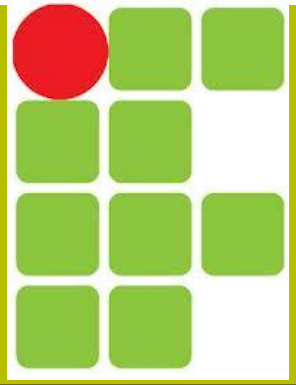
BANCOS DE DADOS II

Neo4j



Julio Cesar Kochhann
Rodrigo Silvestrin

IFRS – CAMPUS FARROUPILHA

Descrição

- Neo4j é um Banco de Dados NoSQL e Open Source (a partir de 2007)
- O modelo de dados do Neo4j é o Grafo, em particular um Grafo com Propriedades
- **Cypher** é a Linguagem de Consulta de Grafos do Neo4j (**SQL para Grafos!**)
- Alta velocidade de consulta através de Passagens (Traversals) que podem realizar milhões de “joins” por segundo
- Transações True ACID (Atomicidade, Consistência, Isolação e Durabilidade)

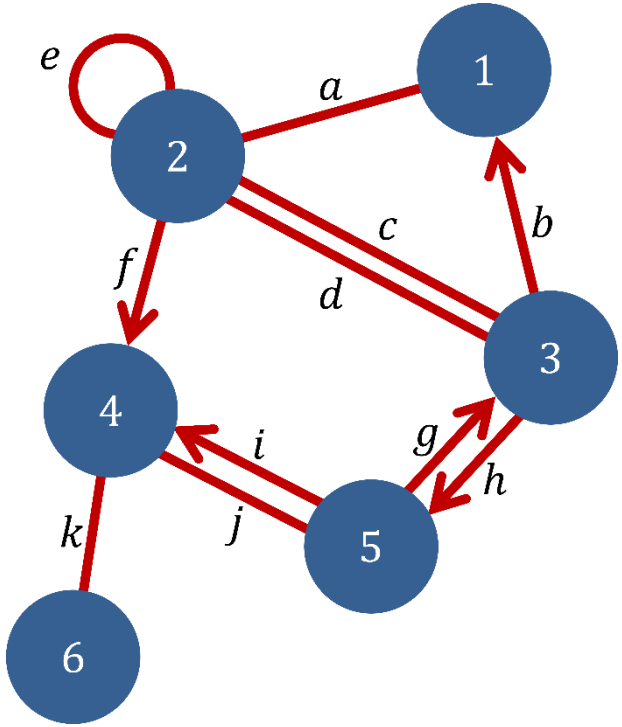
Casos de Uso

Walmart

ebay

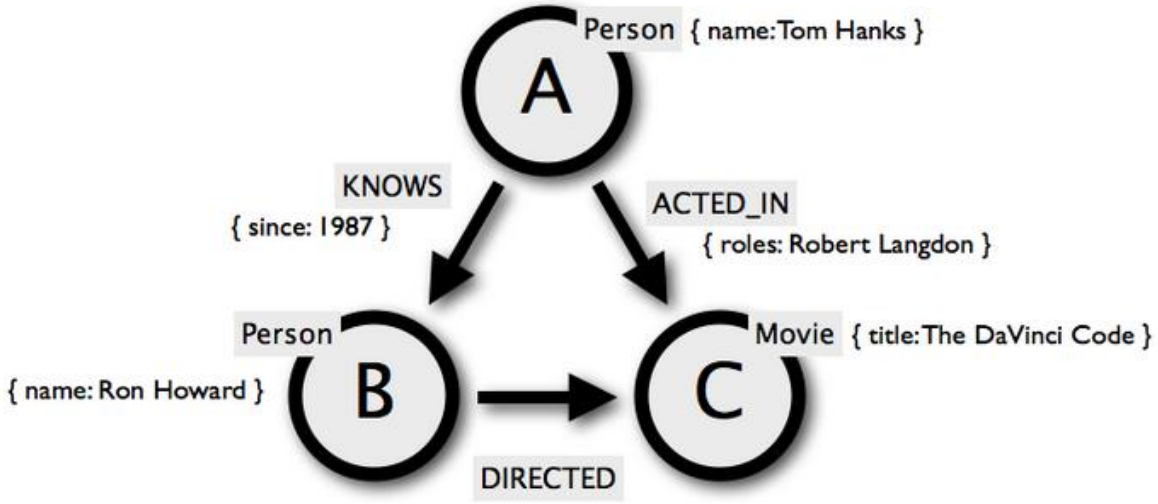
O que é um Grafo?

- Grafos são a maneira mais eficiente e natural de trabalhar com dados
- São profundamente intuitivos e imitam as interconexões de conceitos e ideias da mente humana
- Grafos são compostos de Vértices e Arestas que podem ser Direcionadas ou não



Banco de Dados de Grafos

- Os registros em um Banco de Dados de Grafos são chamados de Nodos
- Nodos são conectados através de Relacionamentos
- Propriedades são pares de Chave/Valor usados para adicionar informação sobre o Nodo ou Relacionamento e são referidos por Nodo.Propriedade ou Relacionamento.Propriedade
- Label é um nome que organiza Nodos em grupos



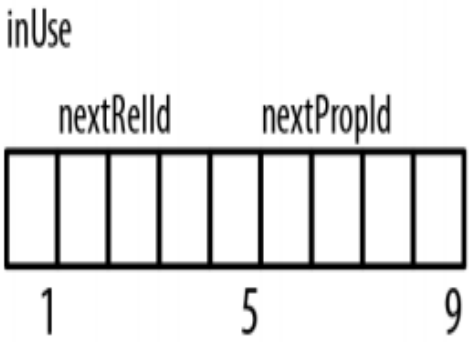
Estrutura de Dados

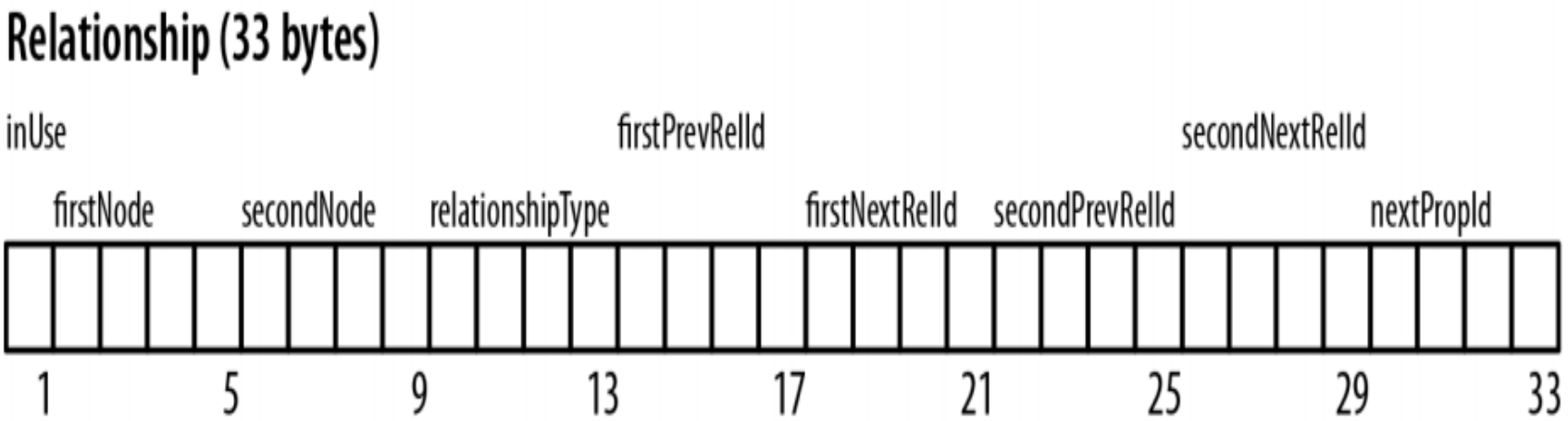
- Neo4j armazena os dados de Grafos em diversos arquivos
- Cada arquivo de armazenamento contêm o dado para uma parte específica do grafo (Ex. nodos, relacionamentos, propriedades)
- Alguns arquivos de armazenamento:
  - neostore.nodestore.db
  - neostore.relationshipstore.db
  - neostore.propertystore.db
  - neostore.propertystore.db.index
  - neostore.propertystore.db.strings
  - neostore.propertystore.db.arrays

Como é armazenado?

- neostore.nodestore.db
  - Tamanho: 9 bytes
  - 1º byte: Flag em-uso
  - Próximos 4 bytes: ID do primeiro relacionamento
  - Últimos 4 bytes: ID da primeira propriedade do nodo
- neostore.relationshipstore.db
  - Tamanho: 33 bytes
  - 1º byte: Flag em-uso
  - Próximos 8 bytes: IDs dos nodos do início e do final do relacionamento
  - 4 bytes: Ponteiro para o Tipo do relacionamento
  - 16 bytes: Ponteiros para o próximo e anterior registros de relacionamento para cada nodo inicial e final
  - 4 bytes: Próximo ID de propriedade

- Estrutura do Nodo e Relacionamento:

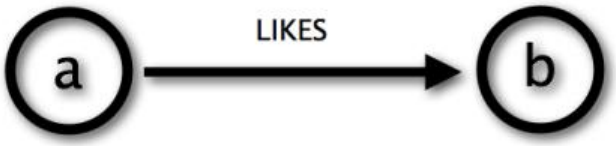




Cypher

- Cypher é uma linguagem de consulta Declarativa destinada a ser bastante Legível e Expressiva
- Permite que a gente descreva o que Selecionar, Inserir, Atualizar ou Deletar de um Banco de Dados de Grafos
- Cypher usa a técnica ASCII_art para representar os padrões

Cypher using relationship 'likes'



Cypher

```
(a) -[:LIKES]-> (b)
```

Cypher - Sintaxe

Nodos:

(a) Atores
(d) Diretores
(f) Filmes
( ) Nodo anônimo

Relacionamentos:

-[r]-> Um relacionamento definido como r

(a)-[r]->(f) Atores que tem um relacionamento r com Filmes

-[:ACTED_IN]-> O *Tipo* de Relacionamento é *ACTED_IN*

(a)-[: ACTED_IN]->(f) Atores que atuaram em algum Filme

(d)-[:DIRECTED]->(f) Diretores que dirigiram algum Filme

Propriedades

- Propriedades podem ser Booleanas, Numéricas ou Strings. Também podem ser Vetores de qualquer um desses três tipos

- Nodos com Propriedades:
  - (f {title:"The Matrix"}) Filme com uma propriedade *title*
  - (a {name:"Keanu Reeves",born :1964}) Ator com propriedades *name* e *born*

- Relacionamentos com Propriedades:

  - (a)-[: ACTED_IN {roles:["Neo“, “ “]}]->(f) Relacionamento *ACTED_IN* com propriedade *roles* (Nesse caso um vetor de Strings)

- Labels e propriedades:

  - (a:Pessoa) a é uma Pessoa
  - (b:Pessoa {nome:”Keanu Reeves”}) b é uma Pessoa com 1 Propiedade
  - (c:Animal {nome:”lula”,pensa:false}) c é um Animal com 2 Propriedades

Cypher - Cláusulas

- Propósito Geral:
  - RETURN: Retorna o resultado da query
  - ORDER BY: Ordena o resultado da query
  - SKIP/LIMIT: Limita os resultados da query
- Consulta:
  - MATCH: Combina um dado padrão com os dados do banco
  - WHERE: Filtra usando propriedades
- Agregação:
  - COUNT: Usado para contar o número de ocorências (identificador ou linha)
  - DISTINCT: Remove valores duplicados do resultado
  - COLLECT: Coleta todos os valores de uma lista (Coleção)

Cypher - Cláusulas

- Cláusulas de Escrita:
  - CREATE: Cria nodos e relacionamentos
  - MERGE: Cria nodos únicos
  - CREATE UNIQUE: Cria relações únicas
  - DELETE: Remove nodos e relacionamentos
  - SET: Atualiza propriedades e labels
  - REMOVE: Remove propriedades e labels
  - FOREACH: Realiza atualizações uma vez por elemento em uma lista
  - WITH: Divide a query em múltiplas partes e repassa o resultado de uma parte para a próxima

Cypher - Cláusulas

- Predicados:

- ALL: Testa se um predicado se encontra em toda a coleção
- ANY: Testa se um predicado se encontra em ao menos um valor da coleção
- NONE: Retorna true se um predicado não se encontra na coleção
- SINGLE: Retorna true se apenas um elemento da coleção corresponde ao predicado
- EXISTS: Retorna true se um Match de um determinado padrão se encontra no grafo

Outros:

- AS: Atribui um nome à uma propriedade
- RANGE: Cria uma coleção de números
- IN: Pesquisa dentro de uma coleção

Criando um Banco

- Gerenciador de Banco usado: Neo4j Community Edition:

- Na linha de comando do server:

```
CREATE (a:Person {name:'Tom Hanks'})
CREATE (b:Person {name:'Ron Howard'})
CREATE (c:Movie {title:'The DaVinci Code'})
CREATE
        (a)-[:ACTED_IN {roles:['Robert Langdon']}]->(c),
        (b)-[:DIRECTED]->(c),
        (a)-[:KNOWS {since:[1987]}]->(b)
```

OBS.: Copie o código acima e cole na linha de comando. Para rodar é só clicar Enter ou o botão Play no canto superior direito da tela.

P.S.: Este exemplo provêm do Grafo da página 5

Exemplos de Queries

- Retorna todos os nodos armazenados no Banco

```
MATCH (n) return n
```

- Retorna o nome do ator e do filme em que atuou:

```
MATCH (actor)-[:ACTED_IN]->(movie) return actor.name,movie.title
```

- Retorna o nome do ator e o papel que fez em algum filme:

```
MATCH (actor)-[role:ACTED_IN]->(movie)
RETURN actor.name,role.roles
```

- Retorna o nodo que satisfaz a cláusula where:

```
MATCH (actor) WHERE actor.name = 'Ron Howard' RETURN actor

MATCH (m:Movie) WHERE m.title CONTAINS 'Vinci' RETURN m
```

- Retorna um Filme que foi dirigido por um conhecido do Tom Hanks:

```
MATCH (actor:Person)-[:KNOWS]-()-[:DIRECTED]-(movie)
WHERE actor.name = 'Tom Hanks'
RETURN DISTINCT movie
```

Exercício

1- O objetivo da tarefa é desenvolver um clone do aplicativo “Tinder”. Crie 5 perfis de usuário (um deve ser o seu) cada qual com as seguintes propriedades: name, age, sex, city e likes (esta deve ser uma coleção com pelo menos 4 Strings que representam os gostos daquele usuário). Uma vez que o “aplicativo” está com a base populada crie relações do Tipo :MATCHES entre usuários que possuem ao menos um interesse em comum, sejam da mesma cidade e do sexo oposto. Depois exiba o(s) perfis (e suas propriedades) que combinam com o seu.

OBS.: Para a atividade funcionar:

- Coloque gostos em comum entre usuários
- Use a mesma formatação nos campos city e likes para todos os perfis

Links Úteis

Site Oficial: http://neo4j.com
Manual do Neo4j: http://neo4j.com/docs/stable/
Curso Online Cypher:
http://neo4j.com/graphacademy/online-training/