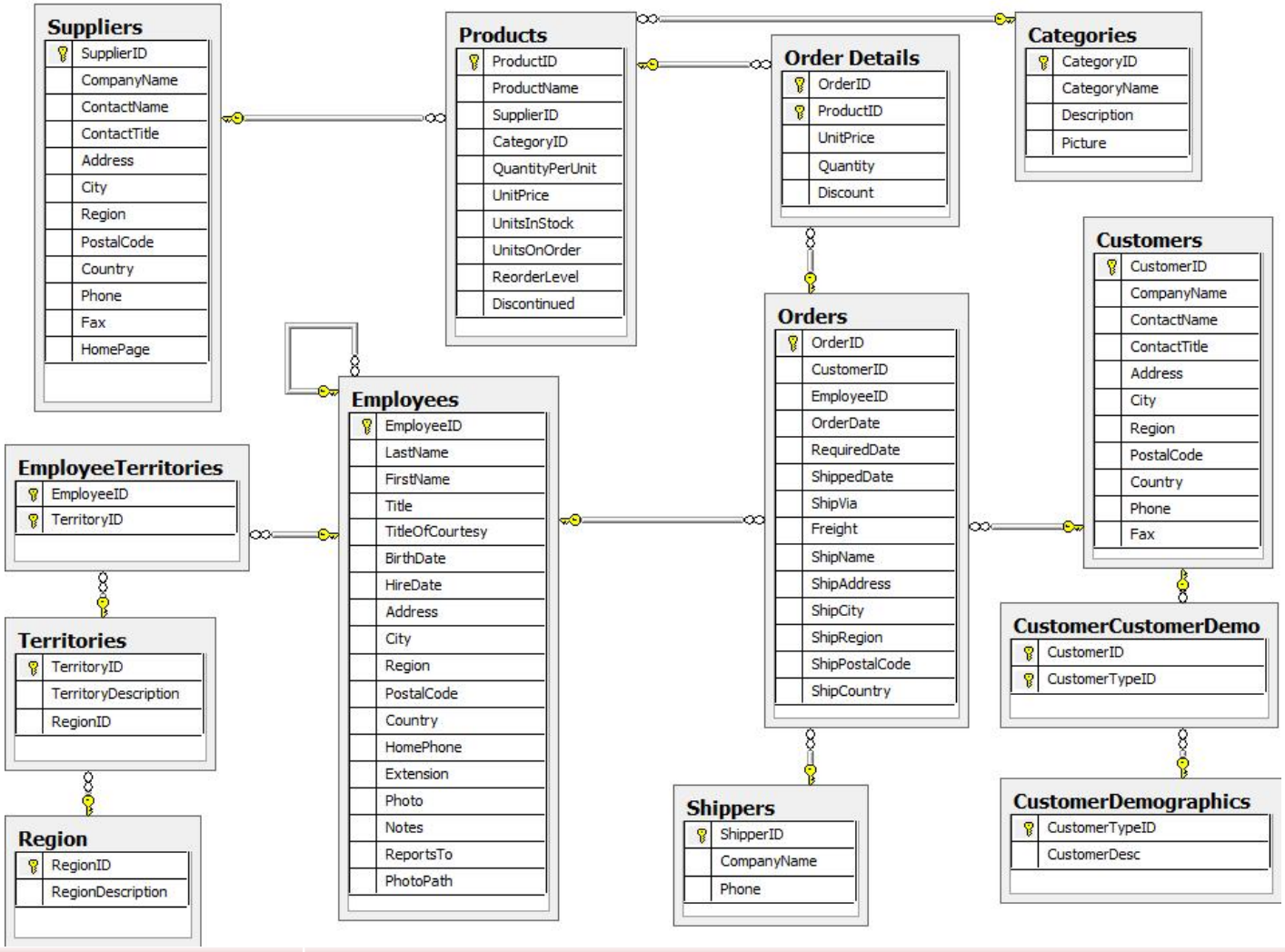
Cypher
Conhecendo o *Cypher* na prática ...

- **Entre no gerenciador do Neo4J no navegador (http://localhost:7474/) e digite e execute :play movie-graph conforme demonstrado abaixo:**



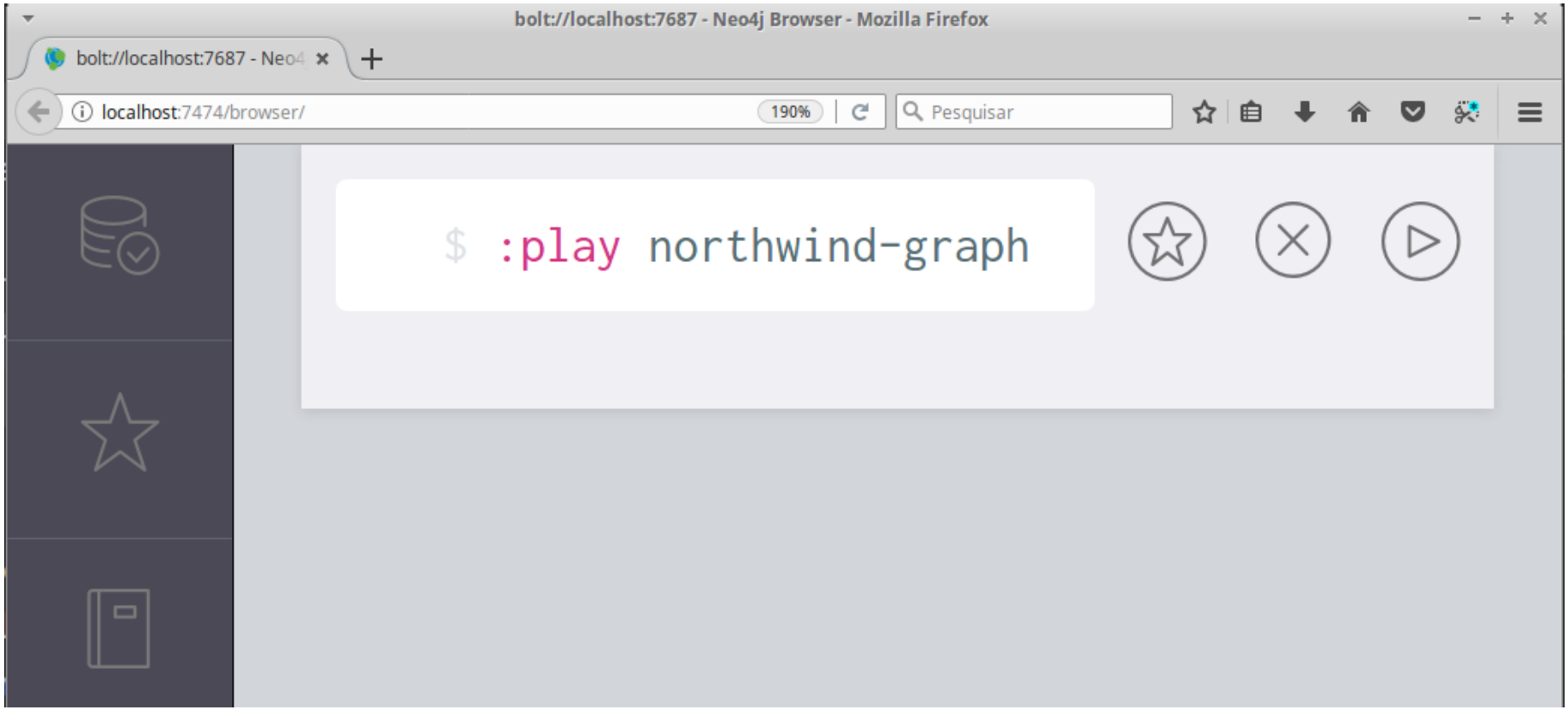
Cypher
Outro exemplo baseado no banco ***NorthWind***

- **Digite :play northwind-graph em http://localhost:7474/**



Cypher
Conhecendo o *Cypher* na prática ...

- **Entre no gerenciador do Neo4J no navegador (http://localhost:7474/) e digite e execute :play northwind-graph conforme demonstrado abaixo:**



Lembre-se

" Unidos, venceremos. Divididos, Cairemos. "

Bob Marley