INTRODUÇÃO A BANCOS DE DADOS ORIENTADOS A GRAFOS

Rafael Sampaio Tavares @rafaelkillua

1

MOTIVAÇÃO

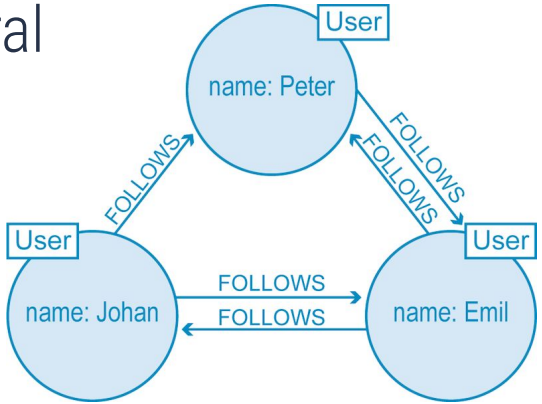
O que são bancos de dados orientados a grafos e por que usá-los?

PROBLEMAS

- Dados muito complexos e altamente conectados
- RDBMS: lentos e difíceis
- Além de dados brutos, **relacionamentos**

SOLUÇÃO

- GRAFOS!!!
- 300+ anos de estudos e conhecimento
- Google, Twitter e Facebook: proprietários
- **Neo4j**: propósito geral



ATENÇÃO

- Existem GDBs que usam tabelas, outros RDBs ou documentos para guardar os dados
- Amazon Neptune, GraphDB, Microsoft Azure: documentos e relacionamentos diretos
- OrientDB, Neo4j: grafos

CONCEITO

- Grafo é uma coleção de vértices (nós ou pontos) e arestas (arcos ou linhas)
- Arestas podem ter **direções** (neste caso, devem)
- Mais próximo dos dados reais

MOTIVAÇÃO

- Então por que utilizar DBs orientados a grafos?
  - ACID ao invés de BASE
  - Performance
  - Flexibilidade
  - Agilidade

MOTIVAÇÃO

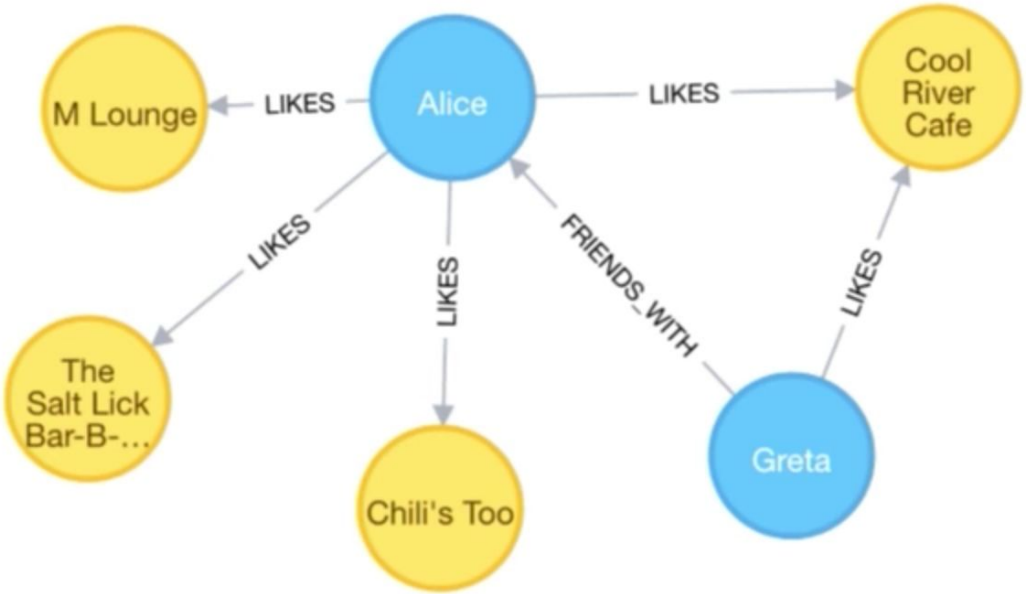
- Além de...
  - Armazenamento e processamento nativos de grafos
  - Adjacência livre de índices

CASOS DE USO MAIS COMUNS

- Sistema de recomendações
- Master Data Management (MDM)
- Detecção de fraudes
- Pesquisa baseada em grafos
- Redes e operações de TI
- Identidade e gerenciamento de acesso

EXEMPLOS

- Sistema de recomendações



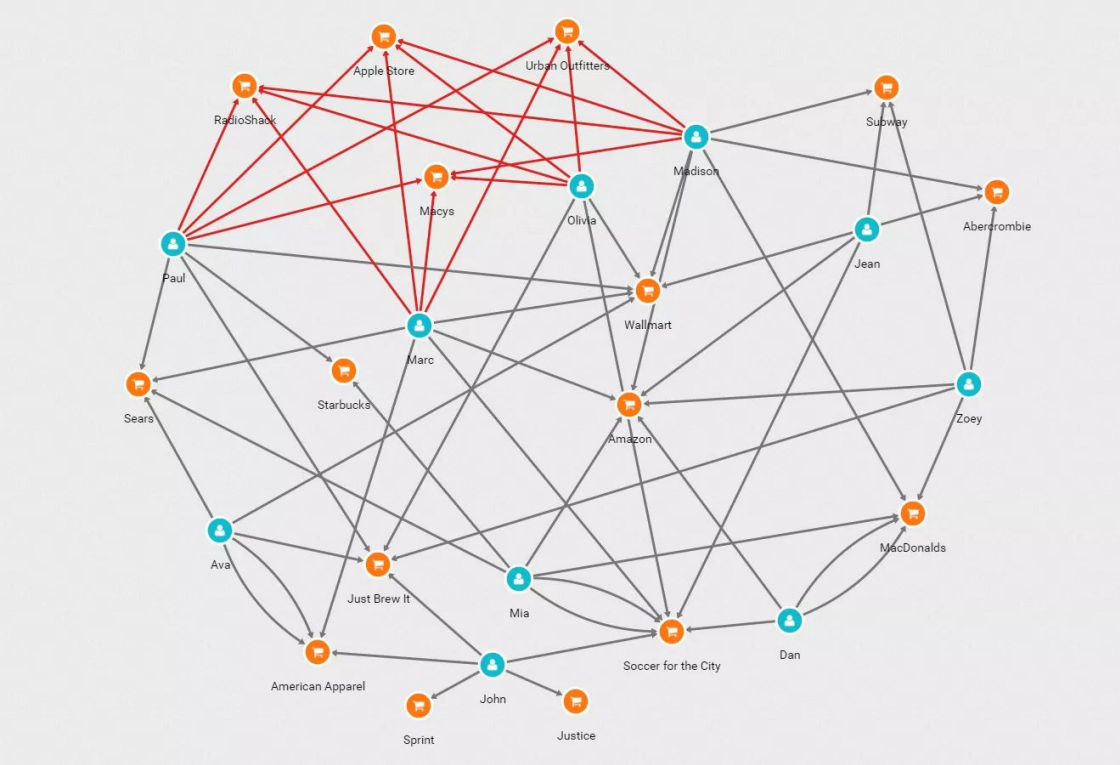
EXEMPLOS

- Sistema de recomendações: Walmart Brazil
- Recomendações estáticas e queries demoradas
- “*We could build a simple and real-time recommendation system with low latency queries*”

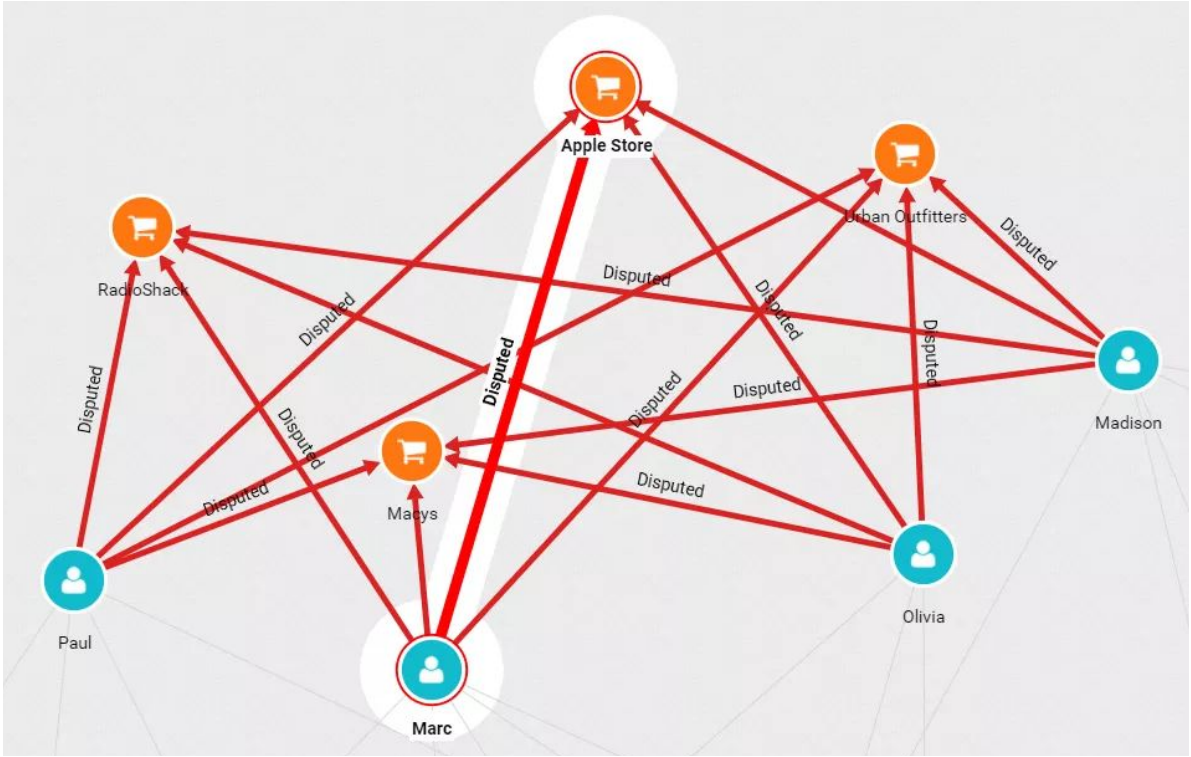
EXEMPLOS

- Detecção de fraudes: Empresa de serviços financeiros no Fortune 500 que movimenta mais de $2.2M por mês
- “*It was taking five minutes or more to run a query*”
- Precisavam de escalabilidade infinita, facilidade de visualização e dados em tempo real
- Além de permitir isso, Neo4j facilitou a visualização de relacionamentos fradulentos mais facilmente e cortou tempo de análise na metade

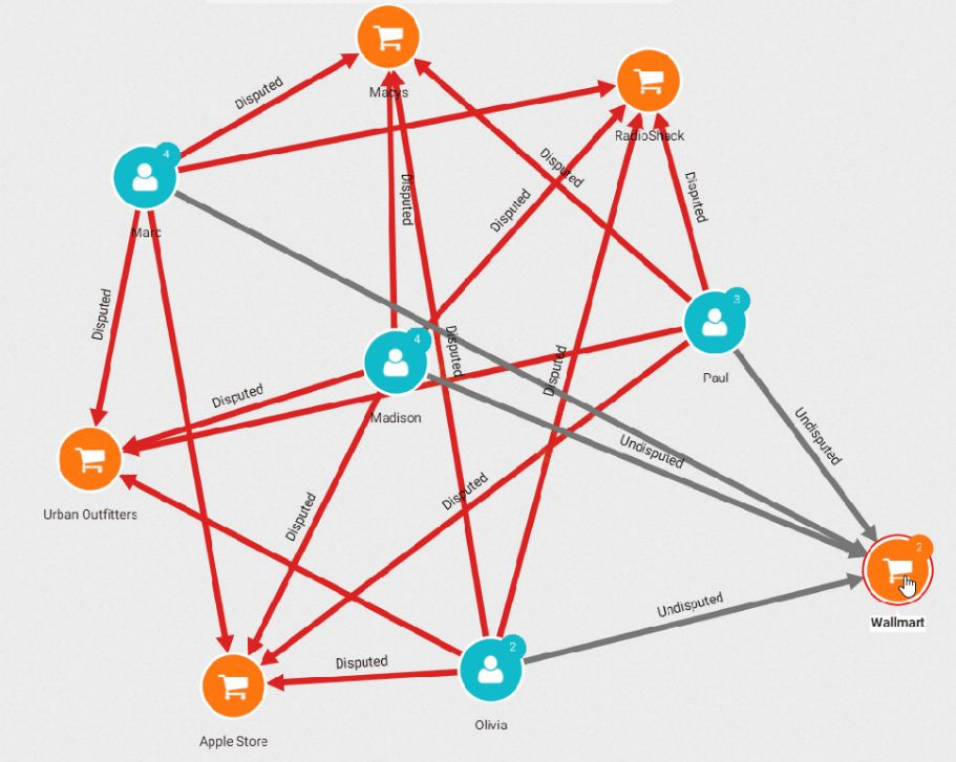
EXEMPLOS


Apple Store
Urban Outfitters
RadioShack
Subway
Madison
Macys
Olivia
Abercrombie
Jean
Paul
Wallmart
Marc
Starbucks
Sears
Amazon
Zoey
MacDonalds
Ava
Just Brew It
Mia
Dan
Soccer for the City
American Apparel
John
Sprint
Justice

EXEMPLOS


Apple Store
Urban Outfitters
RadioShack
Macys
Madison
Paul
Olivia
Marc
Disputed

EXEMPLOS


Macys
RadioShack
Marc
Paul
Madison
Urban Outfitters
Wallmart
Apple Store
Olivia
Disputed
Undisputed

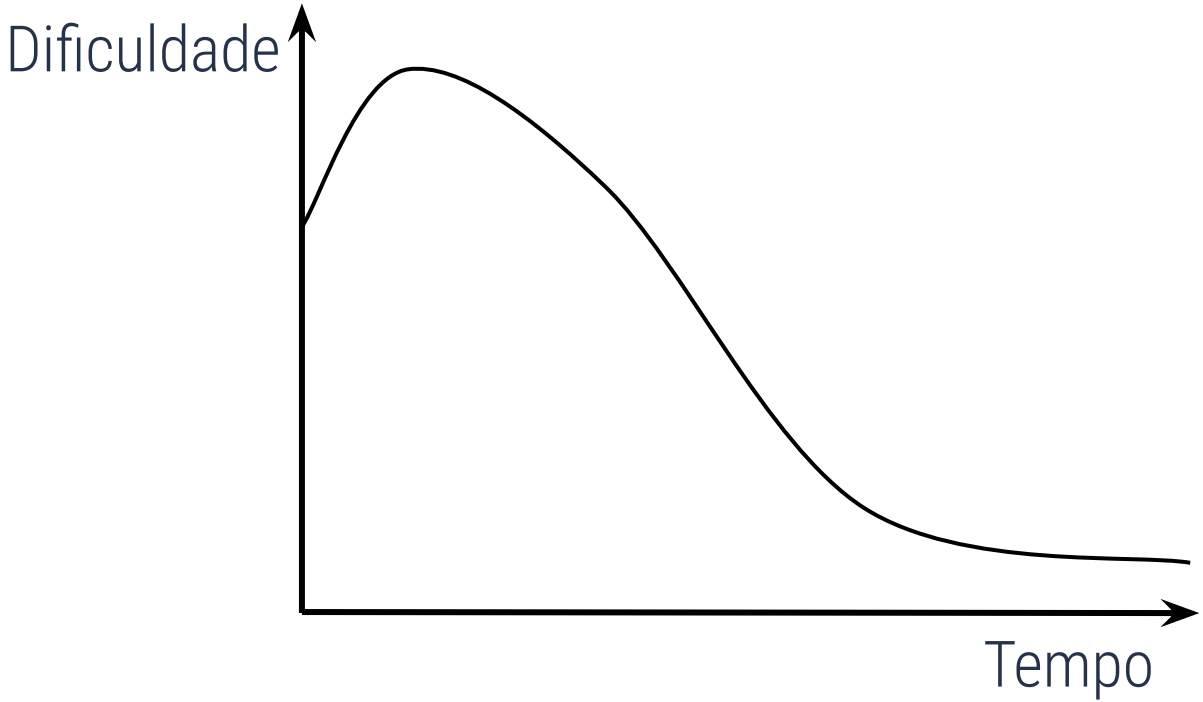
2

PRIMEIROS PASSOS

CONHECIMENTO NECESSÁRIO

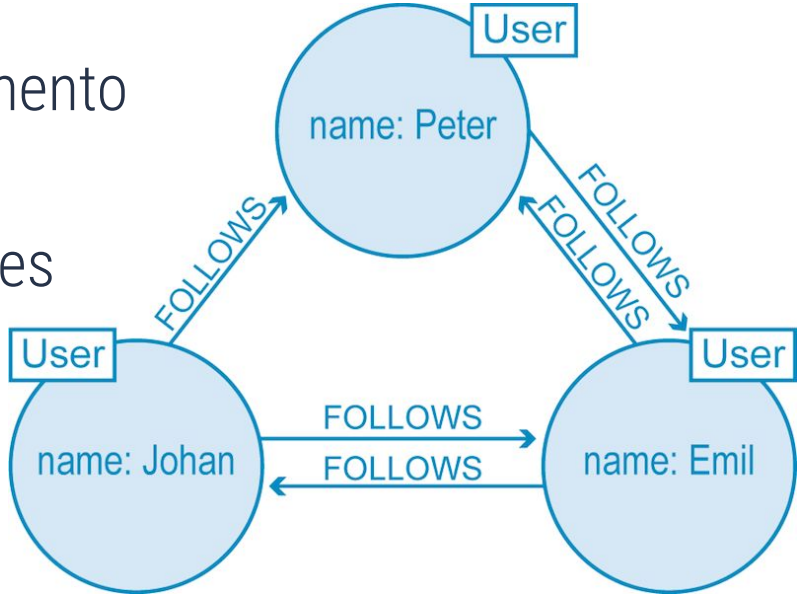
- Entender o que é um grafo é essencial...
- … e basicamente é só isso

CURVA DE APRENDIZADO

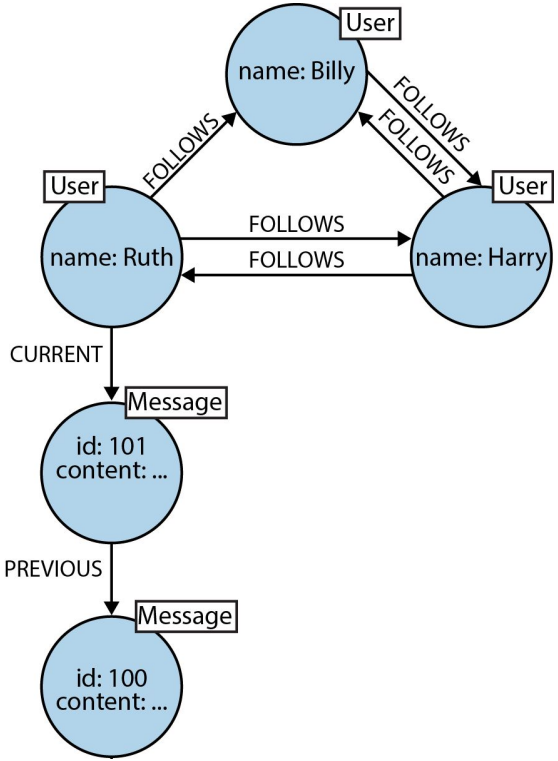


EXEMPLOS BÁSICOS

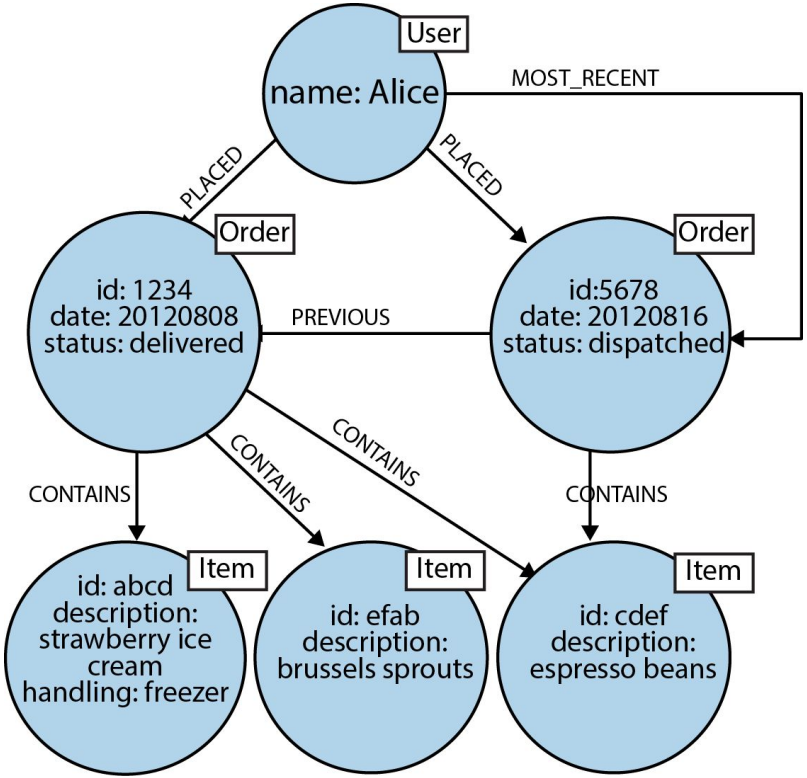
- Labeled Property Graph Model
  - Nó
  - Relacionamento
  - Rótulo
  - Propriedades



EXEMPLOS BÁSICOS



EXEMPLOS BÁSICOS



3

MODELAGEM

MODELAGEM

- RDBs não têm relacionamentos
  - Chave estrangeira (FK)
- Índices

MODELAGEM

User					
UserID	**User**	**Address**	**Phone**	**Email**	**Alternate**
1	Alice	123 Foo St.	12345678	alice@example.org	alice@neo4j.org
2	Bob	456 Bar Ave.		bob@example.org	
...	...	...	...	...	...
99	Zach	99 South St.		zach@example.org	

Order	
OrderID	**UserID**
1234	1
5678	1
...	...
5588	99

LineItem		
OrderID	**ProductID**	**Quantity**
1234	765	2
1234	987	1
...	...	...
5588	765	1

Product		
ProductID	**Description**	**Handling**
321	strawberry ice cream	freezer
765	potatoes	
...	...	
987	dried spaghetti	

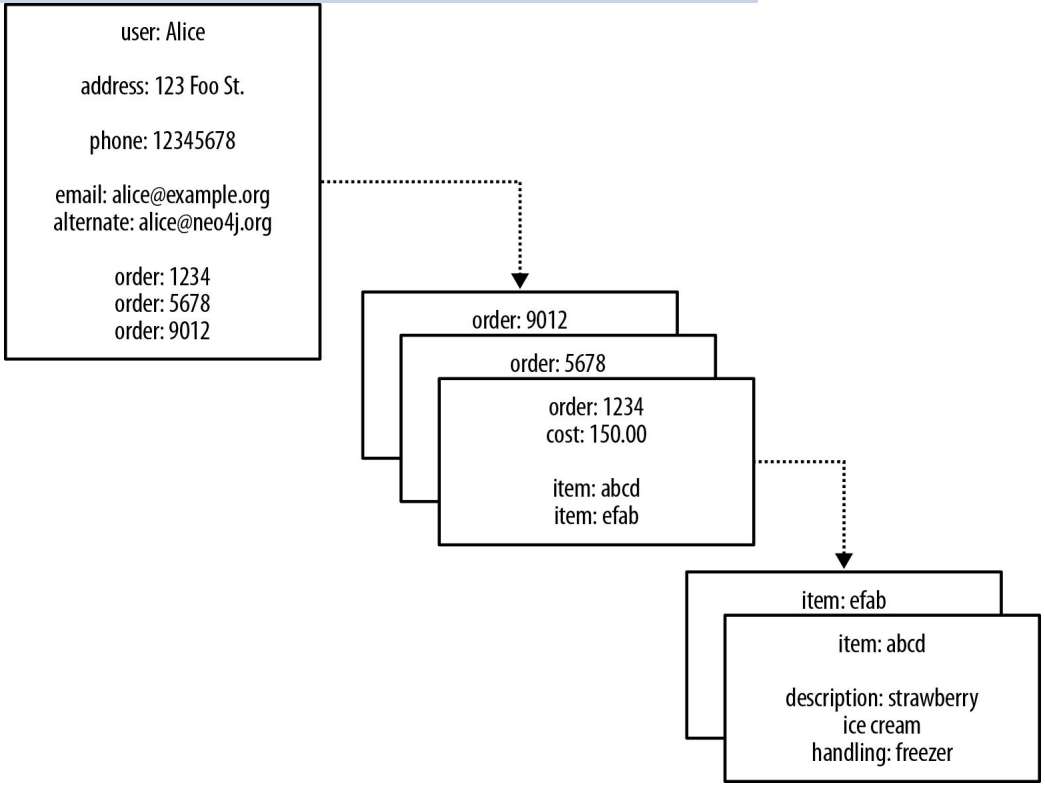
MODELAGEM

- Grande dataset -> vários JOINs
- Queries recíprocas custam mais ainda
- Desnormalização
- Como responder perguntas não pensadas no ato da modelagem?

MODELAGEM

- NOSQL DBs também não têm relacionamentos
  - Referências

MODELAGEM


user: Alice
address: 123 Foo St.
phone: 12345678
email: alice@example.org
alternate: alice@neo4j.org
order: 1234
order: 5678
order: 9012
order: 9012
order: 5678
order: 1234
cost: 150.00
item: abcd
item: efab
item: efab
item: abcd
description: strawberry
ice cream
handling: freezer

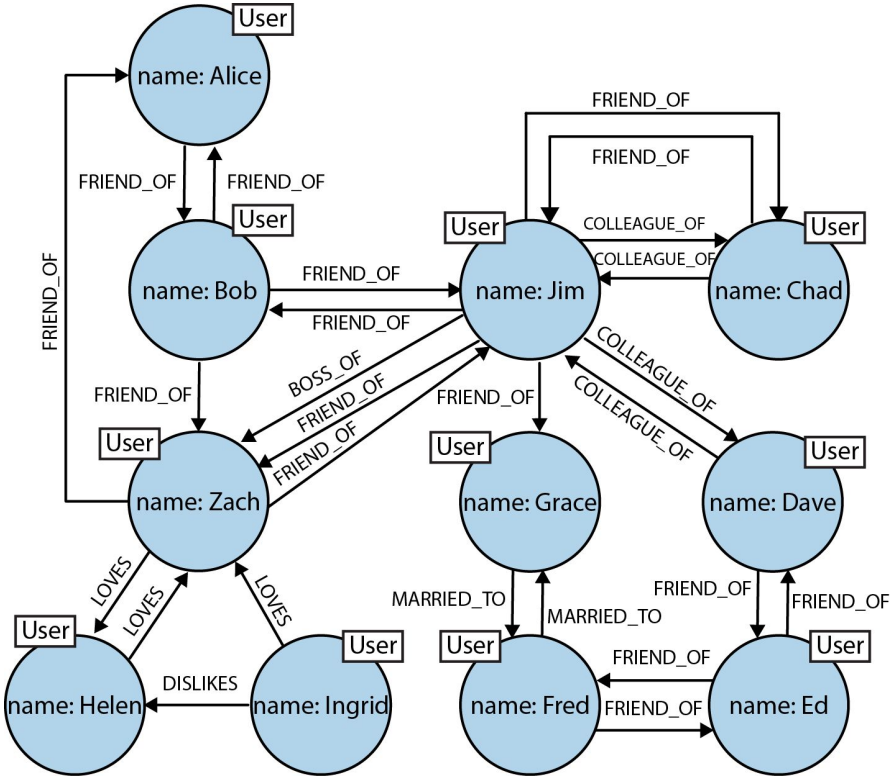
MODELAGEM

- Aggregate não custa pouco para muitas referências
- Aplicação tem que lidar com updates ou deletes nos dados referenciados
- Referências “voltando”
  - Adiciona custo de espaço e de tempo de escrita

MODELAGEM

- Relacionamentos em GDBs existem fisicamente e podem ser facilmente acessados e **atravessados**
- Modelos lógicos e físicos são mais próximos da realidade

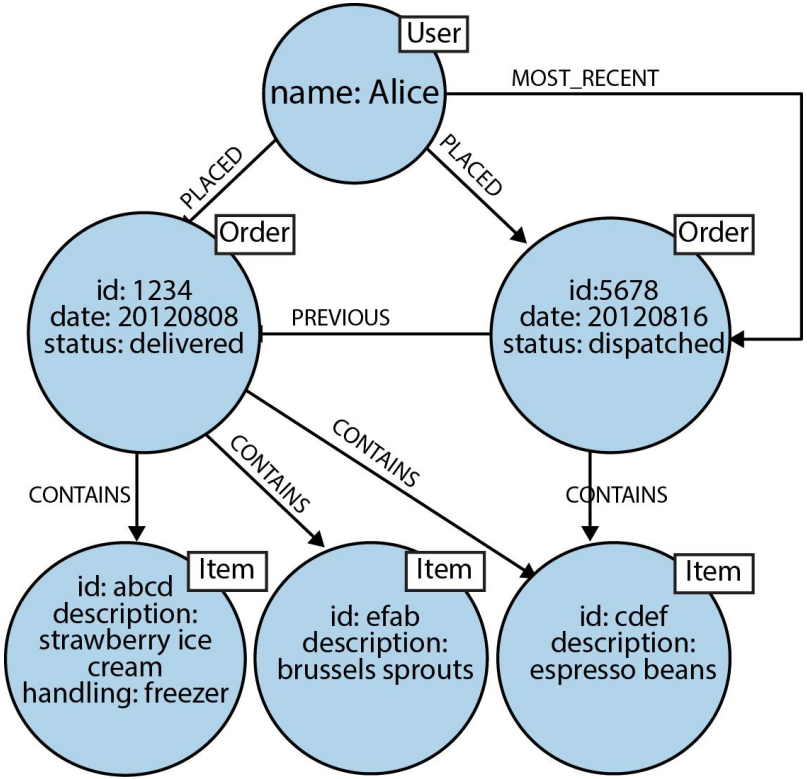
MODELAGEM


User
name: Alice
FRIEND_OF
FRIEND_OF
User
name: Bob
FRIEND_OF
FRIEND_OF
FRIEND_OF
User
name: Jim
FRIEND_OF
COLLEAGUE_OF
COLLEAGUE_OF
User
name: Chad
BOSS_OF
FRIEND_OF
FRIEND_OF
FRIEND_OF
COLLEAGUE_OF
COLLEAGUE_OF
User
name: Zach
User
name: Grace
User
name: Dave
LOVES
LOVES
LOVES
MARRIED_TO
MARRIED_TO
FRIEND_OF
FRIEND_OF
User
name: Helen
DISLIKES
User
name: Ingrid
User
name: Fred
FRIEND_OF
FRIEND_OF
User
name: Ed

MODELAGEM

Depth	RDBMS execution time(s)	Neo4j execution time(s)	Records returned
2	0.016	0.01	~2500
3	30.267	0.168	~110,000
4	1543.505	1.359	~600,000
5	Unfinished	2.132	~800,000

MODELAGEM


User
name: Alice
MOST_RECENT
PLACED
PLACED
Order
id: 1234
date: 20120808
status: delivered
PREVIOUS
Order
id:5678
date: 20120816
status: dispatched
CONTAINS
CONTAINS
CONTAINS
CONTAINS
Item
id: abcd
description:
strawberry ice
cream
handling: freezer
Item
id: efab
description:
brussels sprouts
Item
id: cdef
description:
espresso beans

4

CYPHER

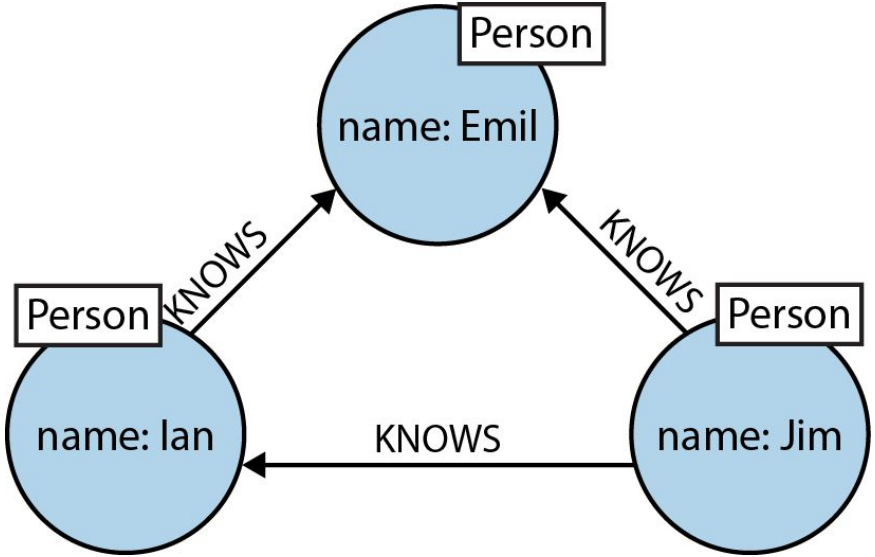
DEFINIÇÃO

- **Cypher:** linguagem de query
  - Criada pro Neo4j
  - OpenCypher Project
- Apache: Gremlin e SPARQL

CLÁUSULAS

- **MATCH**
- CREATE [UNIQUE]
- SET
- DELETE
- MERGE
- FOREACH
- UNION
- WITH
- START

DEFINIÇÃO




```
(emil)<-[:KNOWS]-(jim)-[:KNOWS]->(ian)-[:KNOWS]->(emil)
```

DEFINIÇÃO

```
(emil:Person {name: 'Emil'})
<-[:KNOWS]-(jim:Person {name: 'Jim'})
-[:KNOWS]->(ian:Person {name: 'Ian'})
-[:KNOWS]->(emil)
```

CREATE

```
CREATE
(emil:Person {name: 'Emil'}),
(jim:Person {name: 'Jim'}),
(ian:Person {name: 'Ian'}),
(emil)<-[:KNOWS]-(jim),
(jim)-[:KNOWS]->(ian),
(ian)-[:KNOWS]->(emil)
```

CREATE

```
CREATE
(emil:Person {name: 'Emil'})
<-[:KNOWS]-(jim:Person {name: 'Jim'})
-[:KNOWS]->(ian:Person {name: 'Ian'}),
(ian)-[:KNOWS]->(emil)
```

MATCH

```
MATCH
(a:Person)-[:KNOWS]->(b)-[:KNOWS]->(c),
(a)-[:KNOWS]->(c)
WHERE a.name = 'Jim'
RETURN b, c
```

SET

```
MATCH
(a:Person)
WHERE a.name = 'Jim'
SET a.surname = 'Halpert'
RETURN a
```

SET LABEL

```
MATCH
(a:Person)
WHERE a.name = 'Jim'
SET a:Student
```

EXPRESSÕES

- Operadores
  - `DISTINCT, STARTS WITH, ENDS WITH, CONTAINS, IS NULL, IN, AND, OR, XOR, NOT`
- Temporal
  - `DATE, TIME, DATETIME, LOCALTIME, LOCALDATETIME, DURATION`
- Geográficos
  - `DISTANCE(p1, p2)` - Pode ser 2D ou 3D

INDEXES E CONSTRAINTS

- `CREATE INDEX ON :Person(email)`
- `CREATE CONSTRAINT ON (o:Order) ASSERT o.total > 0`

Próximos passos

- Exemplo básico Neo4j com Express. No README há links diversos sobre desenvolvimento com Neo4j:
- https://github.com/rafaelkillua/express-neo4j-example
- Apresentação feita na Paguru, com codificação:
- https://www.youtube.com/watch?v=gFB7G51aGXM

NA PRÁTICA

- Partiu fazer na prática?

CRÉDITOS

- https://neo4j.com/graph-databases-book/
- https://www.youtube.com/watch?v=AsnXiGQ_Hi4
- https://bcc.ime.usp.br/tccs/2016/taksqth/downloads/monografia.pdf
- https://db-engines.com/en/system/MongoDB%3BNeo4j
- https://www.youtube.com/watch?v=U8ZGVx1NmQg
- https://www.youtube.com/watch?v=GekQqFZm7mA

VALEU!!!