

Graph Databases: Discutindo o Relacionamento dos seus Dados com Python

Nicolle Cysneiros



Quem sou eu?

- › Fullstack Developer @ Labcodes
 - › Django
 - › AngularJS
- › PUG-PE
- › PyLadies Recife

Labcodes

- Software studio de **Recife** para o mundo
- Python, Django, Javascript, React, Postgres
- Labcodes ❤️ Comunidade



Roteiro

- › Relacionamentos
- › Banco de Dados em Grafo
- › Neo4J
- › Neo4J vs Banco Relacional
- › Aplicações

Relacionamentos

Relacionamentos



Representando relacionamentos

CENÁRIO

Uma rede social onde um usuário pode ser amigo de outro usuário or curtir uma página, funcionando de forma similar ao Facebook.

Representando relacionamentos

BANCO DE DADOS RELACIONAL

User			
ID	Name	Gender	Age
1	John	M	28
2	Mary	F	26
3	Francis	F	31

Friends_with	
ID_1	ID_2
1	2
2	3

Representando relacionamentos

BANCO DE DADOS RELACIONAL

Page		
ID	Nome	Categoria
1	Coca-cola	Food/Beverage
2	The Beatles	Musician/Band
3	Apple	Technology

Likes	
ID_User	ID_Page
1	2
3	2
1	1
2	3

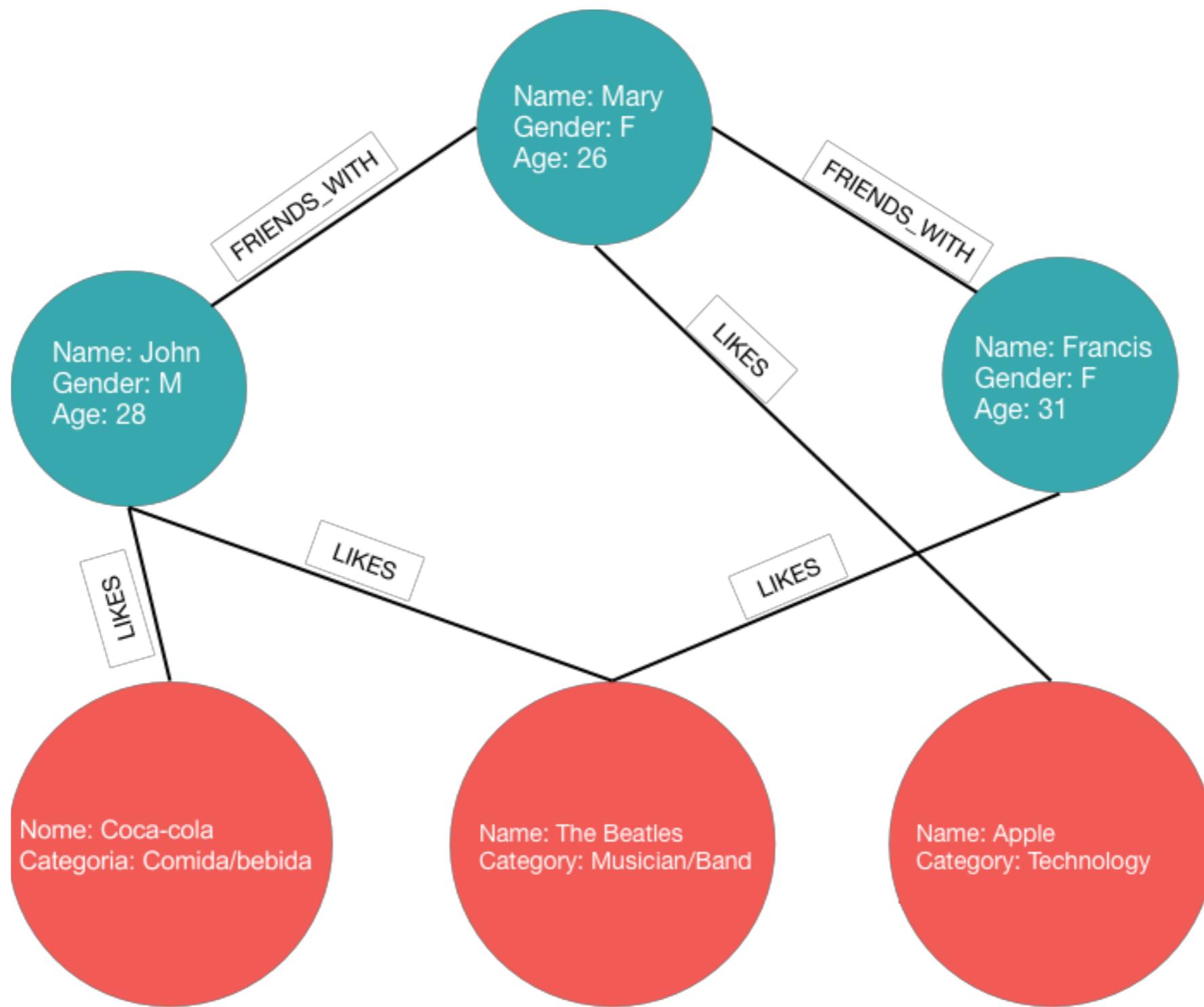
Representando relacionamentos

GRAFOS

$$G = (V, E)$$

Grafo é uma estrutura de dados G formada por um conjunto de vértices V e um conjunto de arestas E.

Representando relacionamentos GRAFOS



Banco de Dados em Grafo

Banco de Dados em Grafo

É um sistema que armazena dados em uma estrutura de grafo, o que permite que o relacionamento entre os dados seja armazenado de forma mais explícita.

Relacionamento Explícito



Recuperação Direta da Informação

Banco de Dados em Grafo

VANTAGENS

Permitem uma análise de dados mais elaborada, usando algoritmos conhecidos da Teoria dos Grafos.

Detecção de Comunidade

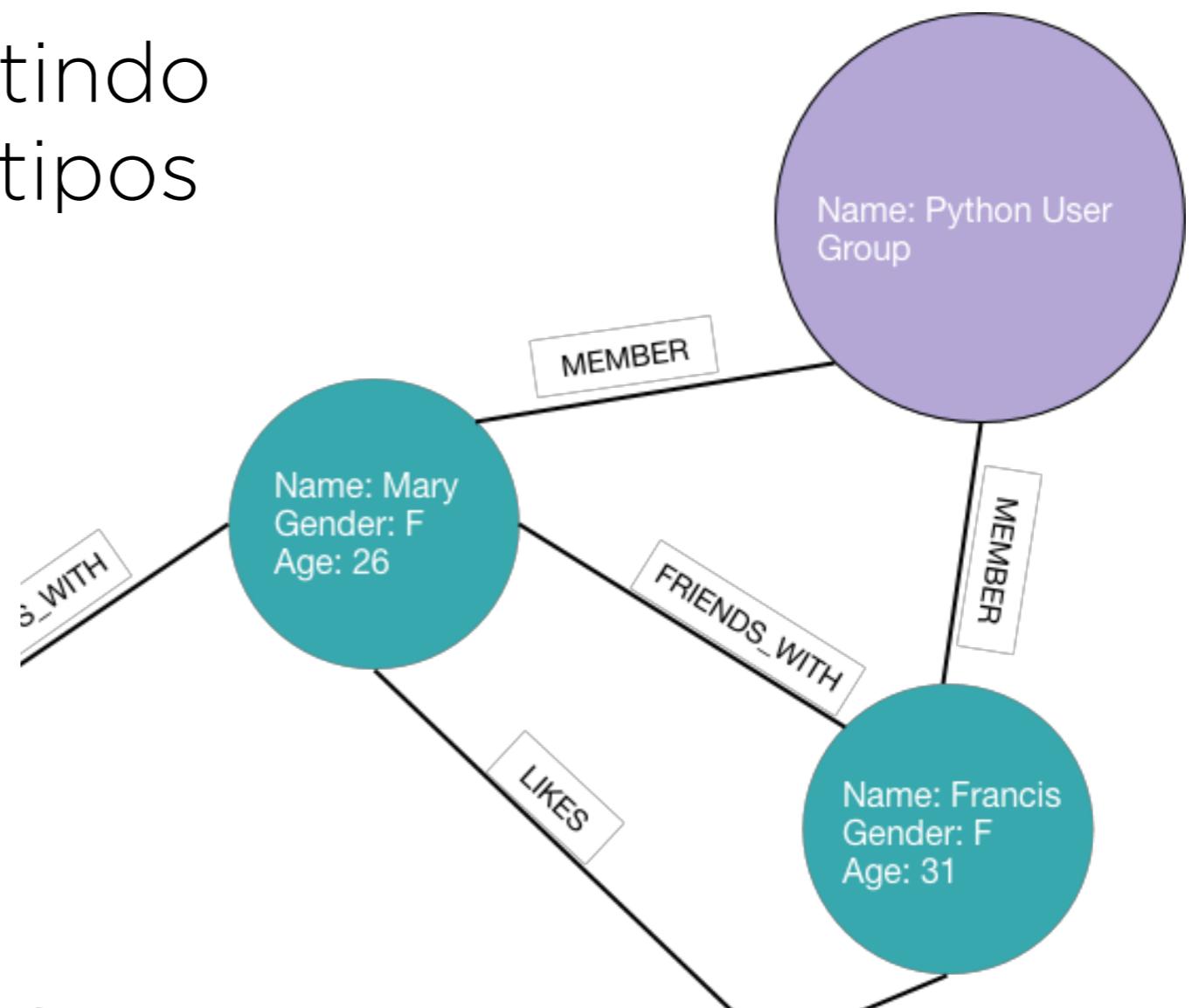
Reconhecimento de Padrões

Medidas de Centralidade

Banco de Dados em Grafo

VANTAGENS

Possuem modelo de dados flexível, permitindo a inserção de novos tipos no banco existente



Banco de Dados em Grafo

VANTAGENS

Apresentam melhor performance por usar mecanismos de armazenamento NoSQL.

Escalabilidade Horizontal

Processamento Distribuído

Neo4J

Neo4J

- Banco de Dados em Grafo mais popular de acordo com o DB Engines
- Implementado em Java
- Tem a própria linguagem de consulta: Cypher
- Dados podem ser acessados através da API REST ou da API Java

Neo4J

EXAMPLES - CREATE VERTEX (NODE)

```
CREATE (john:User {name:"John", gender:"M", age:"28"})  
RETURN john
```

Neo4J

EXAMPLES - CREATE VERTEX (NODE): Result

The screenshot shows the Neo4j browser interface. At the top, a query is displayed:

```
$ CREATE (john:User {name:'John', gender:'M', age:28}) RETURN john
```

Below the query, the results are shown in three tabs: **Graph**, **Rows**, and **Text**. The **Graph** tab is selected, showing a single blue circular node labeled "John". The **Rows** tab shows a single row with one item, and the **Text** tab shows the query text. At the bottom of the results area, it says "Displaying 1 node, 0 relationships." To the right of the results, there are icons for saving, sharing, and deleting, and a close button. At the very bottom right, there is an "AUTO-COMPLETE" toggle switch set to "ON".

```
{ "results": [
{ ...
  "graph": {
    "nodes": [
      {
        "id": "1",
        "labels": [
          "User"
        ],
        "properties": {
          "gender": "M",
          "name": "John",
          "age": "28"
        }
      }
    ],
    "relationships": [ ]
  }
} ],
...
}
```

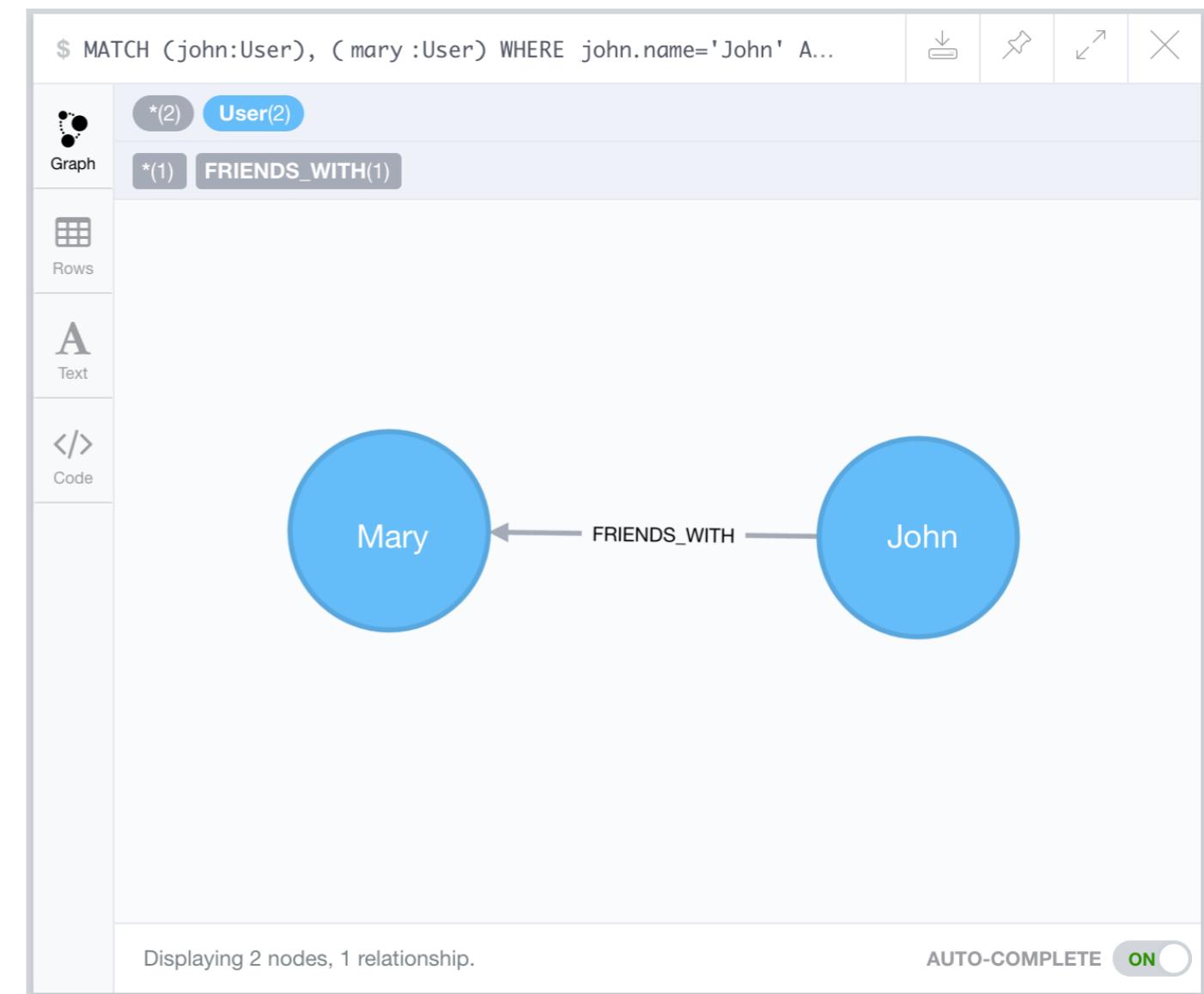
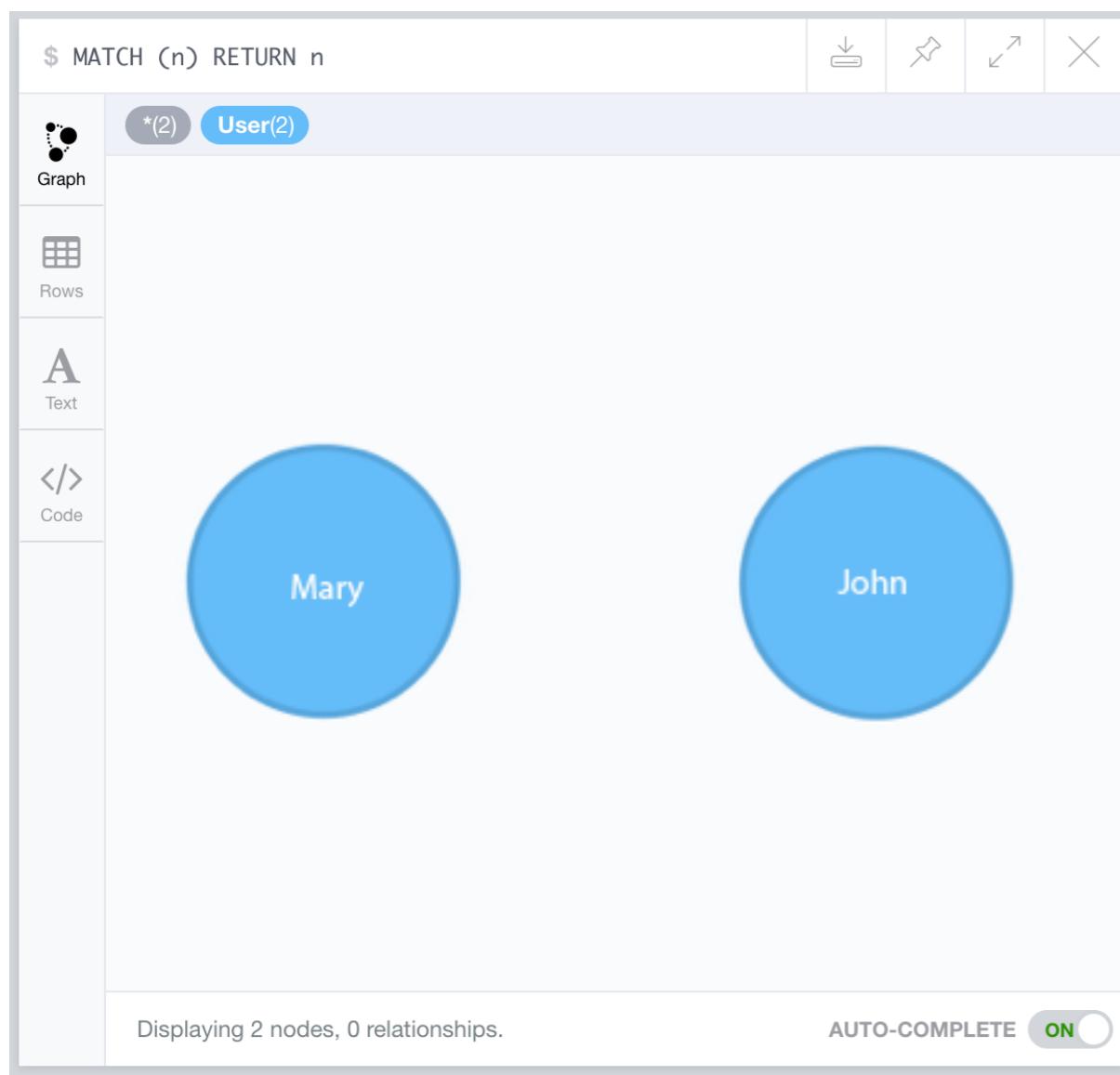
Neo4J

EXAMPLE - CREATE RELATIONSHIP

```
MATCH (john:User), (mary:User)
WHERE john.name="John" AND mary.name="Mary"
CREATE (john)-[f:FRIENDS_WITH]->(mary)
RETURN john, f, mary
```

Neo4J

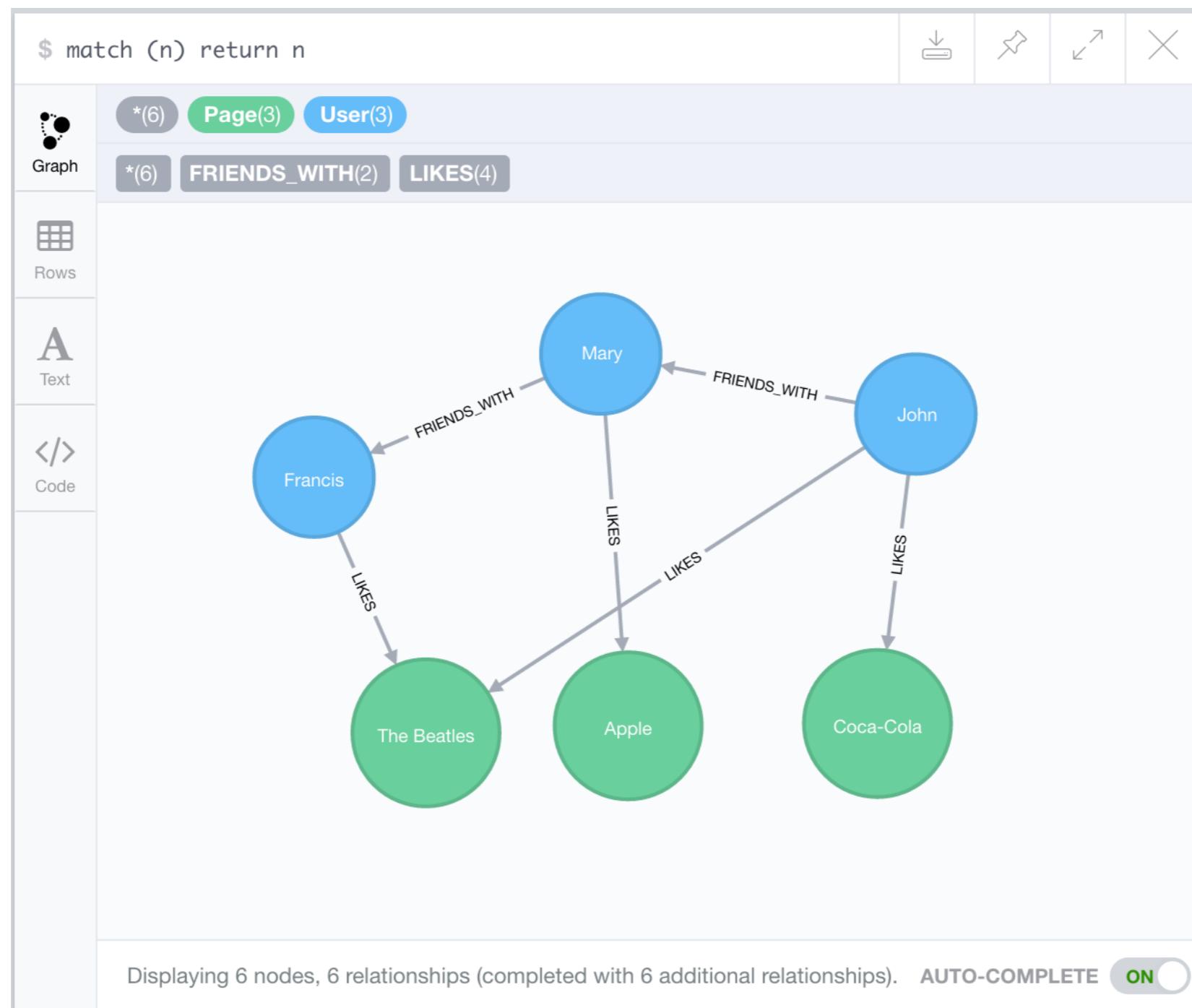
EXAMPLES - CREATE RELATIONSHIP: Result



```
{ "results": [
{...
  "graph": {
    "nodes": [
      { "id": "1",
        "labels": [ "User" ],
        "properties": {
          "gender": "M",
          "name": "John",
          "age": "28"
        } },
      { "id": "2",
        "labels": [ "User" ],
        "properties": {
          "gender": "F",
          "name": "Mary",
          "age": "26"
        } }
    ],
    "relationships": [
      { "id": "0",
        "type": "FRIENDS_WITH",
        "startNode": "1",
        "endNode": "2",
        "properties": {} }
    ]
  } } ],
...
}
```

Neo4J

EXAMPLES - COMPLETE DATABASE



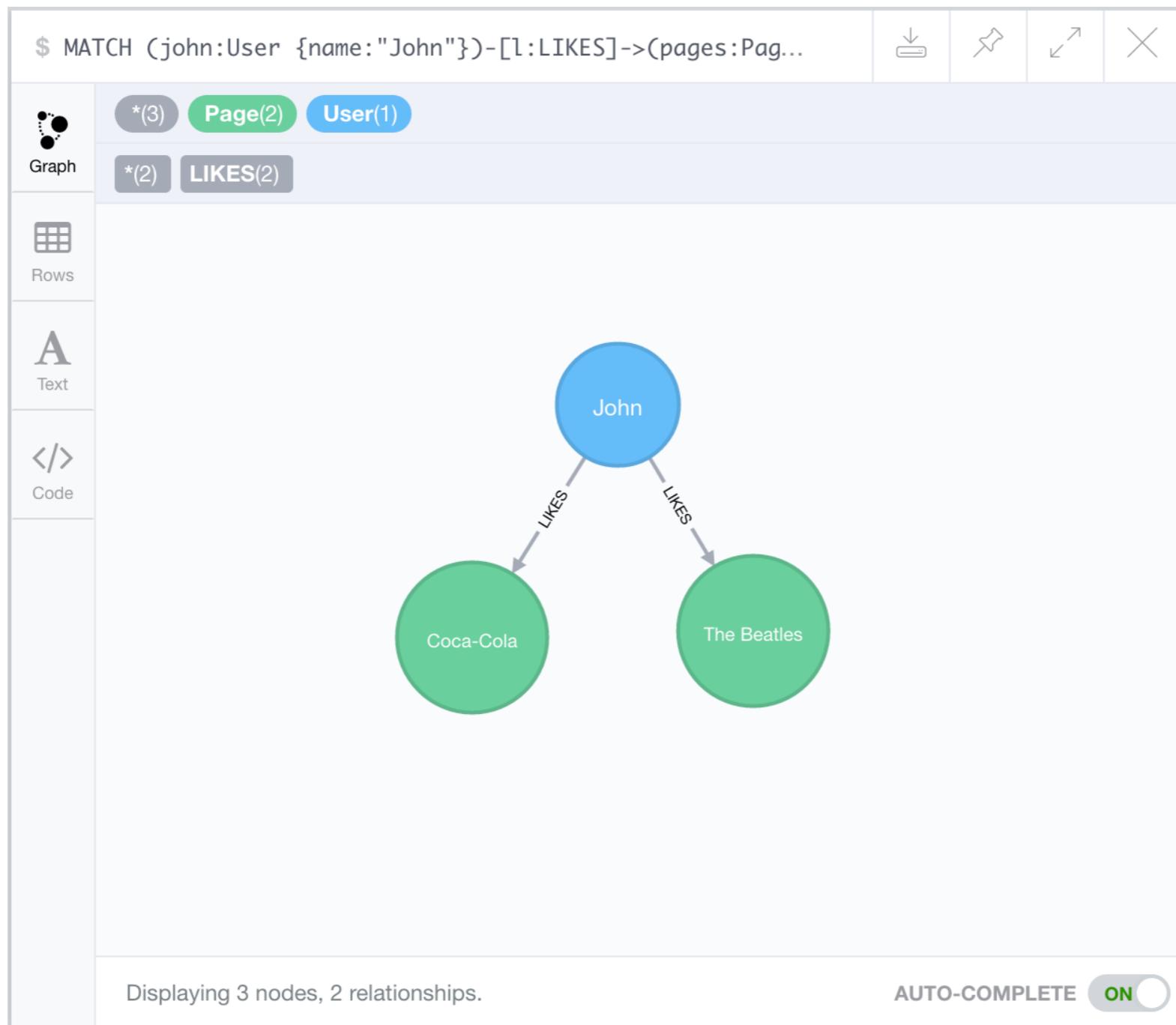
Neo4J

EXAMPLES - QUERY

```
MATCH (john:User {name:"John"})-[l:LIKES]->(pages:Page)
RETURN john, l, pages
```

Neo4J

EXAMPLES - QUERY: Result



```
{ "results": [ {  
    ...  
  "data": [  
    {"row": [  
      { "gender": "M",  
        "name": "John",  
        "age": "28" },  
      {},  
      { "name": "The Beatles",  
        "category": "Musician/Band" }  
    ],  
    ...  
  },  
  {"row": [  
    { "gender": "M",  
      "name": "John",  
      "age": "28" },  
    {},  
    { "name": "Coca-Cola",  
      "category": "Food/Beverage" }  
  ],  
  ... } ],  
  ... }
```

Neo4J + Python

Py2Neo

Módulo Python para trabalhar com Neo4J a partir de aplicações ou de linha de comando. Suporta Python 2 e 3 e garante suporte para distribuições CPython.



Neo4J + Python

CRIAR VÉRTICES (NODES) E RELACIONAMENTOS

```
from py2neo import Graph, Node, Relationship

g = Graph(password="admin")

tx = g.begin()
joao = Node("User", name="João", gender="M", age="28")
tx.create(joao)

maria = Node("User", name="Maria", gender="F", age="26")
tx.create(maria)

jm = Relationship(joao, "FRIENDS_WITH", maria)
tx.create(jm)

tx.commit()
```

Neo4J + Python

CONSULTAR

```
from py2neo import Graph, Node, Relationship, NodeSelector

g = Graph(password="admin")
selector = NodeSelector(g)
joao = selector.select("User", name="João").first()

joao_likes = g.match(start_node=joao, rel_type="LIKES")

for like in joao_likes:
    page = like.end_node()
    print(page)

>>>
(e0f611c:Page {category:"Músico/Banda",name:"The Beatles"})
(ac6964f:Page {category:"Comida/Bebida",name:"Coca-Cola"})
```

Neo4J vs Banco Relacional

Neo4J vs Banco Relacional

Cypher - Neo4J

```
MATCH (User {name: "João"}) -[:LIKES]-> (pages)
RETURN pages
```

SQL

```
SELECT p.nome, p.categoria
  FROM Usuario u, Curte c, Pagina p
 WHERE u.nome = "João"
   AND u.ID = c.ID_Usuario
   AND c.ID_Pagina = p.ID
```

Aplicações

Aplicações

Existem muitas áreas onde a utilização de Banco de Dados em Grafos se torna importante, como:

Redes Sociais

Bioinformática

Análise Genética

Telecomunicações

12 de agosto de 2017

COMO A CÂMARA CONTRIBUI INDIRETAMENTE COM AS CAMPANHAS DOS DEPUTADOS? APENAS UMA ESPECULAÇÃO UTILIZANDO GRAFOS.

| “Saindo do azul e entrando no negro” – Neil Young

Vamos direto a um exemplo que ilustra os resultados obtidos nesta análise. O Deputado Federal Weliton Prado, do Partido da Mulher Brasileira, utiliza regularmente os serviços da empresa Sempre Editora LTDA. Entre 2013 e 2017 foram

Leonardo Sales

Cota Parlamentar x Doação

- › Cotas Parlamentares de 2013 a 2017
- › Doações para a Campanha Eleitoral de 2014
- › Dados disponíveis nos portais da transparência da **Câmara** e do **Tribunal Superior Eleitoral**
- › Python + Networkx

Cota Parlamentar x Doação

RESULTADOS

Ciclo Financeiro Direto



Cota Parlamentar x Doação

USANDO NEO4J

Vamos codar?

Cota Parlamentar x Doação

USANDO NEO4J

Código original e apresentação disponíveis em:

<https://github.com/nicysneiros/workshop-neo4j>

Dúvidas?

www.labcodes.com.br

speakerdeck.com/labcodes

github.com/nicysneiros

twitter.com/nicysneiros

