



NoSQL - Bancos de Dados baseados em Grafos

Prof. Antonio Guardado



AGENDA

- ▶ Grafos
 - ▶ Exemplos
 - ▶ Origem
 - ▶ Aplicações
 - ▶ Definição Matemática
 - ▶ Tipos e Propriedades
 - ▶ Percurso
 - ▶ Representação
- ▶ Bancos de Dados Grafos
 - ▶ Principais SGBDs
 - ▶ Ranking
 - ▶ Quem utiliza
- ▶ NEO4j
 - ▶ Propriedades
 - ▶ Linguagem Cypher
- ▶ Referências

GRAFOS - EXEMPLOS

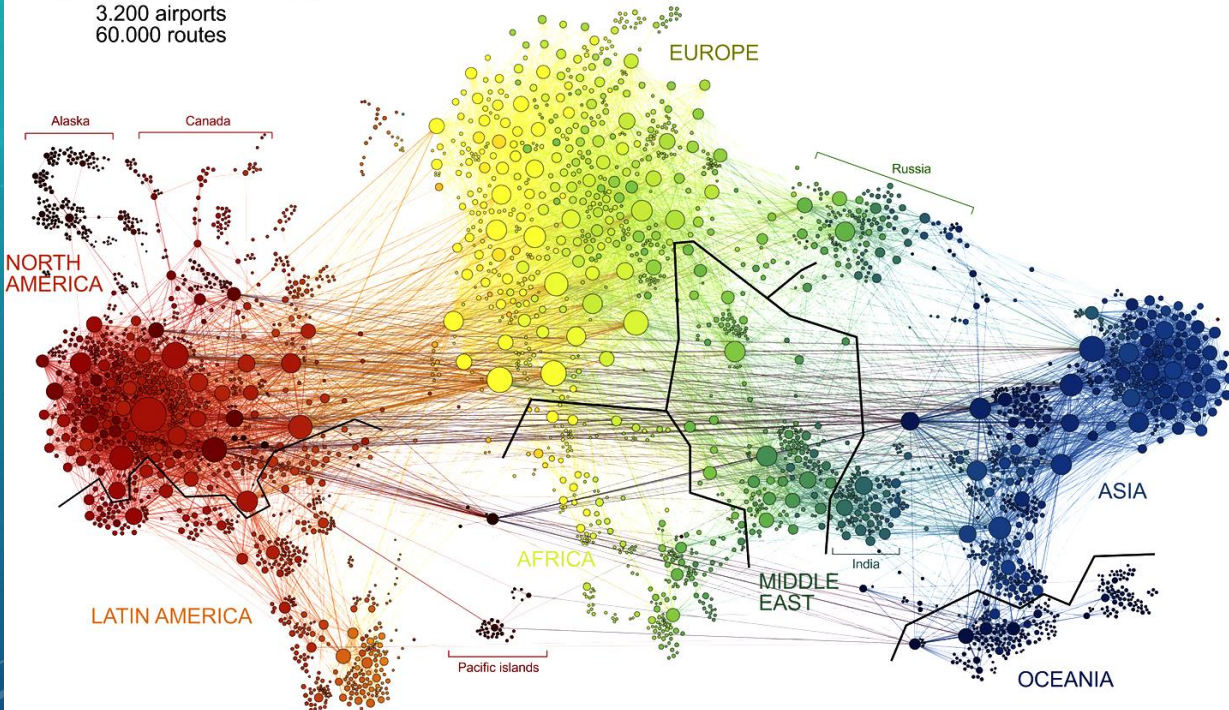


Estações Metrô SP

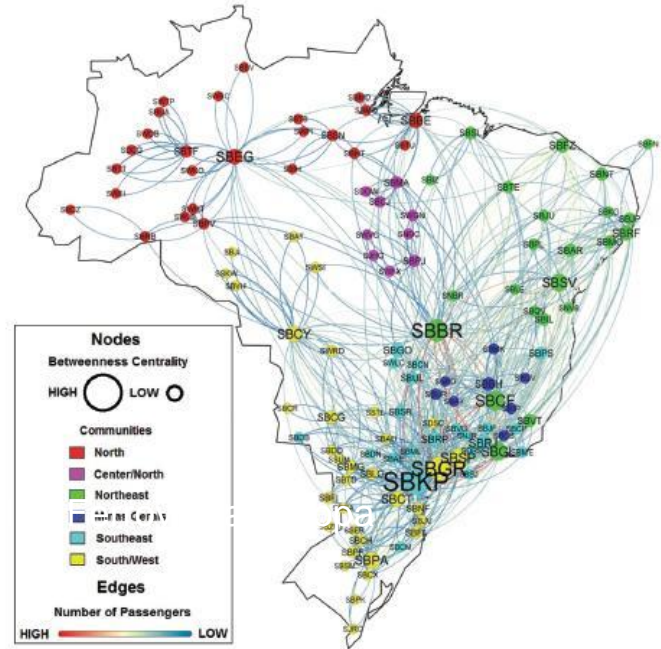
GRAFOS - EXEMPLOS

TRANSPORTATION CLUSTERS

3.200 airports
60.000 routes

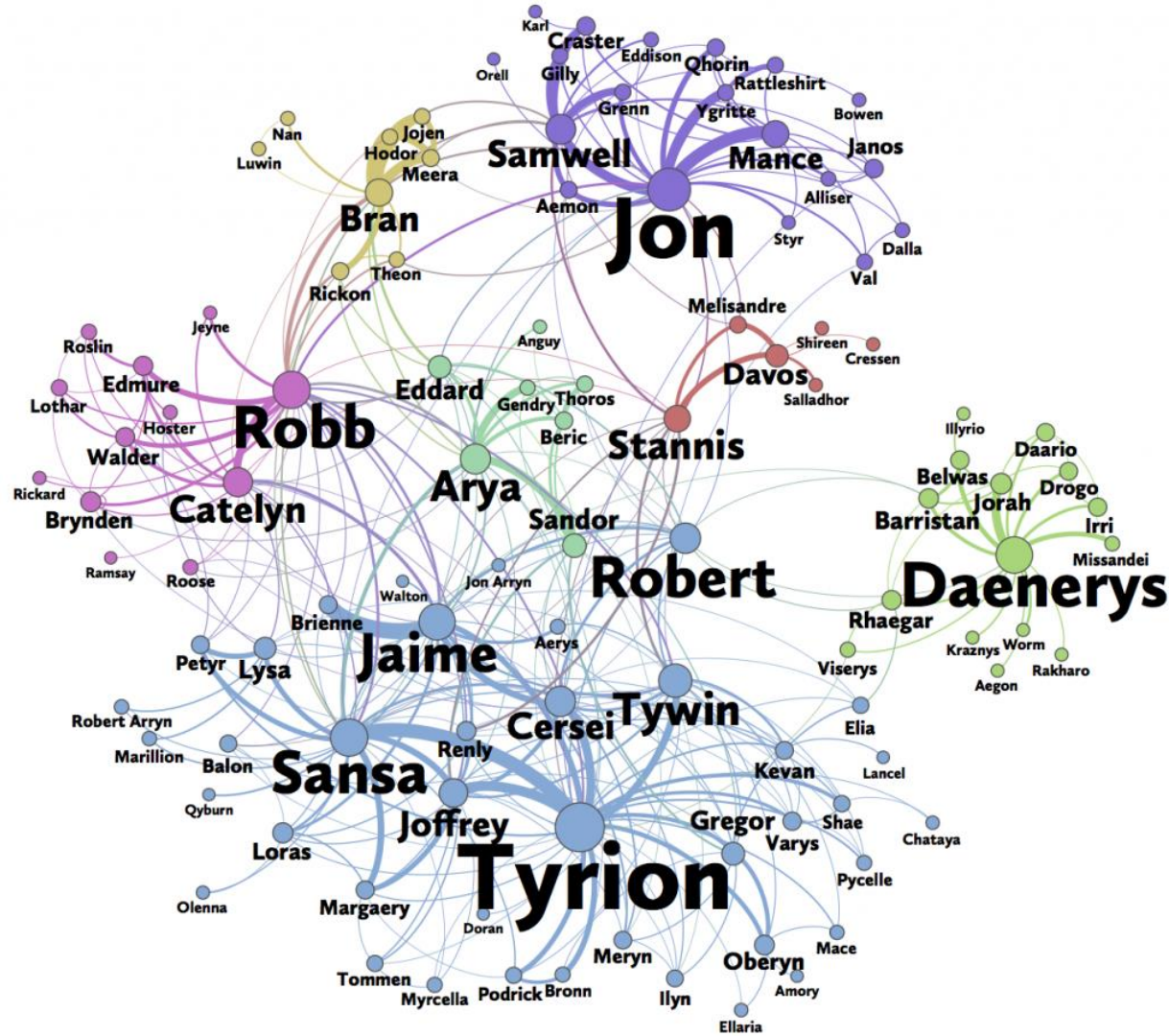


Rotas Aéreas



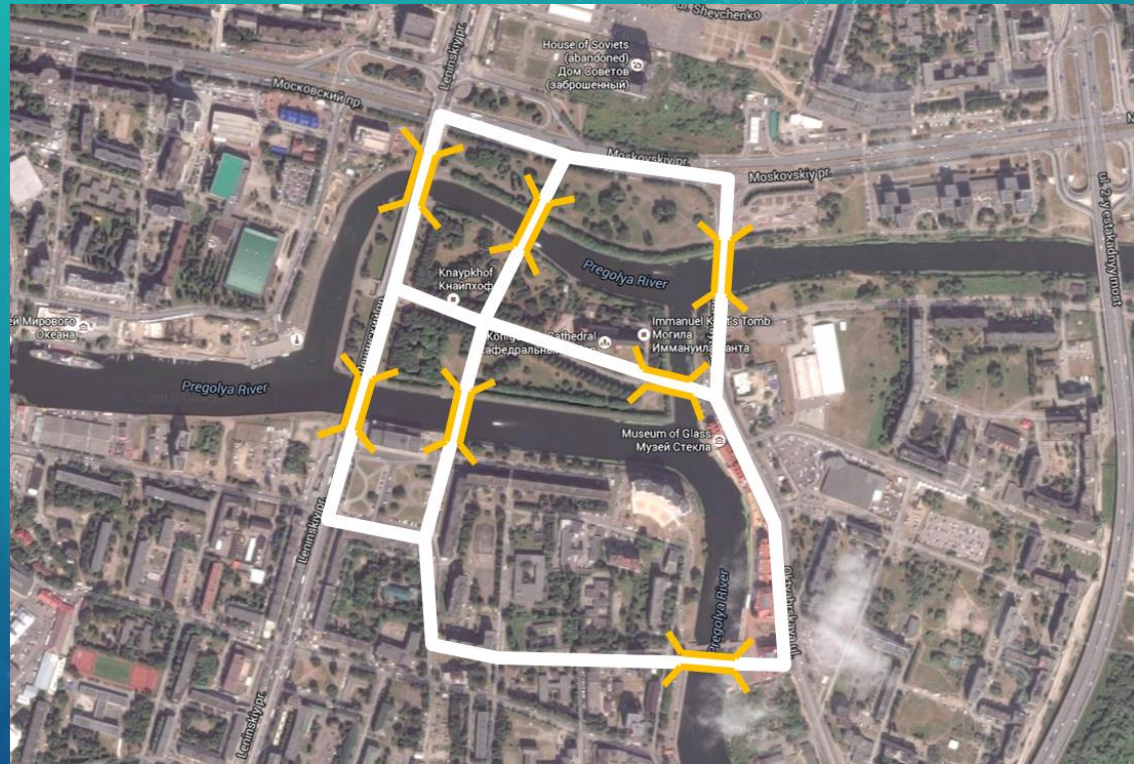
GRAFOS - EXEMPLOS

Teia de Personagens em Filmes - GOT

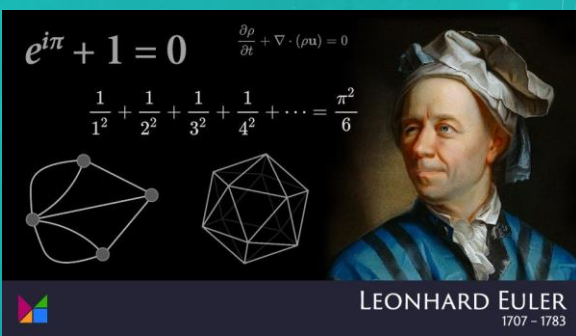


ORIGEM

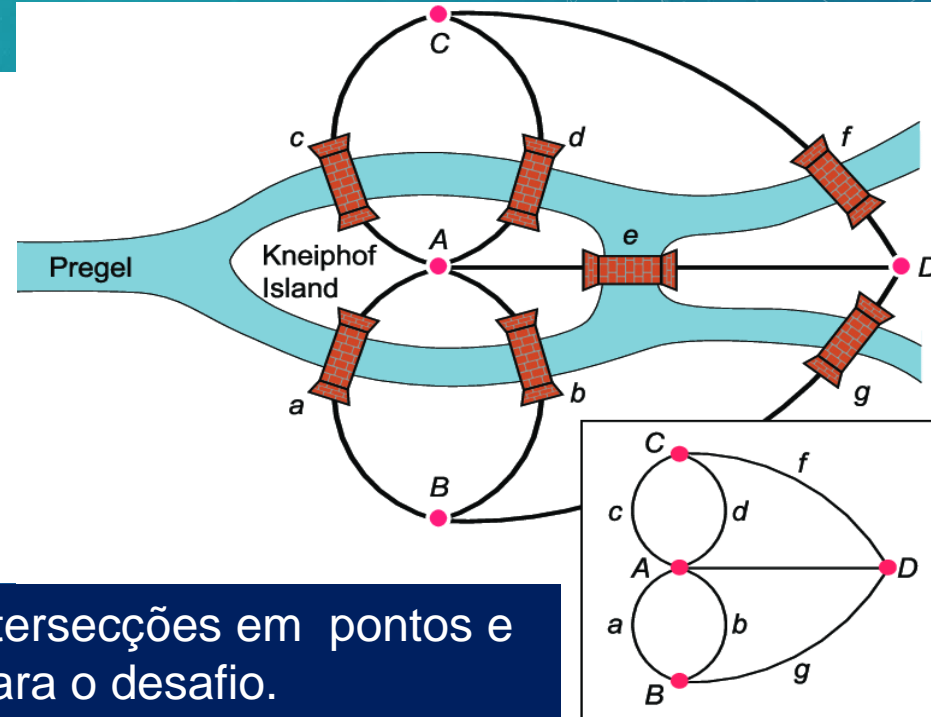
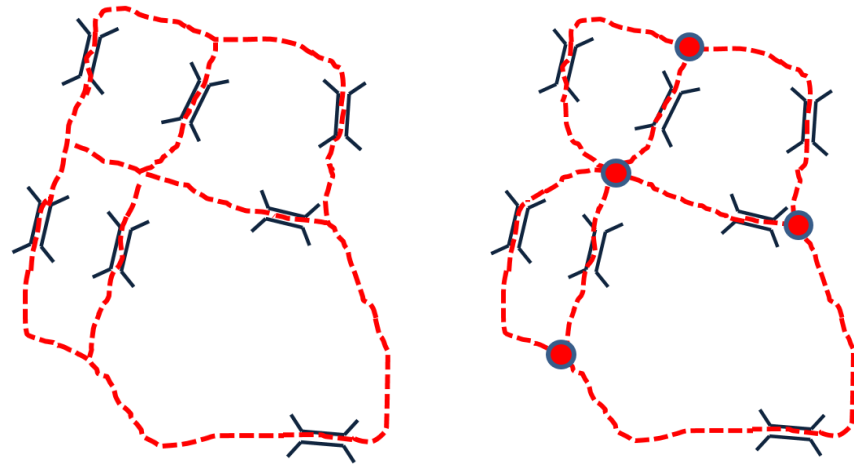
- Königsberg, por volta de 1735, cidade localizada na antiga Prússia (situada em território russo, atualmente tem o nome de Kaliningrado) era, e continua a ser, atravessada pelo rio Pregel.
- Existiam **sete pontes entre duas pequenas ilhas que as ligavam entre si e a cada uma das margens da cidade**. As pontes apresentavam uma configuração como podemos observar na figura ao lado .



Desafio : Dar uma volta pela cidade, partindo de uma das margens ou de uma das ilhas, atravessando cada ponte uma e uma só vez e regressando ao ponto de partida



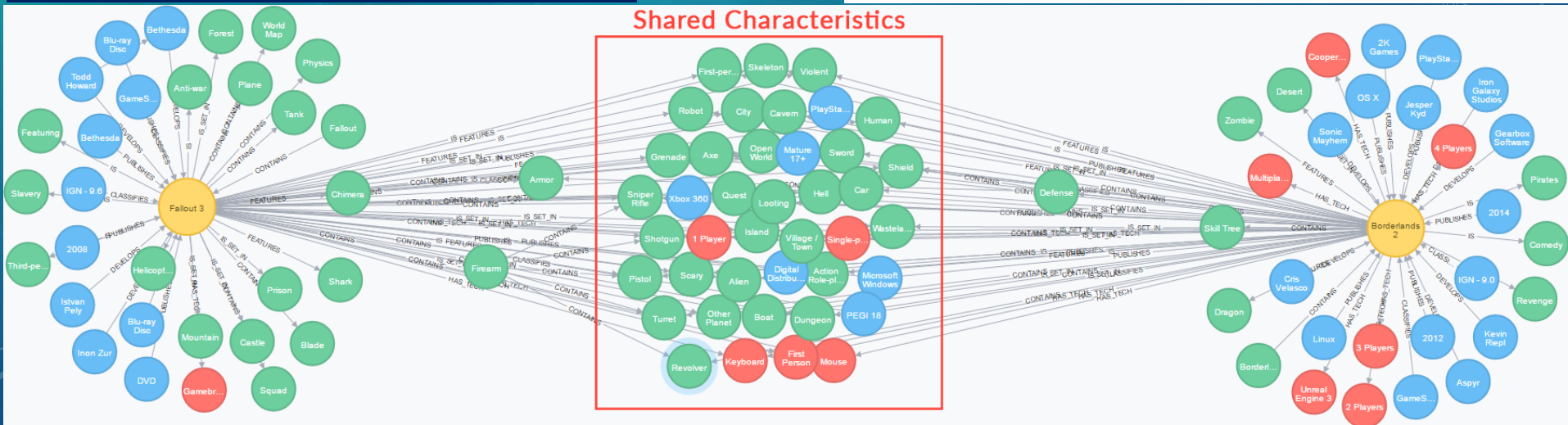
Precursor da Teoria de Grafos - Euler - 1736



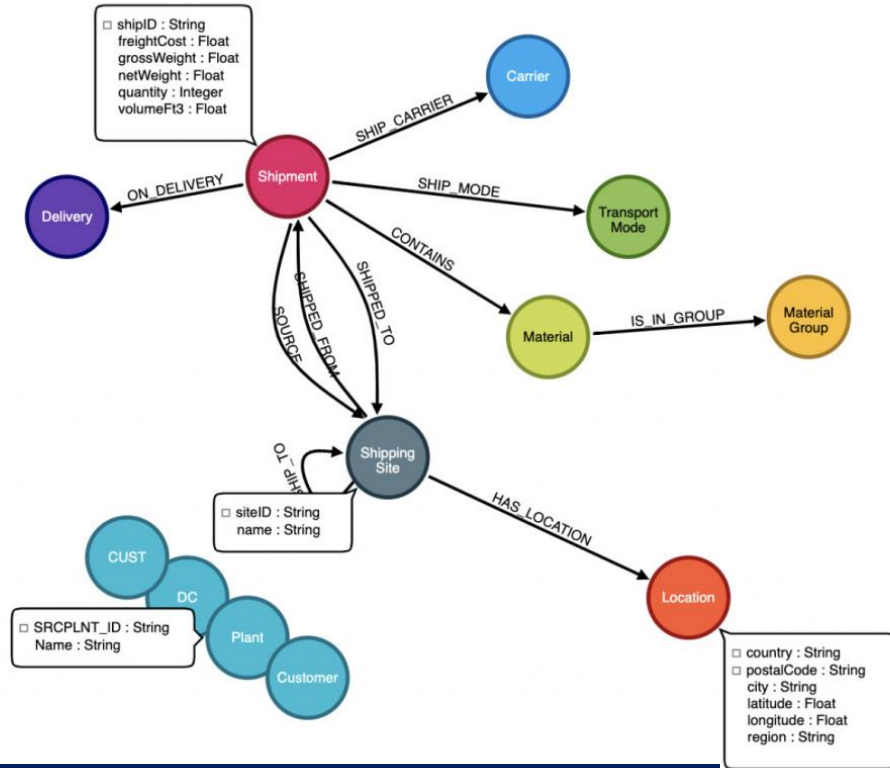
Transformou os caminhos em retas e as intersecções em pontos e provou que não existia caminho possível para o desafio.

Detecção de Fraude

Sistemas de Recomendação em tempo real

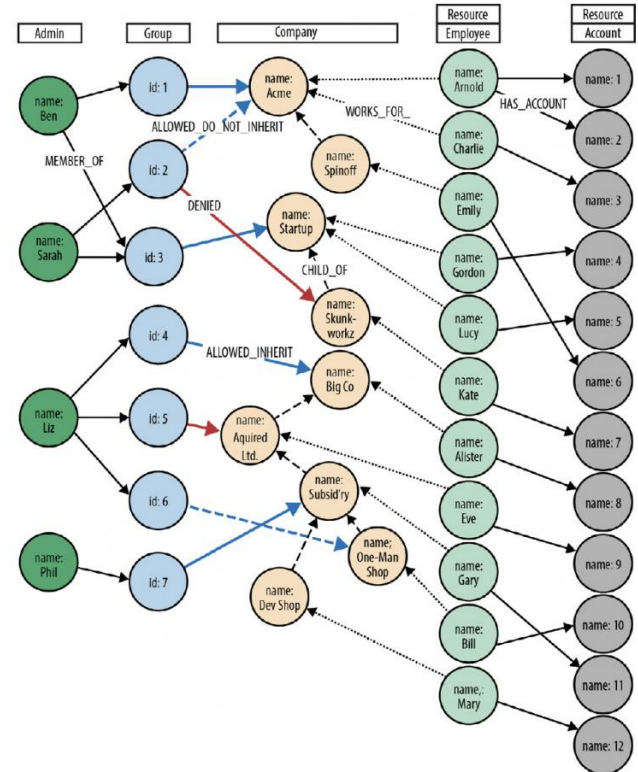


APLICAÇÕES DE GRAFOS



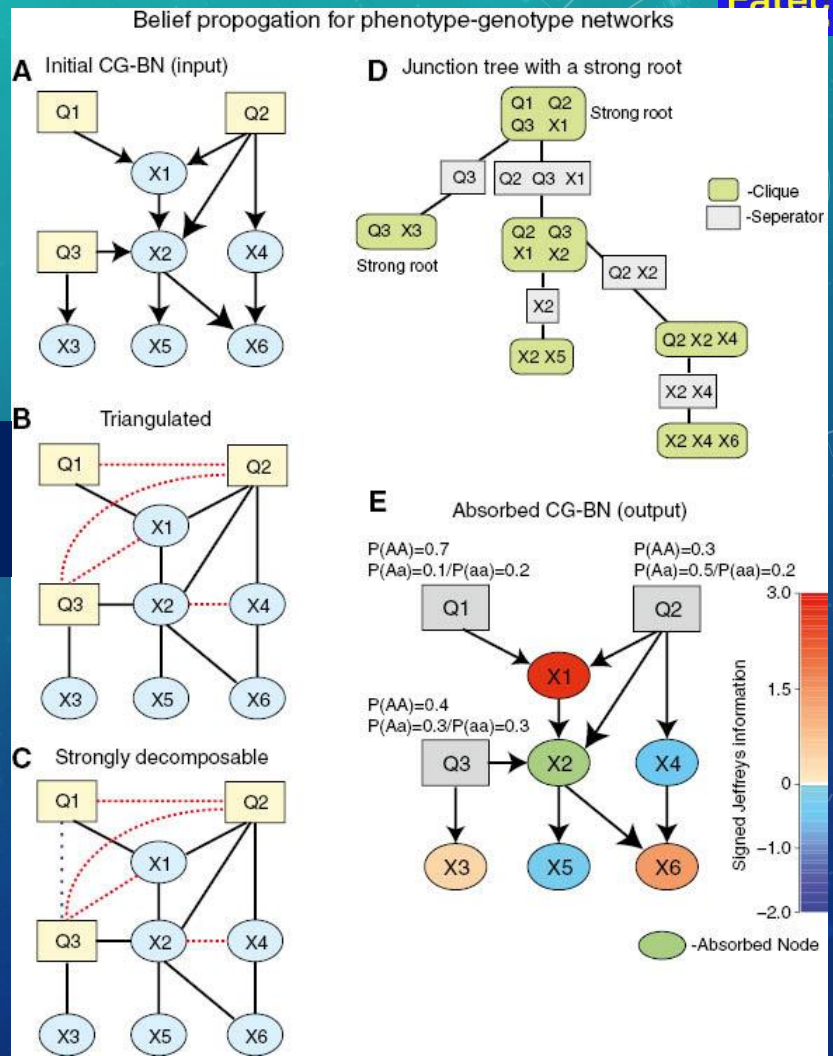
Gerenciamento da Cadeia de Suprimentos

Below is an example of Telenor's data model.



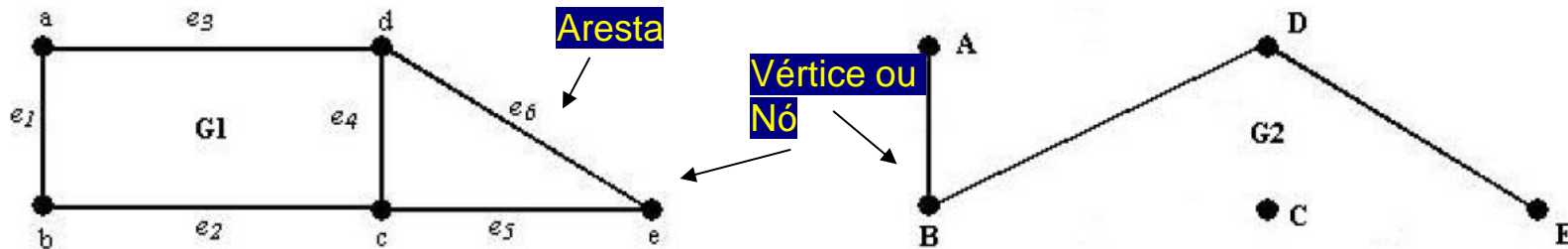
Gerenciamento de identidade e acesso

Fatec

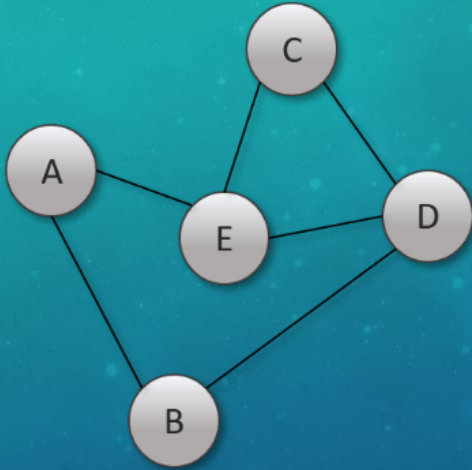


DEFINIÇÃO

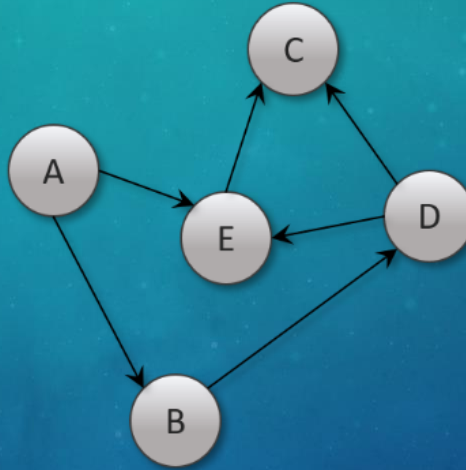
- Um grafo G é definido por $G = (V, E)$, sendo que V representa o conjunto de vértices (nós) e E o conjunto de arestas (i, j) , onde $i, j \in V$. Dois nós i, j são vizinhos, denotado por $i \sim j$, se eles estão conectados por uma aresta.
- A Figura 1.1 mostra dois exemplos de grafos: o grafo $G1$ consiste dos conjuntos $V = \{a, b, c, d, e\}$ e $E = \{e1, e2, e3, e4, e5, e6\}$; $G2$ possui o nó C que não é conectado com nenhum outro nó do grafo



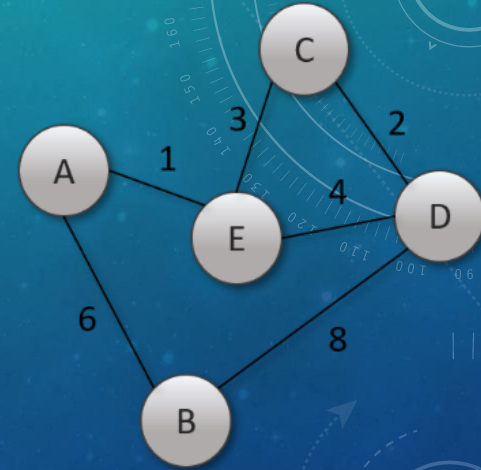
TIPOS MAIS COMUNS DE GRAFOS



Grafo Não Dirigido

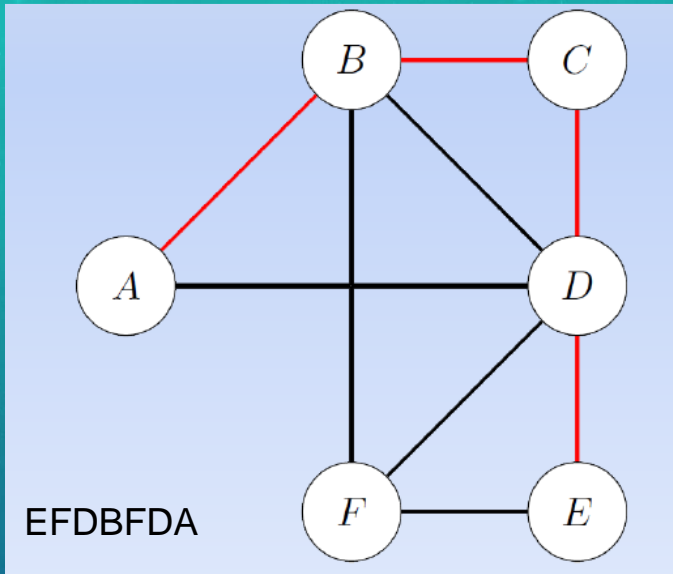


Grafo Dirigido



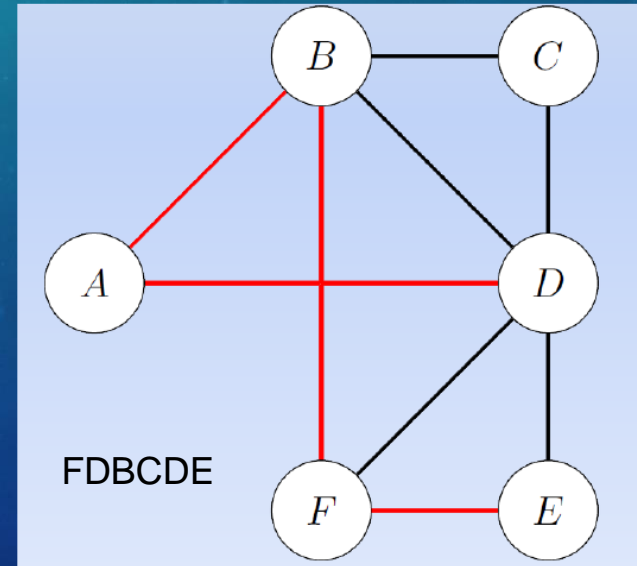
Grafo Ponderado

PERCURSOS EM UM GRAFO



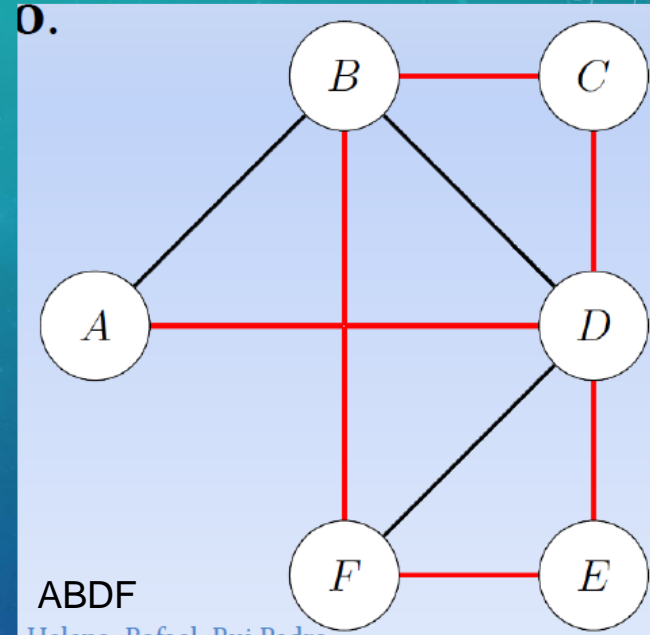
Passeio num grafo G é qualquer sequência de vértices adjacentes. Um passeio pode **repetir os vértices num grafo**. Se o passeio começa e acaba no mesmo vértice diz-se **fechado**, caso contrário diz-se aberto.

Trajetos num grafo G é um passeio cujas **arestas que o constituem são todas distintas**. Um trajeto, de comprimento não nulo, que comece e acabe no mesmo vértice designa-se por trajeto fechado ou circuito.



PERCURSOS EM UM GRAFO

Caminho num grafo é um passeio cujos **vértices e arestas que os constituem são todos distintos**. Um caminho, de comprimento não nulo, que comece e acabe no mesmo vértice designa-se por **caminho fechado** ou **ciclo**.



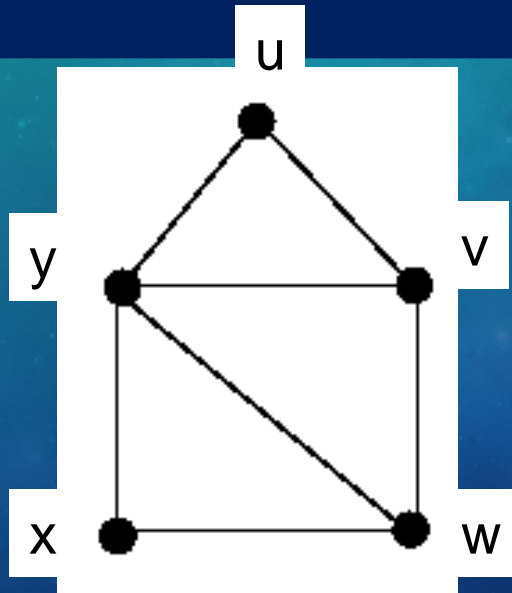
FDBCDE

REPRESENTAÇÕES DE UM GRAFO

- Lista de adjacência
- Lista de adjacência direcionado
- Matriz de adjacência
- Matriz de Incidência

LISTA DE ADJACÊNCIA

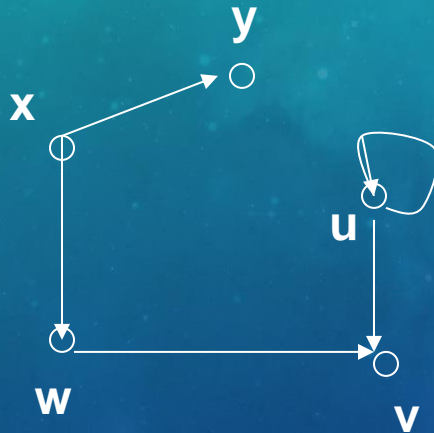
Uma maneira simples de armazenar grafos, é listando os vértices adjacentes a cada vértice do grafo



u: v,y
v: u,y,w
w: v,x,y
x: w,y
y: u,v,w,x

LISTA DE ADJACÊNCIA EM GRAFOS DIRECIONADOS

Tabela com vértices iniciais e finais (terminais)

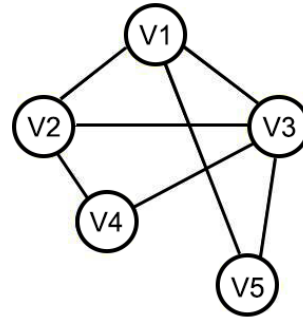


Inic.	Terminais
u:	u,v
v:	
w:	v
x:	y,w
y:	

MATRIZ DE ADJACÊNCIA

Se G é um grafo com vértices $\{1,2,3,\dots,n\}$, sua matriz de adjacência é a matriz $n \times n$ cujo elemento ij é o número de arestas ligando o vértice i ao vértice j

Matriz de Adjacência



	V1	V2	V3	V4	V5
V1	0	1	1	0	1
V2	1	0	1	1	0
V3	1	1	0	1	1
V4	0	1	1	0	0
V5	1	0	1	0	0

Imagem: Paulo Martins

MATRIZ DE INCIDÊNCIA

Se G é um grafo com vértices $\{1, 2, 3, \dots, n\}$ e arestas $\{1, 2, 3, \dots, m\}$, sua matriz de incidência é a matriz $n \times m$ cujo elemento ij é igual a

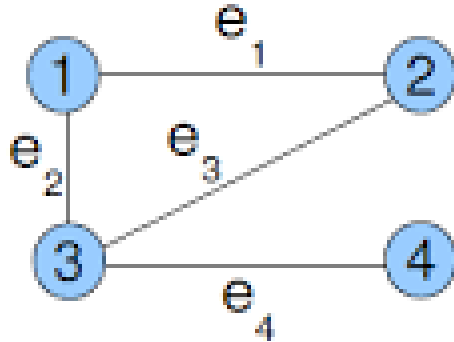
1 se a aresta e_j é incidente ao vértice v_i , ou
0, caso contrário

Arestas múltiplas são representadas usando colunas com entradas idênticas.

Laços são representados usando colunas com exatamente uma entrada igual a 1.

MATRIZ DE INCIDÊNCIA

Exemplo



	e_1	e_2	e_3	e_4
1	1	1	0	0
2	1	0	1	0
3	0	1	1	1
4	0	0	0	1

SGBDS BASEADOS EM GRAFOS



DATASTAX



Semantic Graphs

Follow Semantic standards (RDF/SPARQL)
Some support properties via RDF*



Property Graphs

Support labelled properties with Cypher or proprietary languages



Big Vendors with Graph Support

Support graph with proprietary database



Multi-model Graphs



























Support graph with APIs on top of NoSQL DBs



RANKING

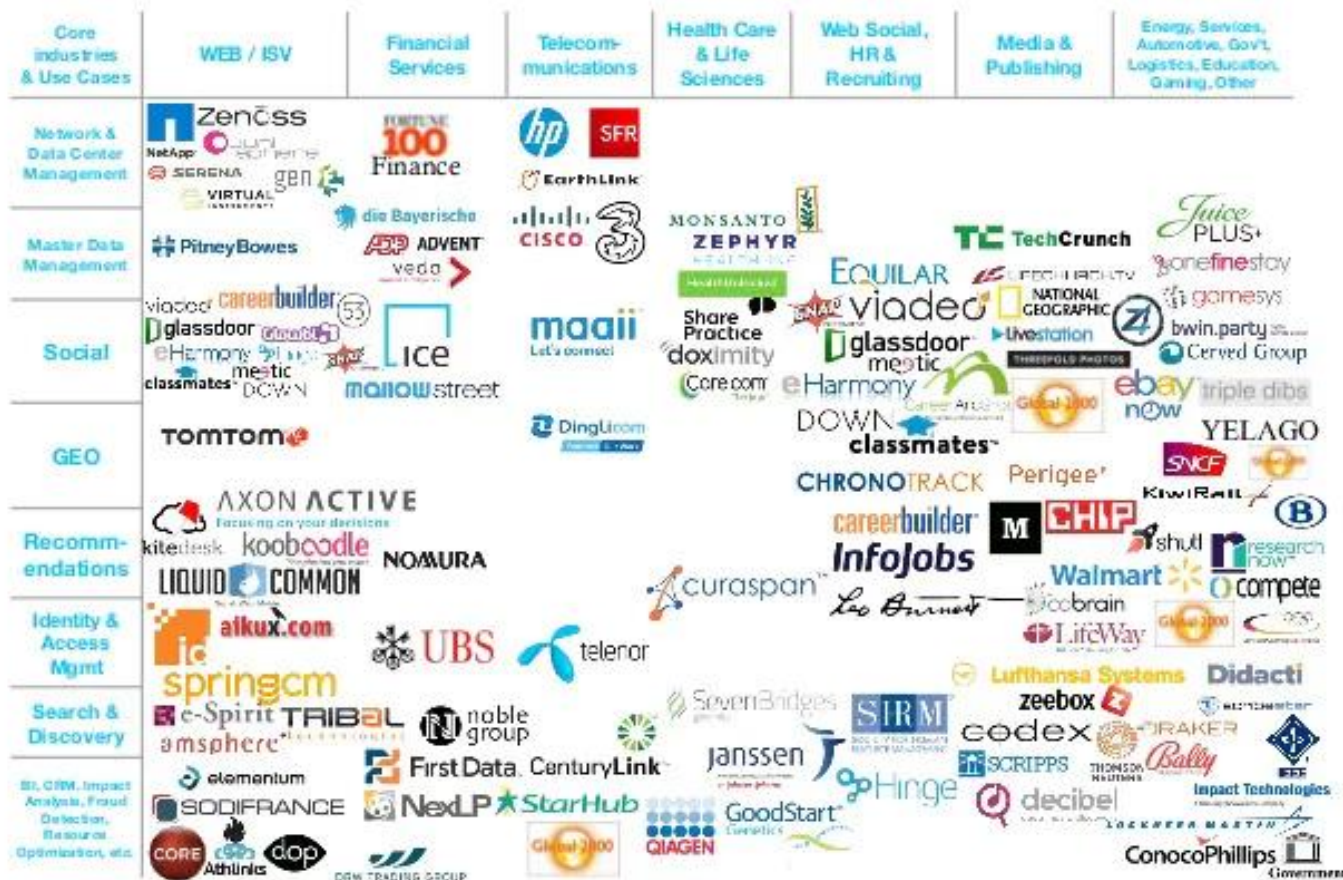
☐ include secondary database models

43 systems in ranking, October 2024

Rank			DBMS	Database Model	Score		
Oct 2024	Sep 2024	Oct 2023			Oct 2024	Sep 2024	Oct 2023
1.	1.	1.	Neo4j 	Graph	42.51	-0.17	-5.93
2.	2.	2.	Microsoft Azure Cosmos DB 	Multi-model 	24.50	-0.47	-9.80
3.	3.	3.	Aerospike 	Multi-model 	5.57	+0.41	-0.86
4.	4.	4.	Virtuoso 	Multi-model 	3.91	-0.08	-1.51
5.	5.	 6.	ArangoDB 	Multi-model 	3.44	+0.13	-0.83
6.	6.	 5.	OrientDB	Multi-model 	3.03	+0.01	-1.24
7.	7.	7.	Memgraph 	Graph	2.82	-0.09	+0.01
8.	8.	 9.	GraphDB 	Multi-model 	2.77	+0.01	+0.19
9.	9.	 10.	Amazon Neptune	Multi-model 	2.17	-0.03	-0.37
10.	10.	 12.	Stardog	Multi-model 	1.92	-0.01	-0.34
11.	11.	 8.	NebulaGraph 	Graph	1.86	-0.06	-0.91
12.	12.	 11.	JanusGraph	Graph	1.78	-0.07	-0.52
13.	13.	 14.	Fauna	Multi-model 	1.50	-0.05	-0.39
14.	14.	 13.	TigerGraph	Graph	1.46	+0.02	-0.64
15.	15.	15.	Dgraph	Graph	1.39	0.00	-0.47
16.	16.	16.	Giraph	Graph	1.11	-0.02	-0.60

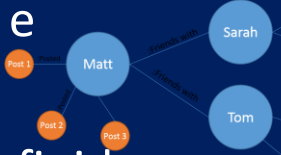
<https://db-engines.com/en/ranking/graph+dbms>

Neo4j Adoption Snapshot



QUEM
UTILIZA

Neo4J – Características Principais

- ▶ Index-free: não utiliza índices para organizar e buscar os dados, todos os relacionamentos são gravados com um ponteiro para o próximo nó
 - ▶ Transações ACID completas
 - ▶ Gerenciamento fácil, armazenamento e percurso de nós e relacionamentos
 - ▶ Schemaless: não possui estrutura e nem tipo de dados definido
 - ▶ Suporte a desenvolvimento rápido
 - ▶ Projetado para operações críticas de negócio e alto desempenho
- 
- ```
graph LR; Matt((Matt)) --- Post1((Post 1)); Matt --- Post2((Post 2)); Matt --- Post3((Post 3)); Matt --- Sarah((Sarah)); Matt --- Tom((Tom));
```



# PROPRIEDADES DO MODELO DE GRAFOS

- ▶ Nós : contém propriedades que são armazenados na forma chave-valor. Podem ser marcados com um ou mais rótulos; grupos de rótulos agrupam os nós e indicam os papéis que tem no conjunto de dados.
- ▶ Relacionamentos : são as conexões entre os nós (arestas ou arcos). Tem uma direção, um nome único e um nó inicial e final. Da mesma forma que os nós podem ter propriedades, acrescentando semântica às relações.
- ▶ Propriedades : características que se deseja registrar (atributos)
- ▶ Rótulos : ou Label, é a identificação do tipo de nó ou relacionamento, qual seu papel no modelo de dados







# Cypher

- ▶ Linguagem de consulta declarativa (como SQL)
- ▶ Maior legibilidade
- ▶ Otimizações de consultas
- ▶ É ótimo para consultas ad hoc, sem ter que escrever rotinas complexas de software para fazê-lo
- ▶ Combina a padrões de nós e relacionamentos para extrair informações ou modificar dados

(Neo4j)

- [:] ->

(Cypher)

# Cypher Queries

## ► CREATE :

- É uma cláusula para criar dados (nós ou relacionamentos)
- ( ) usado para indicar um nó
- p:peessoa p é uma variável, Pessoa é o rótulo
- {} para adicionar propriedades ao nó

CREATE (p:Pessoa {nome: "Ana", Localização: "São Paulo",  
RedeSocial: "Twitter"})



Propriedades

# Cypher Queries

## ► MATCH :

- É uma cláusula para buscar nós e relacionamentos (SELECT)
- (p1:Pessoa) um único padrão de nó com o rótulo 'Pessoa' que atribuirá correspondências à variável 'p1'
- WHERE a cláusula é usada para restringir os resultados
- p1.nome="Ana" compara a propriedade do nome ao valor "Ana"
- Cláusula DE RETORNO é usada para mostrar resultados específicos

```
MATCH (p1:Pessoa)-[r:conhece]-(p2:Pessoa)
```

```
WHERE p1.nome='Ana' AND p2.nome= 'Maria'
```

```
RETURN p1, p2, r
```

# Cypher Queries

## ► Atualizando propriedades de um nó

```
MATCH (p:Pessoa {nome: 'Ana'})
SET p.profissão = 'Engenheiro de Dados'
RETURN p
```

## ► Criando Relacionamento

```
MATCH (p:Pessoa {nome: 'Germano'}), (e:Empresa {nome: 'Google'})
CREATE (p)-[t:TRABALHA {início: 'MARÇO/2017', função: 'Desenvolvedor'}]->(e)
RETURN p, e, t
```

Rótulo do relacionamento



# Cypher Queries

## ► Excluindo um nó

```
MATCH (p:Pessoa)
WHERE p.nome = 'Jesuíno'
DELETE p
```

## ► Excluindo um relacionamento

```
MATCH (p:Pessoa)-[t:Trabalha]-(e: Empresa)
WHERE p.nome='Flora' AND e.nome = 'Prodesp'
DELETE t
```

# Cypher Queries

## ► INDEX

- É uma cópia redundante de informações para aumentar a eficiência da recuperação de dados.

```
CREATE INDEX idx_nome FOR (p:Pessoa) ON (p.nome)
```

```
MATCH (n:Pessoa) WHERE n.nome=$value
```

(índice pode ser usado diretamente para comparação da igualdade)

OU

```
MATCH (p:Pessoa)
```

```
WHERE p.nome IN ['Júlia', 'Flávia', 'Giovane']
```

```
RETURN p
```

# Cypher Queries

## ▶ Restrições (Constraints)

- ▶ Neo4J permite impor a integridade dos dados com a ajuda de restrições
- ▶ Restrições podem ser aplicadas a nós ou relacionamentos
- ▶ Adicionar uma restrição de unicidade para uma propriedade também vai adicionar um índice nessa propriedade

```
CREATE CONSTRAINT uq_cpf ON (p:Pessoa)
ASSERT p.cpf IS UNIQUE
```

# REFERÊNCIAS

- ✓ <https://neo4j.com/graphgists/> - Essencial sobre Grafo
- ✓ <https://neo4j.com/developer/cypher/> - Cypher
- ✓ <https://github.com/Readify/Neo4jClient/wiki> - Neo4j Client Documentation
- ✓ Instalação
- ✓ <https://www.youtube.com/watch?v=IKVB5b1PJMA>



# REFERÊNCIAS

