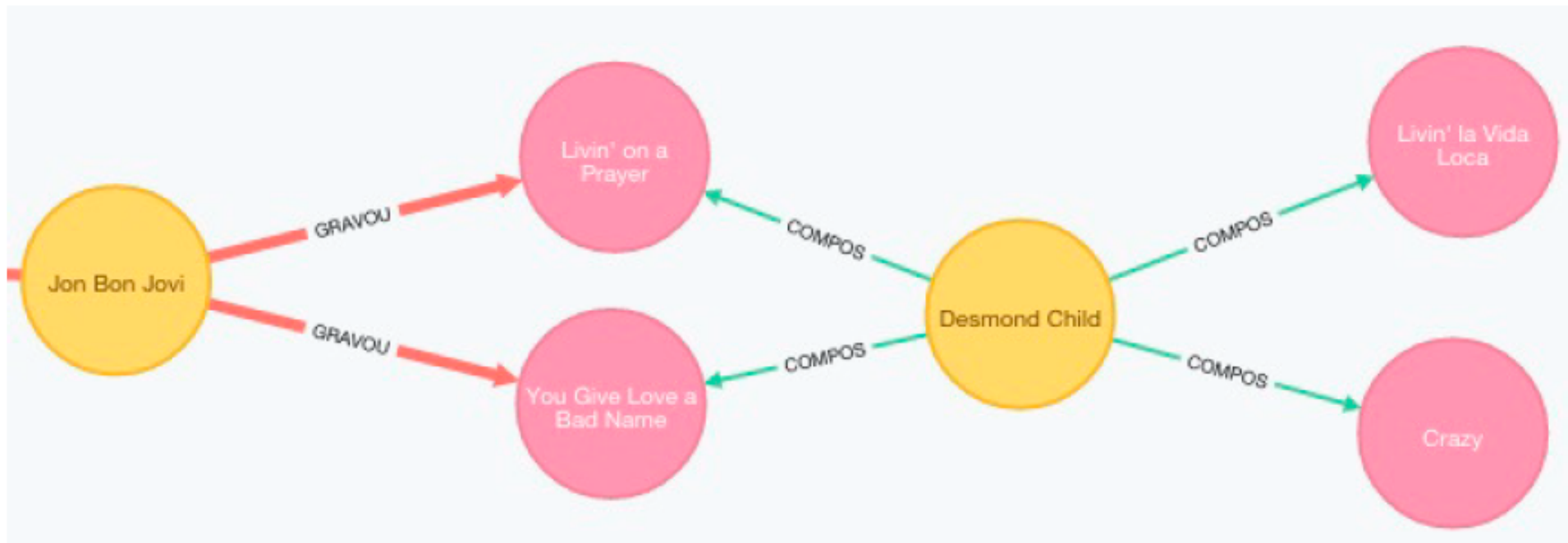




Profa. Andrêza Leite

Modelo de dados

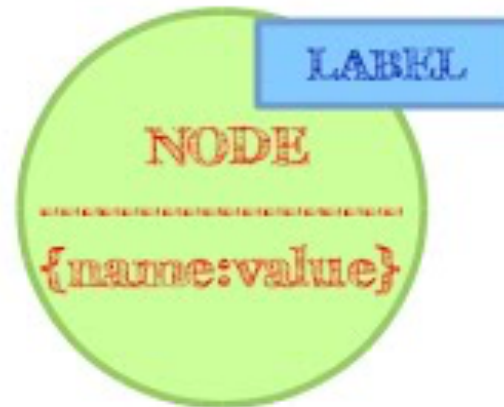
- Grafos – nós(entidades) e relacionamentos podem ter atributos/propriedades



Modelo de dados

NODE + PROPERTY

=



Características e Recursos

- Consistência
 - Dentro de um único servidor os dados são consistentes: Neo4J é compatível com ACID.
 - Os nós estarão disponíveis para leitura. Gravações nos escravos são permitidas mas aguardam sincronia com o mestre, que propaga para os demais escravos.
 - Garantem consistência por meio de transações.
 - Não permitem relacionamentos pendentes: os dois nós tem de existir e só podem ser excluídos se não houver relacionamentos.

Características e Recursos

- Transações
 - Se não envolver operações em transações: `NotInTransactionException`
 - Operações de leitura podem executar sem iniciar uma transação.
 - Uma transação precisa ser marcada como bem sucedida senão o Neo4J supõe que ela falhou: `transaction.success()`;
 - `transaction.finish()`; - necessário para confirmar os dados no BD

Características e Recursos

- Disponibilidade
 - Obtém alta disponibilidade ao permitir escravos replicados
 - Gravações são sincronizadas (e confirmadas) com o mestre depois nos demais escravos
 - Outros como Infinite Graph e FlockDB fornecem armazenamento distribuídos para os nodos.
 - Neo4J usa o Zookeeper para registrar Ids da última transação persistida em cada nodo escravo e em cada mestre atual.

Características e Recursos

- Escalabilidade
 - Usar RAM para carregar os nós e relacionamentos que estão sendo trabalhados.
 - Melhorar leitura adicionando mais escravos apenas leitura.
 - Útil quando o conjunto de dados não couber na RAM.
 - Fragmentar a partir do lado do app utilizando conhecimento específico do domínio.
 - Ex: nós que relacionam com a América ser criados em um servidor e os que relacionam com a Ásia em outro.
 - BDs fisicamente diferentes

Casos Apropriados

- Dados conectados
 - Redes sociais e qualquer domínio rico em links
- Roteamento, envio e serviços baseados em localização
 - Local ou endereço considerado um nó e pode ser modelado na forma de grafo → entrega mais eficiente
 - Grafos de lugares de interesse para recomendações
- Mecanismos de recomendação
 - “Seus amigos também compraram este produto” ou “ao faturar este item estes geralmente também são ” ou recomendar lugares próximos visitados a viajantes

Casos Inapropriados

- Atualizar todas as entidades
 - Atualizar propriedades em todos os nós não é uma opção direta
- Alguns BDs podem não conseguir lidar com muitos dados
 - Operações globais = envolvendo o grafo inteiro

Prática

Iniciando Neo4j

```
$ neo4j start
```

```
http://localhost:7474/
```

Criando o 1º nó

```
$ CREATE(dylan:Musico {nome : 'Bob  
Dylan', data_de_nascimento : '1941-05-24'})
```

Added 1 label, created 1 node, set 2
properties, statement executed in 671 ms.

Visualizando o 1º nó

The screenshot displays the Neo4j Browser interface. On the left sidebar, the 'Node labels' section has 'Musico' selected and circled in red. Below it, the 'Relationship types' and 'Property keys' sections are visible. The 'Database' section at the bottom shows the location and size of the database. The main area on the right shows a Cypher query: `S MATCH (n:Musico) RETURN n LIMIT 25`. Below the query, the 'Graph' view is selected, and a single yellow node labeled 'Bob Dylan' is visible. A red circle highlights the 'Table' icon in the left sidebar, indicating the next step to view the query results in a tabular format.

Criando relacionamentos

```
CREATE (no1)-[:TIPO_RELACIONAMENTO]->(no2)
```

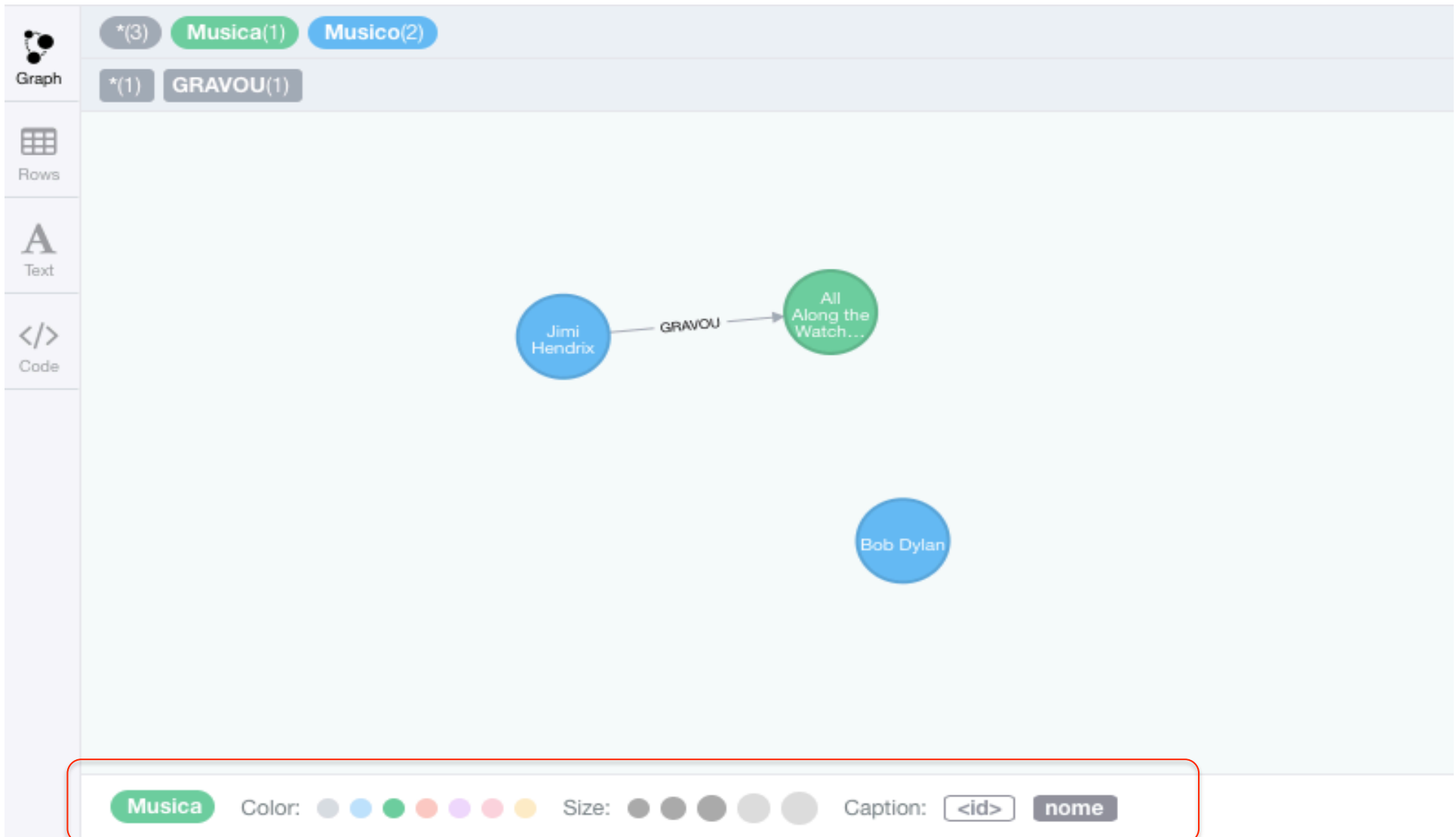
```
//criando mais nós
```

```
$ CREATE(all_along:Musica {nome: "All Along the  
Watchtower"})
```

```
$ CREATE(hendrix:Musico {nome: "Jimi Hendrix"})
```

```
//criando o relacionamento gravou
```

```
$ CREATE (hendrix)-[:GRAVOU]->(all_along)
```



Fazendo buscas

- **Cypher** = query language
- Principais palavras : MATCH e RETURN
 - RETURN é parecido ao SELECT , enquanto o MATCH faz o papel do FROM

Fazendo buscas

//listando todos os musicos

MATCH (m:Musico)

RETURN m.nome

//equivalente em sql

SELECT m.nome FROM musicos m

Fazendo buscas

//retornando nós e visualização do grafo

MATCH (m:Musico)

RETURN m

//carregar o grafo todo

MATCH (m)

RETURN m

Filtrando buscas

```
//usando WHERE  
MATCH (m:Musico)  
WHERE m.nome = 'Bob Dylan'  
RETURN m  
  
//dentro do MATCH  
MATCH (m:Musico { nome : 'Bob Dylan' })  
RETURN m
```

Mais relacionamentos

```
MATCH (bob:Musico { nome : 'Bob Dylan' }),  
(all_along:Musica { nome : 'All Along the  
Watchtower'})
```

```
CREATE (bob)-[:GRAVOU]->(all_along)
```

```
CREATE (bob)-[:COMPOS]->(all_along)
```

```
//visualizando o grafo
```

```
MATCH (n) RETURN n
```

Mais buscas

//buscando um nó () com um relacionamento []
com outro nó ()

MATCH (n)-[]-() RETURN n

//visualize em rows

//execute com o count

MATCH (n) RETURN COUNT(n)

Buscas

//o count retorna um valor bem diferente agora

MATCH (n)-[]-() RETURN COUNT(n)

//estamos buscando pelo padrão NO-
relacionado-NO.

Buscas

Jimi Hendrix - GRAVOU - All Along the Watchtower

Bob Dylan - GRAVOU - All Along the Watchtower

Bob Dylan - COMPOS - All Along the Watchtower

All Along the Watchtower - GRAVOU - Jimi Hendrix

All Along the Watchtower - GRAVOU - Bob Dylan

All Along the Watchtower - COMPOS - Bob Dylan

Buscas

//Filtrando pela direção do relacionamento >ou<
MATCH (n)-[]->>() RETURN n

| Jimi Hendrix - GRAVOU - All Along the Watchtower |
| Bob Dylan - GRAVOU - All Along the Watchtower |
| Bob Dylan - COMPOS - All Along the Watchtower |

Buscas

//Filtrando pela direção do relacionamento >ou<
MATCH (n)<-[]-() RETURN n

| All Along the Watchtower - GRAVOU - Jimi Hendrix |
| All Along the Watchtower - GRAVOU - Bob Dylan |
| All Along the Watchtower - COMPOS - Bob Dylan |

Buscas

//dando nome para a visualização ficar parecida

```
MATCH (n)-[r]->(n2) RETURN n, r, n2
```

//como não há atributo no relacionamento ele
retorna vazio mas podemos ver o tipo dele

```
MATCH (n)-[r]->(n2) RETURN n, type(r), n2
```

Buscas

//filtrando pelo tipo de nó

```
MATCH (n:Musica)-[r]-(n2:Musico) RETURN type(r)
```

//filtrando pelos relacionamentos

```
MATCH (n:Musico)-[r:COMPOS]-(n2:Musica) RETURN n
```

```
MATCH (n:Musico)-[r:GRAVOU]-(n2:Musica) RETURN n
```

Editando

//adicionando data de nascimento

MATCH (hendrix: Musico { nome : 'Jimi Hendrix' })

SET hendrix.data_de_nascimento =
'1942-11-27'

RETURN hendrix

Editando

//para remover um atributo set null

```
MATCH (hendrix:Musico { nome : 'Jimi  
Hendrix' })
```

```
SET hendrix.data_de_nascimento = null
```

```
RETURN hendrix
```

Apagando

```
MATCH (hendrix:Musico { nome : 'Jimi  
Hendrix' })
```

```
DELETE hendrix
```

//necessário apagar relacionamentos antes

```
MATCH (hendrix:Musico { nome : 'Jimi  
Hendrix' })-[rel]-()
```

```
DELETE rel
```

Apagando

//apagando nó e relacionamento

MATCH (dylan:Musico {nome : 'Bob Dylan' })-[rel]-()

DELETE rel, dylan

//para apagar tudo

//nó pode ou não ter relacionamento

MATCH (n) OPTIONAL MATCH (n)-[rel]-()

DELETE rel, n

Importando dados <csv>

//arquivo de composições

Bob Dylan,All Along the Watchtower

Bob Dylan,It Ain't Me, Babe

Conhecendo o Merge

Sintaxe: MERGE (compositor)-[COMPOS]-> (musica)

//testando o merge

MERGE (n1:Musico {nome: "Bob Dylan"})

MERGE (n2:Musico {nome: "Bob Dylan"})

RETURN id(n1), id(n2)

Added **1 label**, created **1 node**, set 1 property, statement executed in 25 ms.

OBS: cria nó ou relacionamento apenas se não existir

Carregando o arquivo

LOAD CSV WITH HEADERS

FROM "file:/arquivo.csv"

AS linha

RETURN linha

//vamos visualizar os arquivos

Arquivo gravações

interprete,musica

"Jimi Hendrix","All Along the Watchtower"

"Johnny Cash","It Ain't Me, Babe"

"Jack White","One More Cup of Coffee"

"George Harrison","If Not For You"

"Joey Ramone","My Back Pages"

"Jon Bon Jovi","Knockin' on Heavens"

"Steve Tyler","Crazy"

"Ricky Martin","Livin' la Vida Loca"

"Jon Bon Jovi","Livin' on a Prayer"

"Jon Bon Jovi","You Give Love a Bad Name"

Arquivo composições

compositor,musica

"Bob Dylan","All Along the Watchtower"

"Bob Dylan","It Ain't Me Babe"

"Bob Dylan","One More Cup of Coffee"

"Bob Dylan","If Not For You"

"Bob Dylan","My Back Pages"

"Bob Dylan","Knockin' on Heavens"

"Desmond Child","Crazy"

"Desmond Child","Livin' la Vida Loca"

"Desmond Child","Livin' on a Prayer"

"Desmond Child","You Give Love a Bad Name"

Inserindo dados do arquivo

LOAD CSV WITH HEADERS

FROM "file:/composicoes.csv"

AS linhaCsv

MERGE (compositor:Musico {nome:
linhaCsv.compositor})

MERGE (musica:Musica {nome: linhaCsv.musica})

MERGE (compositor)-[:COMPOS]->(musica)

Inserindo dados do arquivo

LOAD CSV WITH HEADERS

FROM "file:/gravacoes.csv"

AS linhaCsv

MERGE (interprete:Musico {nome:
linhaCsv.interprete})

MERGE (musica:Musica {nome: linhaCsv.musica})

MERGE (interprete)-[:GRAVOU]->(musica)

Buscando

“quem compôs as músicas que cada músico gravou”

- procurar um músico com um relacionamento do tipo COMPOS
- MATCH (i:Musico)-[g:GRAVOU]->(m:Musica)<-[e:COMPOS]-(c:Musico)
- *"Intérprete que gravou uma música escrita por um compositor"*

Buscando

```
MATCH (i:Musico)-[g:GRAVOU]->(m:Musica)<-  
[e:COMPOS]-(c:Musico)
```

```
RETURN i.nome, m.nome, c.nome
```

//mesmo resultado com dois matches

```
MATCH (i:Musico)-[g:GRAVOU]->(m:Musica)
```

```
MATCH (m:Musica)<-[e:COMPOS]-(c:Musico)
```

```
RETURN i, m, c
```


Aggregação

//quantas músicas cada intérprete gravou de cada compositor

MATCH (interprete:Musico)-[gravou:GRAVOU]-
>(musica:Musica)

MATCH (compositor:Musico)-[compos:COMPOS]-
>(musica:Musica)

RETURN interprete.nome, COUNT(musica),
compositor.nome

Queries complexas

// Buscando por Jimi Hendrix

MATCH (interprete:Musico)-[gravou:GRAVOU]-
>(musica:Musica)

MATCH (compositor:Musico)-[compos:COMPOS]-
>(musica:Musica)

WHERE interprete.nome = "Jimi Hendrix"

RETURN interprete.nome, COUNT(musica),
compositor.nome

Queries complexas

// Buscando por Jon Bon Jovi

MATCH (interprete:Musico)-[gravou:GRAVOU]-
>(musica:Musica)

MATCH (compositor:Musico)-[compos:COMPOS]-
>(musica:Musica)

WHERE interprete.nome = "Jon Bon Jovi"

RETURN interprete.nome, COUNT(musica),
compositor.nome

Queries complexas

Queremos responder:

"quem gravou músicas escritas por **compositores que escreveram músicas que um determinado músico gravou?**"

- Falta mais um salto no grafo para identificar quem gravou as músicas que os compositores em questão escreveram

Queries complexas

//quem gravou músicas escritas por **compositores que escreveram músicas que Jimi hendrix gravou**

MATCH (interprete:Musico)-[gravou:GRAVOU]->(musica:Musica)

MATCH (compositor:Musico)-[compos:COMPOS]-
>(musica:Musica)

MATCH (compositor:Musico)-[compos:COMPOS]-
>(outraMusica:Musica)

WHERE interprete.nome = "Jimi Hendrix"

RETURN interprete.nome, compositor.nome,
COUNT(outraMusica)

Queries complexas

Na nossa base de dados, temos 10 músicas, sendo 6 escritas pelo Bob Dylan e 4 escritas pelo Desmond Child mas última busca retornou apenas uma música.

- precisamos nos preocupar com o nome que usamos nos relacionamentos.

Queries complexas

```
//renomeando a variável dos relacionamentos "compos"  
MATCH (interprete:Musico)-[gravou:GRAVOU]->(musica:Musica)  
MATCH (compositor:Musico)-[com1:COMPOS]->(musica:Musica)  
MATCH (compositor:Musico)-[com2:COMPOS]->(outraMusica:Musica)  
WHERE interprete.nome = "Jimi Hendrix"  
RETURN interprete.nome, compositor.nome, COUNT(outraMusica)  
//todas as músicas escritas pelos compositores que escreveram as  
músicas que o Jimi Hendrix gravou.
```

Queries complexas

//o mesmo para Bon Jovi

MATCH (interprete:Musico)-[gravou:GRAVOU]->(musica:Musica)

MATCH (compositor:Musico)-[com1:COMPOS]->(musica:Musica)

MATCH (compositor:Musico)-[com2:COMPOS]-
>(outraMusica:Musica)

WHERE interprete.nome = "Jon Bon Jovi"

RETURN interprete.nome, compositor.nome,
COUNT(outraMusica)

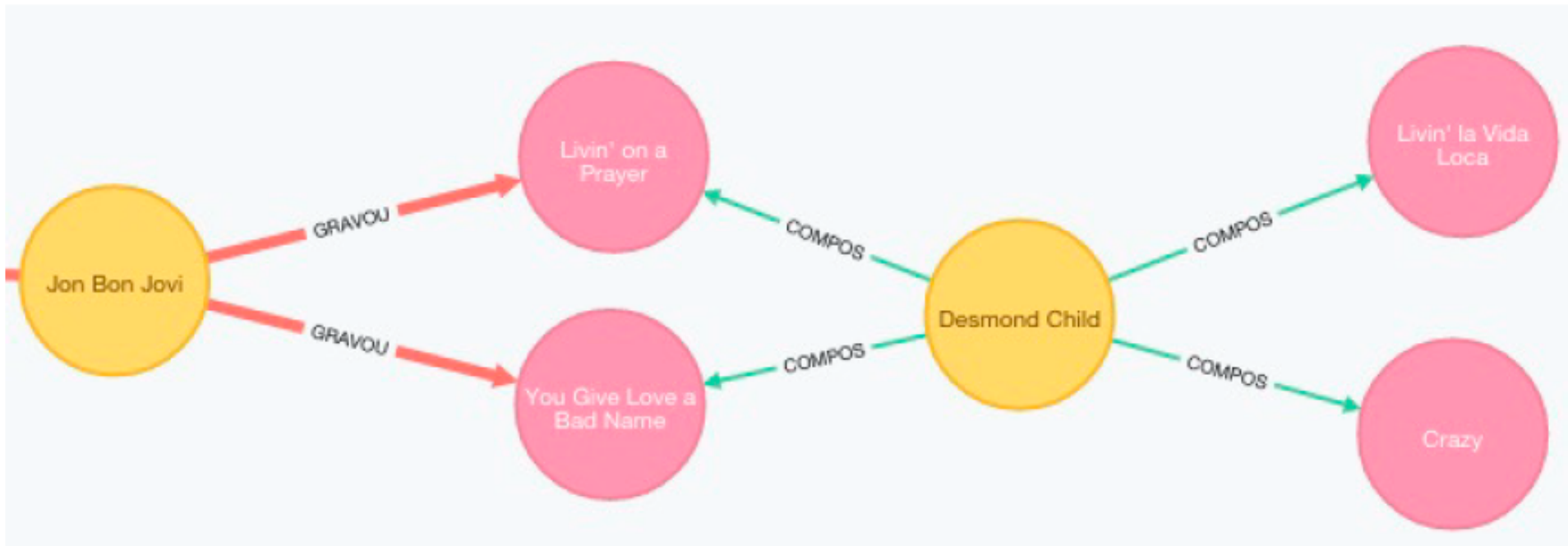
Queries complexas

interprete.nome	compositor.nome	COUNT(outraMusica)
Jon Bon Jovi	Bob Dylan	6
Jon Bon Jovi	Desmond Child	8

- Deveriam ser 4 músicas
- o valor é 2 vezes maior porque nosso intérprete gravou 2 músicas do compositor
- Se formos seguir nó por nó, relacionamento por relacionamento, encontraremos dois caminhos que levam do nó Jon Bon Jovi até os nós de cada uma das músicas escritas pelo Desmond Child

Queries complexas

//analizando o grafo do Bon Jovi até a música Crazy



Queries complexas

//usando o distinct

MATCH (interprete:Musico)-[gravou:GRAVOU]->(musica:Musica)

MATCH (compositor:Musico)-[com1:COMPOS]->(musica:Musica)

MATCH (compositor:Musico)-[com2:COMPOS]-
>(outraMusica:Musica)

WHERE interprete.nome = "Jon Bon Jovi"

RETURN interprete.nome,

compositor.nome,

COUNT(DISTINCT outraMusica)

Queries complexas

//Chegando aos interpretes que as gravaram

MATCH (interprete:Musico)-[gravou:GRAVOU]->(musica:Musica)

MATCH (compositor:Musico)-[com1:COMPOS]->(musica:Musica)

MATCH (compositor:Musico)-[com2:COMPOS]->(outraMusica:Musica)

MATCH (outraMusica:Musica)<-[gravou2:GRAVOU]-
(outroInter:Musico)

WHERE interprete.nome = "Jimi Hendrix"

RETURN interprete.nome, compositor.nome, outroInter.nome

Queries complexas

interprete.nome	compositor.nome	outroInter.nome
Jimi Hendrix	Bob Dylan	Jon Bon Jovi
Jimi Hendrix	Bob Dylan	Joey Ramone
Jimi Hendrix	Bob Dylan	George Harrison
Jimi Hendrix	Bob Dylan	Jack White
Jimi Hendrix	Bob Dylan	Johnny Cash
Jimi Hendrix	Bob Dylan	Jimi Hendrix

É preciso garantir que o outroInterprete seja de fato outro intérprete.

Queries complexas

//adicionando outra clausula com AND e <>

MATCH (interprete:Musico)-[gravou:GRAVOU]->(musica:Musica)

MATCH (compositor:Musico)-[com1:COMPOS]->(musica:Musica)

MATCH (compositor:Musico)-[com2:COMPOS]->(outraMusica:Musica)

MATCH (outraMusica:Musica)<-[gravou2:GRAVOU]-
(outroInter:Musico)

WHERE interprete.nome = "Jimi Hendrix"

AND interprete <> outroInter

RETURN interprete.nome, compositor.nome, outroInter.nome

Queries complexas

//ainda há problemas de duplos caminhos e é necessário agregar(count) para distinguir

MATCH (interprete:Musico)-[gravou:GRAVOU]->(musica:Musica)

MATCH (compositor:Musico)-[com1:COMPOS]->(musica:Musica)

MATCH (compositor:Musico)-[com2:COMPOS]->(outraMusica:Musica)

MATCH (outraMusica:Musica)<-[gravou2:GRAVOU]-(outroInter:Musico)

WHERE interprete.nome = "Jon Bon Jovi"

AND interprete <> outroInter

RETURN outroInter.nome AS interprete,

compositor.nome AS compositor,

COUNT(DISTINCT outraMusica) AS total_musicas

ORDER BY compositor.nome

Inserindo Banco Movies

```
LOAD CSV WITH HEADERS FROM "file:/movie_metadata.csv" AS linha
MERGE (movie:Movie{titulo:linha.movie_title})
MERGE (actor:Actor{nome_ator:coalesce(linha.actor_1_name,"Unknown")})
MERGE (director:Director{nome_diretor:coalesce(linha.director_name,"Unknown")})
MERGE (genre:Genre{genero:linha.genres})
MERGE (year:Year{ano:coalesce(linha.title_year,"Unknown")})
MERGE (language:Language{linguagem:coalesce(linha.language,"Unknown")})
MERGE (score:Score{pontuacao:linha.imdb_score})
MERGE (duration:Duration{duracao:coalesce(linha.duration,"Unknown")})
MERGE (actor)-[:ATUOU]->(movie)
MERGE (director)-[:DIRIGIU]->(movie)
MERGE (movie)-[:POSSUI]->(genre)
MERGE (movie)-[:LANCOU]->(year)
MERGE (movie)-[:TEM]->(language)
MERGE (movie)-[:PONTUOU]->(score)
MERGE (movie)-[:DURA]->(duration)
```