UFSC-CTC-INE-PPGCC

INE 410131 – Gerencia de Dados para Big Data

Aula 5 – SGBD NoSQL Neo4j

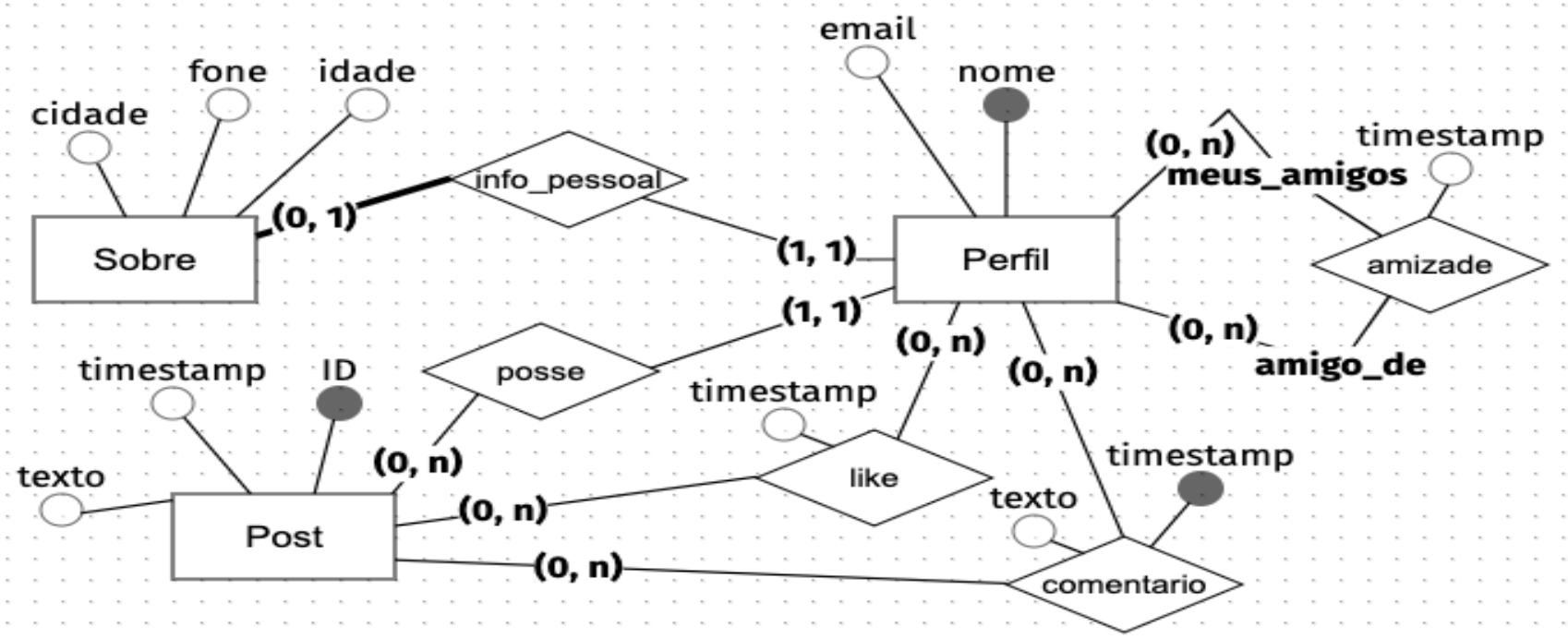
Neo4j

- SGBD NoSQL orientado a grafos
  - SGBD mais utilizado nesta categoria
- 23º SGBD em uso comercial no mundo
  - *https://db-engines.com/en/ranking*
- Quando um BD orientado a grafos é interessante?
  - quando os dados possuem muitos relacionamentos entre si
  - quando os relacionamentos são muito dinâmicos
  - quando os relacionamentos são muito acessados
- Exemplos de domínios de aplicação
  - Redes sociais
  - Sistemas de recomendação
  - Sistemas baseados em rotas
  - ...

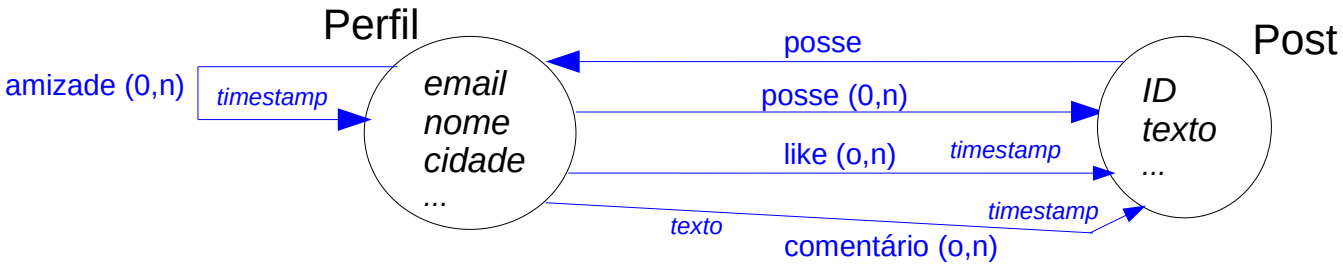
Neo4j – Linguagem Cypher

- Linguagem declarativa para definição e acesso a vértices (ou nodos) e arestas de um grafo
- Indexa propriedades de nodos e arestas

Estudo de Caso – Rede Social



projeto lógico



Criação de Nodos

- Comando `CREATE`
  - `CREATE (n);`
  - `CREATE (p:Perfil);`
  - `CREATE (p1:Perfil{nome:"Karla"});`
  - `CREATE (p2:Perfil{nome:"Ronaldo",`
    `email:"r.mello@ufsc.br"});`

Criação de Arestas

- Comando `CREATE`
  - `CREATE (nodo1)-[]→(nodo2);`
  - `CREATE (nodo`$_1$`)-[:label]->(nodo`$_2$`);`
  - `CREATE (nodo`$_1$`)-[:label{att`$_1$`:"val`$_1$`", …,att`$_n$`:"val`$_n$`"}]->(nodo`$_2$`);`
- Exemplo

```
CREATE (p1:Perfil{nome:"Lucas"}),
       (p2:Perfil{nome:"Luma"}),
       (p1)-[:amizade{timestamp:"4/4/2023
              15:00"}]->(p2);
```

Criação de Arestas

- Quando os nodos já existem
  - deve-se localizar os nodos primeiramente
- Exemplo

```
MATCH (p1:Perfil),(p2:Perfil)
WHERE p1.nome = “Karla”
AND p2.nome = "Ronaldo”
CREATE (p1)-[:amizade{timestamp:”4/4/2023
15:00”}]->(p2);
```

Consultas sobre um Grafo

- Comando `MATCH`
  - buscar todos os nodos do BD
    - **`MATCH (n) RETURN n;`**
  - buscar os nodos de um determinado rótulo
    - `MATCH (n:Post) RETURN n;`
  - buscar propriedades de nodos de um determinado rótulo com uso de filtro
    - `MATCH (n:Post) WHERE n.ID > 100 RETURN n.texto;`
  - buscar propriedades de nodos com aresta em qualquer direção
    - `MATCH (Perfil{nome:"Karla"})-[]-(n:Perfil) RETURN n.nome ORDER BY n.nome;`
  - buscar propriedades de nodos com aresta tendo direção específica e filtro de existência
    - `MATCH (Perfil{nome:"Karla"})-[]->(n:Perfil) WHERE exists(n.cidade) RETURN n.nome;`

Consultas sobre um Grafo

- Comando `MATCH`
  - buscar dados relacionados com aresta específica
    - `MATCH (Perfil{nome:"Karla"})-[:amizade]-(n:Perfil) RETURN n.nome;`
  - buscar rótulos ou propriedades de arestas
    - `MATCH (Perfil{nome:"Karla"})-[c:comentario]-(n:Post) RETURN c.texto;`
  - buscar dados com múltiplos relacionamentos
    - `MATCH (n:Perfil)-[:comentario|:like]-(p:Post) RETURN n.nome, p.ID;`
  - buscas com filtros envolvendo saltos no grafo
    - `MATCH (Perfil{nome:"Karla"})-[:amizade]-()-[:amizade]-(n:Perfil) RETURN n.nome;`
    - `MATCH (Perfil{nome:"Karla"})-[:amizade*2..3]-(n:Perfil) WHERE (n.cidade = "Florianopolis") RETURN n.nome LIMIT 100;`

Funções de Agregação

- Comando `MATCH`
  - `MATCH (p:Perfil) RETURN count(p.foto);`
  - `MATCH (p:Perfil{nome:"Luma"})-[t]->()`
    `RETURN type(t), count(*);`
  - `MATCH (p:Perfil) RETURN avg(p.idade);`
  - `MATCH (p:Post) RETURN min(p.ID), max(p.ID);`
  - `MATCH (p:Perfil) RETURN sum(p.idade);`

Exclusão de Dados

- Comando **DELETE**
  - exclui nodos e arestas
  - `MATCH (p:Perfil{nome:"Karla"}) DELETE p;`
    `/* um nodo com arestas não é excluído */`
  - `MATCH (p:Perfil{nome:"Karla"}) DETACH DELETE p;` *`/* exclui o nodo e todas as suas arestas */`*
  - `MATCH (p:Perfil{nome:"Karla"})-[r:posse]->() DELETE r;`
- Comando **REMOVE**
  - exclui propriedades e rótulos de nodos e arestas
  - `MATCH (p:Perfil{nome:"Karla"}) REMOVE p.foto;`
  - `MATCH (p {nome:"Karla"}) REMOVE p:Perfil;`
  - `MATCH (p:Perfil{nome:"Karla"})-[r:like]->() REMOVE r.timestamp;`

Alteração de Dados

- Comando **SET**
  - altera ou cria propriedades e rótulos para nodos e arestas já existentes
  - `MATCH (p {nome:"Karla"}) SET p.cidade = "Lages";`
  - `MATCH (p {nome:"Karla"}) SET p:Perfil;`
  - `MATCH (p:Perfil{nome:"Karla"})-[r:comentario]->()) WHERE (r.timestamp:"4/4/2023 15:00") SET r.texto = "muito bom!";`
  - `MATCH (p {nome:"Karla"}) SET p = {name: 'Karla Nayumi', email:'karla@gmail.com'}`
    `/* substitui todas as propriedades */`

Exercícios

- Interface gráfica do Neo4j para uso da Cypher
  - *https://console.neo4j.org/*

- Criar uma rede social com os seguintes perfis:
  - Ronaldo, r.mello@ufsc.br, 55, Porto Alegre, 999999999
  - Luma, luma@gmail.com, 30, Florianopolis, 888888888
  - Karla, karla@gmail.com, 27, Florianopolis, 777777777
  - Lucas, lucas@gmail.com, 35, Lages, 666666666

- Criar as seguintes relações de amizade (em ambos os sentidos):
  - Ronaldo e Luma, 4/4/2023 15:00
  - Ronaldo e Karla, 3/4/3023 10:00
  - Luma e Lucas, 2/4/2023 09:00
  - Luma e Karla, 1/4/2023 14:00

Exercícios

- Criar os seguintes posts da Karla (em ambos os sentidos):
  - ID 100, 1/4/2023 10:00, “Estressada!”
  - ID 101, 3/4/2023 14:30, “Que dia lindo!”

- Criar os seguintes posts do Ronaldo (em ambos os sentidos):
  - ID 102, 1/4/2023 11:00, “Novo mês iniciando!”
  - ID 103, 4/4/2023 14:30, “Estou gripado!”

- Lucas deu um like nos posts 101 e 102 em 4/4/2023 11:30 e 4/4/2023 15:00, respectivamente

- Karla comentou o post 103 em 4/4/2023 16:00: “melhoras!”

Exercícios

1) Buscar todos os posts do BD
2) Buscar os perfis de Florianopolis
3) Buscar os nomes dos amigos de Karla
4) Buscar a idade da pessoa mais velha da rede social
5) Buscar o número de amigos de Ronaldo
6) Buscar os nomes das pessoas que comentaram ou deram like nos posts de Ronaldo
7) Remover os comentários feitos pela Karla
8) Alterar a idade de Lucas para 37 e sua cidade para Florianopolis