

Bases de Dados

Módulo 22: Introdução a banco de dados noSQL

Prof. André Bruno de Oliveira

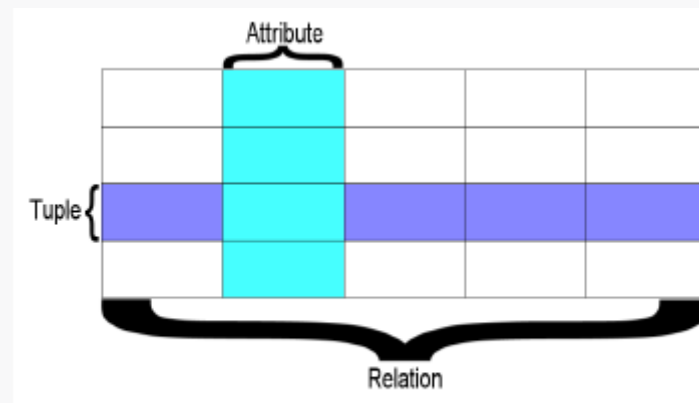
09/06/24 18:55

SGBD relacionais e sua aceitação

- Introdução SGBD relacionais: benefícios e frustrações importante para a visão de um projetista de BD.
- **NoSQL**
 - O que é NoSQL?
- **Teorema de CAP**
- **Computação em cluster**
 - O Que é um cluster?
 - Tipos de cluster
- **Tipos de Banco de dados NoSQL**
 - Chave-valor
 - Documento
 - Colunar
 - Grafo

SGBD relacionais e sua aceitação

- Opção padrão para armazenamento de dados de aplicações corporativas. Costumava ser uma das poucas unanimidades na informática (até pouco tempo atrás...).
- Proporcionam muitos benefícios:
 - Tecnologia madura com grande comunidade de usuários;
 - Embasamento matemático (Teoria Relacional);
 - Linguagem SQL;
 - Esquemas, integridade referencial, concorrência, transações,



FONTE: [https://en.wikipedia.org/wiki/Relation_\(database\)](https://en.wikipedia.org/wiki/Relation_(database))

SGBD relacionais e suas características

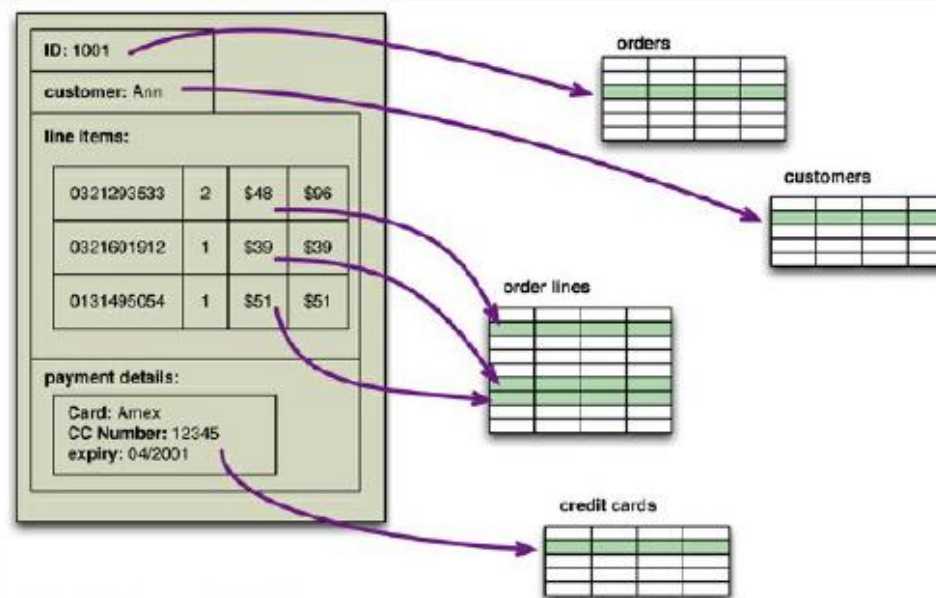
- O banco de dados relacional surgiu como um sucessor dos modelos hierárquicos de rede. Estas estruturas, por sua vez, foram muito utilizadas nos primeiros sistemas de mainframe.
- O Modelo relacional se destaca por suas restrições de integridade para garantir a consistência e pela capacidade de realizar consultas. Sua organização em tabelas é regida pelas regras de normalização que visa evitar redundâncias e inconsistência. Seu processamento de transações inclui ACID (**A**tomicidade, **C**onsistência, **I**solamento, **D**urabilidade).

Limitações do Banco de dados Relacional

- Atualmente o volume de dados de certas organizações chega ao nível de petabytes e continua crescendo. No caso destes tipos de organizações, a utilização dos SGBD relacionais tem se mostrado muito problemática e não tão eficiente.
- Um solução comum é melhorar a capacidade de processamento do servidor, mas chega um ponto que o orçamento para investir no *upgrade* de um servidor que ofereça uma eficiência adequada se torna inviável.
 - Escalabilidade vertical (*scale up*) - adicionar mais memória, processador e disco em um único servidor → custo alto e nem sempre é eficaz.
 - **Proposta tem sido** o escalonamento horizontal (*scale out*): utilizar muitas máquinas em um cluster → mais em conta, mais resiliente, tendência atual. Os SGBD relacionais não foram originalmente projetados para *scale out*, apesar de ser possível.

Limitações do Banco de dados Relacional

- *Impedance Mismatch* (descompasso de impedância)
 - Aplicações modelam dados em memória como objetos e coleções.
 - Essas estruturas precisam ser “traduzidas” para o modelo relacional antes de serem gravadas no BD. Esta transformação requer custo de processamento maior.
 - A proposta tem sido mudar a forma de armazenar os dados, saindo de uma estrutura com um padrão rígido para investir numa solução mais flexível



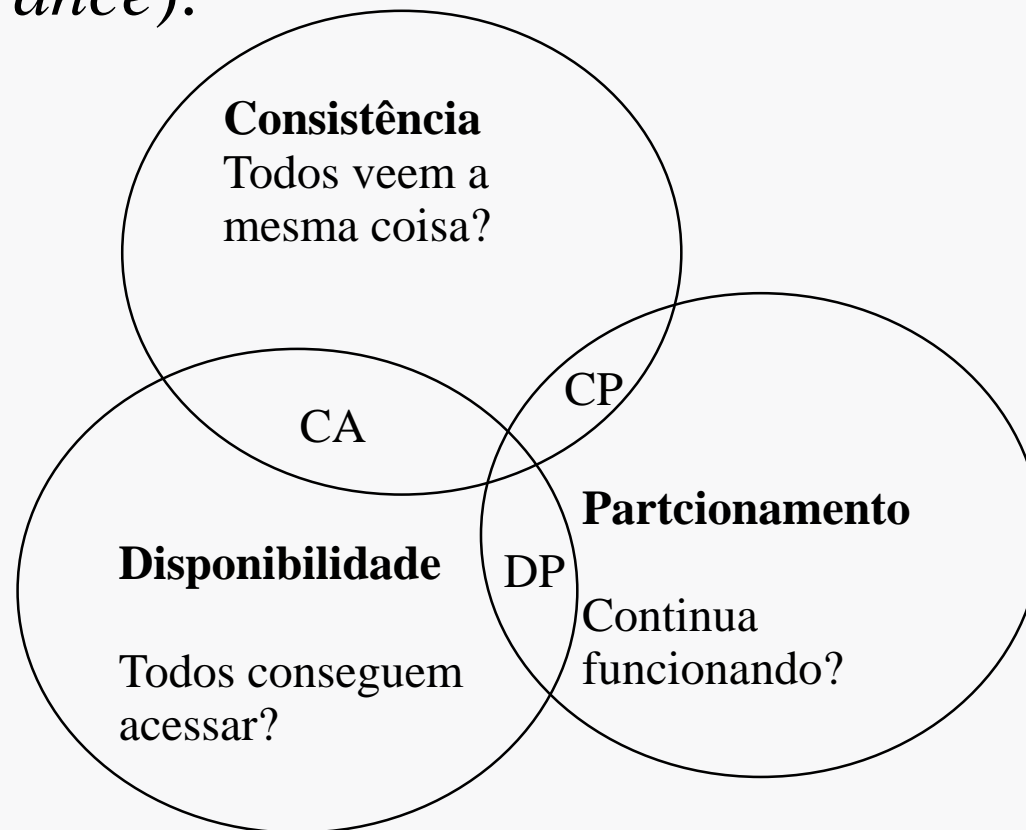
FONTE: Fowler & Sadalage, 2012

Um novo paradigma: BD NoSQL

- O que é NoSQL - Not Only SQL
 - Antes mesmo deste movimento de mudança, o data warehouse já vinha com uma proposta de arquitetura diferente volta para extração de relatórios. Neste sentido, existem outros perfis de aplicação que demandam uma um atenção diferente para a questão da disponibilidade, velocidade de processamento e volume dados.
- Pensando em solucionar diversos problemas relacionados à escalabilidade, performance e disponibilidade, projetistas do NoSQL promoveram uma alternativa de alto armazenamento com velocidade e grande disponibilidade, procurando se livrar de certas regras e estruturas que norteiam o Modelo Relacional.

Um novo paradigma: BD NoSQL

- Teorema de CAP
- Um sistema distribuído pode entregar apenas duas das três características desejadas: consistência (*consistency*), disponibilidade (*availability*) e tolerância de partição (*partition tolerance*).



Um novo paradigma: BD NoSQL

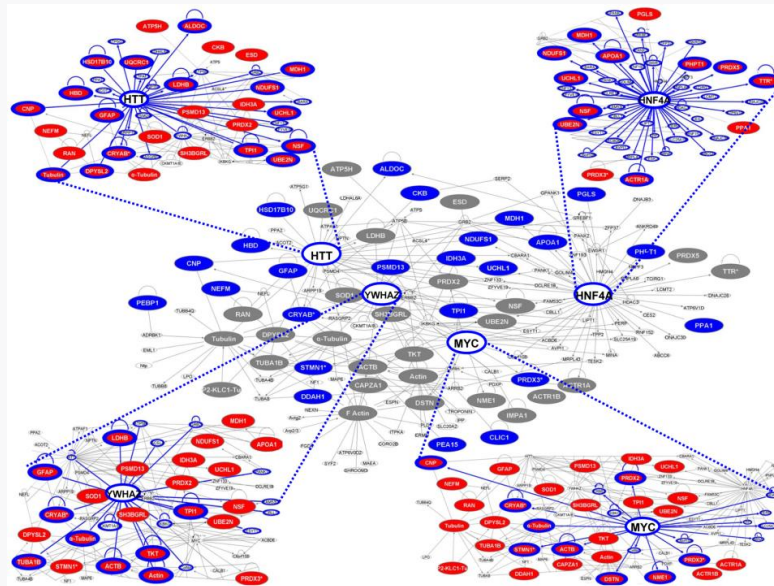
- Teorema de CAP
- Consistência (*consistency*) significa que todos os clientes veem os mesmos dados ao mesmo tempo, não importa em qual nó eles se conectem. Para que isso aconteça, sempre que os dados forem gravados em um nó, ele deve ser instantaneamente encaminhado ou replicado para todos os outros nós do sistema antes que a gravação seja considerada 'bem-sucedida'.
- Disponibilidade (*availability*) significa que qualquer cliente que fizer uma solicitação de dados obterá uma resposta.
- Uma partição (*partition tolerance*) é uma quebra de comunicação dentro de um sistema distribuído, uma conexão perdida ou qualquer outro problema temporariamente no *cluster*. Tolerância de partição significa que o cluster deve continuar a funcionar mesmo se ocorrer uma ou mais falhas de comunicação entre os nós do *cluster*.

Um novo paradigma: BD NoSQL

- Exemplos de SGBD que entregam CP, AP, CA.
- MongoDB é um exemplo de banco de dados CP, ou seja entrega consistência e tolerância a falhas. Ele funciona como um sistema de mestre único onde ele tem apenas um nó primário que recebe todas as operações de escrita. Adota a forma de armazenamento em documentos.
- Cassandra é um exemplo de banco de dados AP, não garante consistência. Permite armazenar os dados em uma rede distribuída. No entanto ao contrario do mongoDB ele tem uma arquitetura sem mestre. Adota a forma de armazenamento em colunas.
- Ambos trabalham com cluster.
- Bancos de dados Relacionais como Oracle, PostgreSQL e MySQL podem ser enquadrados na classificação CA, possuem alta preocupação com a consistência e disponibilidade.

Computação em Cluster

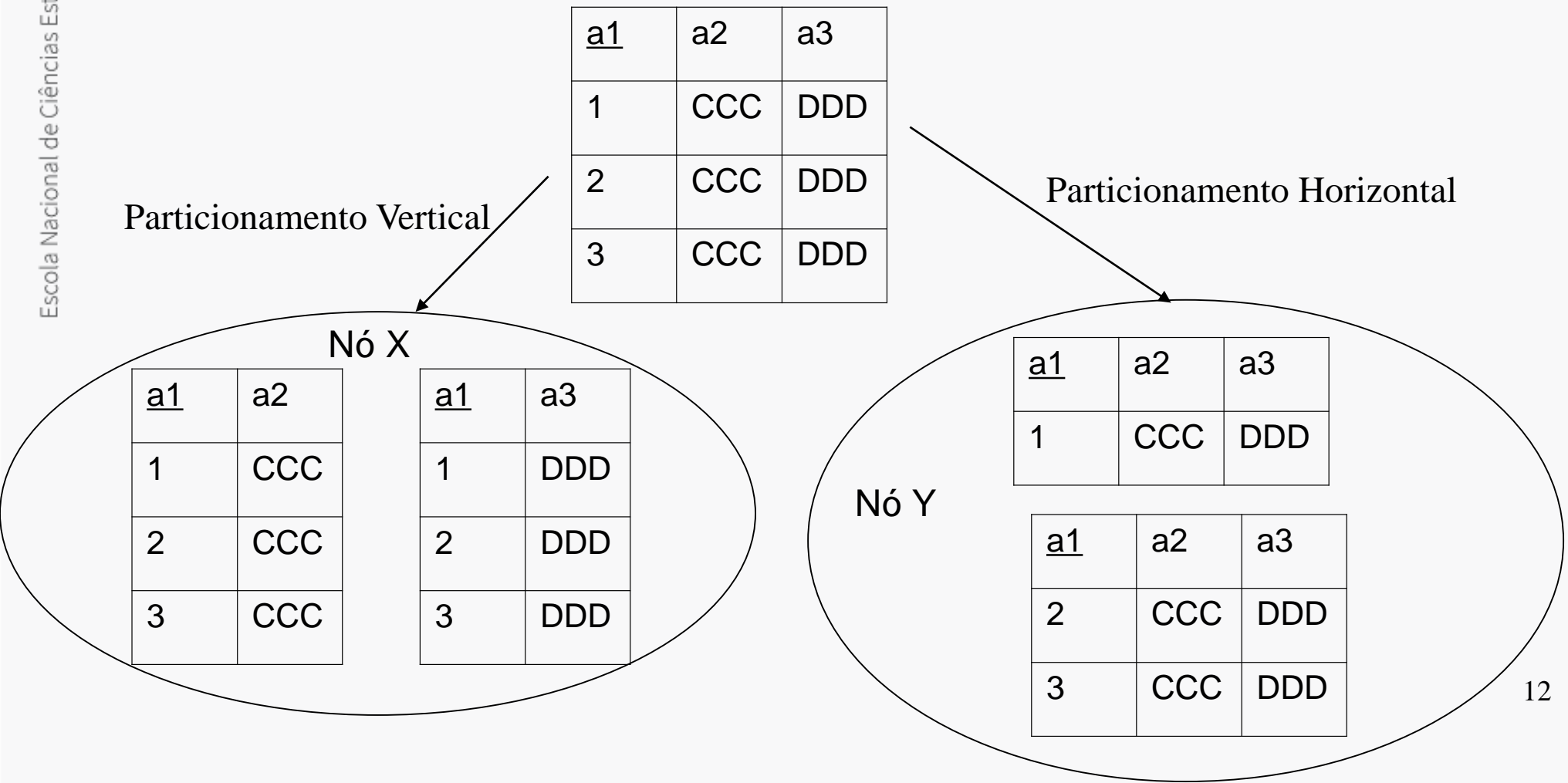
- Mas o que é Banco de Dados em Cluster?
 - Sistema de processamento distribuído, onde dois ou mais computadores (nós) trabalham em conjunto como se fossem um único recurso computacional.
 - Grosso modo o usuário usa o sistema como se fosse um único repositório. Há dois modelos de distribuição de dados:
 - Sharding
 - Replication



Computação em Cluster

- Arquitetura Sharding

- Diferentes partes dos dados são colocadas em diferentes nós. Isso favorece as aplicações em que usuários diferentes acessam partes diferentes do BD. Melhora o desempenho da leitura e da escrita.



Computação em Cluster

- **Sharding - Desvantagens:**
 - Exige maior controle para manter os diferentes nós com as informações fragmentadas.
 - Pode ocorrer que um nó recebe mais informações do que outro.
 - Problemas em um nó pode tornar as informações inconsistente até que seja resolvido.
- Caso de uso:
 - Algumas tecnologias de banco de dados especializadas – como o MySQL Cluster ou determinados produtos de banco de dados como o MongoDB Atlas – incluem o fragmento automático como um recurso (Fonte: <https://www.digitalocean.com/community/tutorials/understanding-database-sharding>).
 - Blockchain - é um repositório de registros, compartilhado e imutável (é imutável na comunicação de blocos de dados) e que facilita gravação de transações e rastreamento de ativos numa rede de negócios (Muito popular nas transações de criptomoedas).

Computação em Cluster

- *Replication*
 - Pega os mesmos dados e os copia em diversos nós:
 - (ii) peer-to-peer (um nó pode ser cliente e servidor)– Reduz o gargalo de escrita, mas a consistência torna-se complexa. Exemplo Apache Hadoop.
 - Porque replicar dados?
 - A maior razão é criar redundância extra. Se uma das máquinas que hospeda um banco de dados cair, a recuperação é exponencialmente mais rápida, pois uma das outras máquinas em que o banco de dados é replicado pode atender a demanda solicitada.
 - **Vantagens**
 - (i) master-slave – Bom para leitura, alta resiliência
 - Isso significa que uma manutenção no nó A não vai causar inatividade, pois a outra versão contida no nó B será acionada.
 - Favorece a disponibilidade para leitura.

Um novo paradigma: BD NoSQL

- Tipos de banco de dados NoSQL: **CHAVE-VALOR**
- Todos os registros fazem parte da mesma coleção de elementos, e a única coisa que todos eles têm em comum é uma chave única. Tanto as chaves quanto os valores podem ser qualquer coisa, desde objeto simples até um complexo.
- Permitem funcionamento em computação de *clusters*.
- Consultas mais complexas devem ser resolvidas do lado da programação através do desenvolvedores.
- Caso de uso:
 - i) Compartilhamento de dados de sessão de usuário entre vários servidores de aplicação, conhecido como session storage. Esta solução está disponível nos principais navegadores de internet atuais.
 - ii) Carrinho de compras de sites podem se beneficiar do seu poder de processamento para resolver diversos pedidos.

BD NoSQL – Chave-Valor Redis

- Exemplos de banco chave-valor: Redis, Memcached, DynamodDB estão entre os mais populares, além deste podemos citar o Riak. DynamoDB oferece o recurso de ACID comum nos BD relacionais.
- **Redis**
 - Redis persiste em memória e oferece uma estrutura simples e mais complexas de armazenamento como string, hash, listas, e set (é uma coleção tipo um vetor). É muito usado como cache. Entre as organizações que usam Redis estão Twitter, GitHub, Instagram, Pinterest e Snapchat.
 - Foi desenvolvido em C ANSI e recomenda-se que seja implantado no Linux, pois não há suporte oficial para a Windows.
 - É extremamente rápido.
 - É possível configurar para fazer copia de segurança em disco, assim caso a máquina seja reiniciada por algum problema os dados podem ser recuperados.
 - Pode rodar em clusters

BD NoSQL – Chave-Valor Redis

- Exemplo de uso do Redis com um cliente de conexão conhecido como CLI.

- SET ultimo_usuario_logado "Fabiola"
 - OK

Este comando SET informa a chave do ultimo usuário logado e diz para o CLI persistir com o valor "Fabiola". O CLI retorna OK para informar que a operação ocorreu com sucesso.

- GET ultimo_usuario_logado
 - "Fabiola"

Este comando GET informa que deseja recuperar o valor da chave ultimo_usuario_logado e o CLI retorna "Fabiola".

Digamos que o usuário saiu do sistema seja necessário excluí-lo. Para isso, usa-se o comando DEL chave. Veja que o CLI retornou a quantidade de chaves excluídas, 1.

- DEL ultimo_usuario_logado
 - 1

BD NoSQL – Chave-Valor Redis

Exemplo de uso Redis

- Imagine um site que mantém informações que são alteradas com um espaço de tempo maior, por exemplo, 2 semanas. Neste sentido, trabalhar com buscas para recuperar a informação no BD torna-se custosa e desnecessária. Exemplo resultados de loterias. Possuem muitos acessos com picos em períodos determinados de tempo. Assim, manter estes dados em cache parece ser uma boa medida para reduzir o custo de processamento.

BD NoSQL – Chave-Valor DynamoDB

Como funciona o DynamoDB?

- Garante ACID.
- Não pode ser baixado, está disponível como serviço da *Amazon Web Services* (AWS)
- Trabalha com o tipo chave-valor para montar cada documento. A estrutura de cada documento é semelhante a uma tabela com esquema flexível.

Aluno

pk_id	sk_id	Atributos			
15	Jonas	nome	idade	raça	nota
		Jonas	18	pardo	8,3
16	Kely	nome	idade	nota	
		Kely	20	9,5	

BD NoSQL – Chave-Valor DynamoDB

Como funciona o DynamoDB?

- Conforme a tabela abaixo, a representação dos dados é feita na horizontal, chamada de partição. Cada partição deve ter uma *partition key* única para cada partição(pk_id). A inclusão de chave de ordenação (*sorted key*) na estrutura serve para ajudar nas buscas (sk_id).
- Seu funcionamento adota replicação dos dados em três partes iguais entre os nós para garantir disponibilidade.

Aluno

pk_id	sk_id	Atributos			
15	Jonas	nome	idade	raça	nota
		Jonas	18	pardo	8,3
16	Kely	nome	idade	nota	
		Kely	20	9,5	

BD NoSQL – Chave-Valor

Como funciona o DynamoDB?

- Exemplo da tabela Aluno:
- A lógica de busca faz uso da pk_id e sk_id.
 - Exemplo: pk_id=15 e sk_id= "Jonas"

Aluno

pk_id	sk_id	Atributos			
15	Jonas	nome	idade	raça	nota
		Jonas	18	pardo	8,3
16	Kely	nome	idade	nota	
		Kely	20	9,5	

BD NoSQL – Chave-Valor

Como funciona o DynamoDB?

- Exemplo da tabela Pedido com itens comprados pelo cliente.
- A lógica de busca faz uso da pk_cliente e sk_item.
 - Exemplo: pk_nota=15 e sk_item= "001".

Pedido

id_pedido	item	Atributos				
15	item001	nome_cliente	produto	preço	quantidade	peso
		Carla	caneta	30	100	10g
	item002	nome_cliente	produto	preço	quantidade	
		Mateus	Kely	20	9,5	

- Uma abordagem possível para este tipo de modelo é a agregação de relações, por exemplo, as relações *Cliente* e *Pedido* podem fazer parte da mesma tabela, com isso elimina a necessidade do JOIN muito comum nos modelos relacionais tradicionais.

BD NoSQL - Grafo

- **O que é um BD de Grafo?** É um tipo de BD que os registros são nós num grafo interligados por relacionamentos.
- **O que é um grafo ?** Um grafo é composto de dois elementos: um nó e uma aresta, que vamos chamar de relacionamento.
- Cada nó representa uma entidade (uma pessoa, um lugar, uma coisa, uma categoria ou outra peça de dados) e cada associação representa como dois nós estão associados. Por exemplo, dois nós, como: “bolo” e “sobremesa” possuem um relacionamento “é um tipo de” apontando de “bolo” para “sobremesa”. Isso define que bolo é um tipo de sobremesa.
- Um banco de dados orientado à grafo é um SGBD que oferece operações para criar, ler, atualizar e excluir (CRUD). O foco deste modelo são os relacionamentos entre os dados. Assim, quando se tem grande volume de dados interconectados esta solução ajuda a melhorar o desempenho e capacidade de resposta, pois seus relacionamentos são físicos, diferindo do modelo relacional tradicional que usa JOIN.
- Funciona numa arquitetura horizontal.

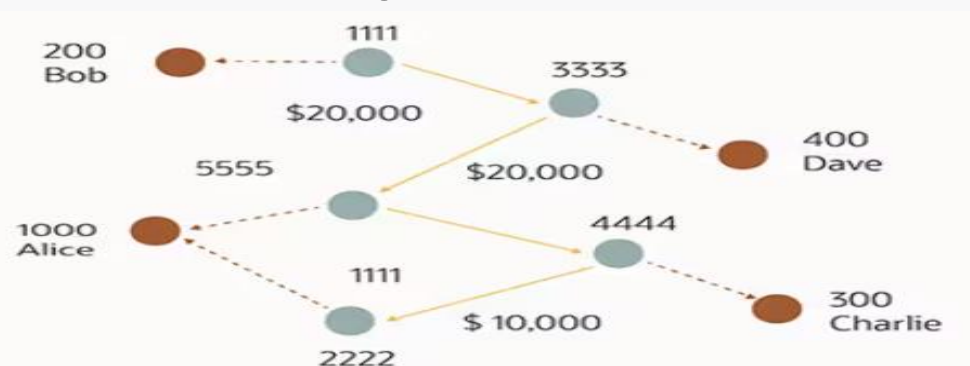
BD NoSQL - Grafo

- Casos de uso:
 - Redes sociais como Facebook, linkedIn, Twitter
 - Detecção de fraude
 - Mecanismo de recomendação em tempo real
 - Operações de rede em TI
 - Pesquisa baseada em grafo

BD NoSQL - Grafo

- Casos de uso:
 - Exemplo: Especificamente, uma transferência de dinheiro circular envolve um criminoso que envia grandes quantias de dinheiro obtidas de forma fraudulenta para si mesmo, mas oculta isso por meio de uma longa e complexa série de transferências válidas entre contas "normais". Essas contas "normais" são, na verdade, contas criadas com identidades roubadas (endereço de e-mail, endereço, documento).
 - Com o uso do BD de grafo é possível modelar associações entre entidades para detectar informações semelhantes, assim é possível retornar todos os clientes com informações conectadas.

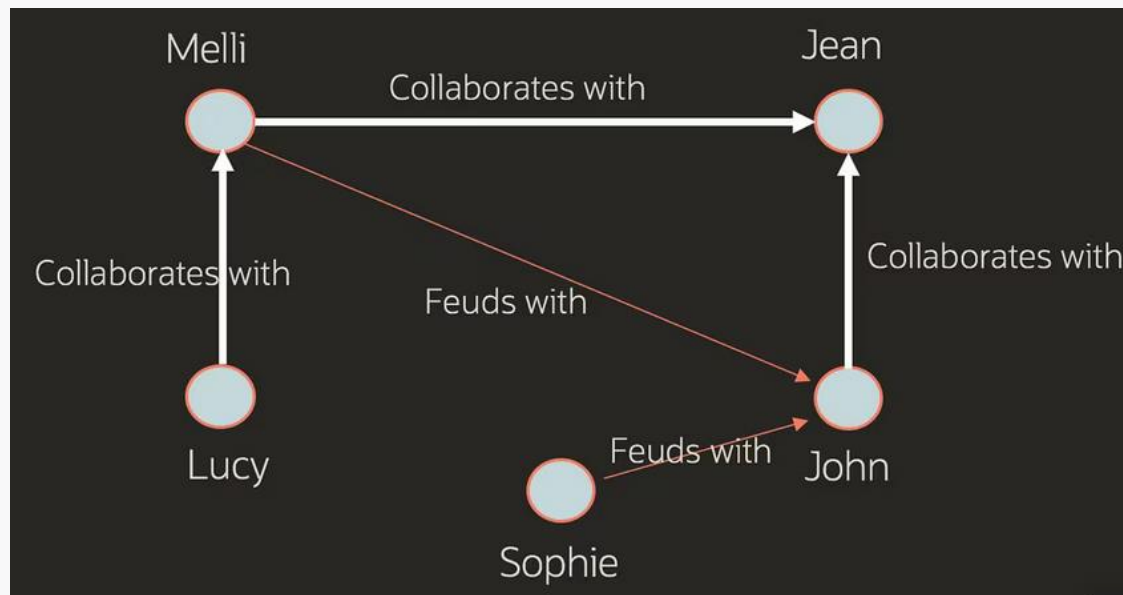
Fluxo de lavagem de dinheiro



BD NoSQL - Grafo

- Por exemplo, algoritmos de grafos podem identificar qual indivíduo ou item está mais conectado a outros nas redes sociais ou processos comerciais. Para retornar a rede de amigos de John até a terceira ordem (amigo do amigo do amigo) um BD relacional teria que executar vários JOIN, que é uma operação custosa.

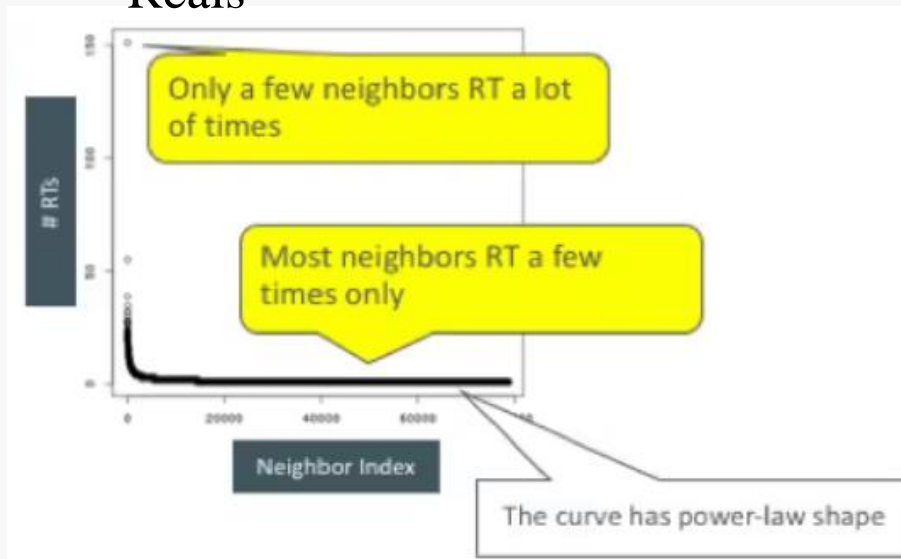
Exemplo de rede de amigos



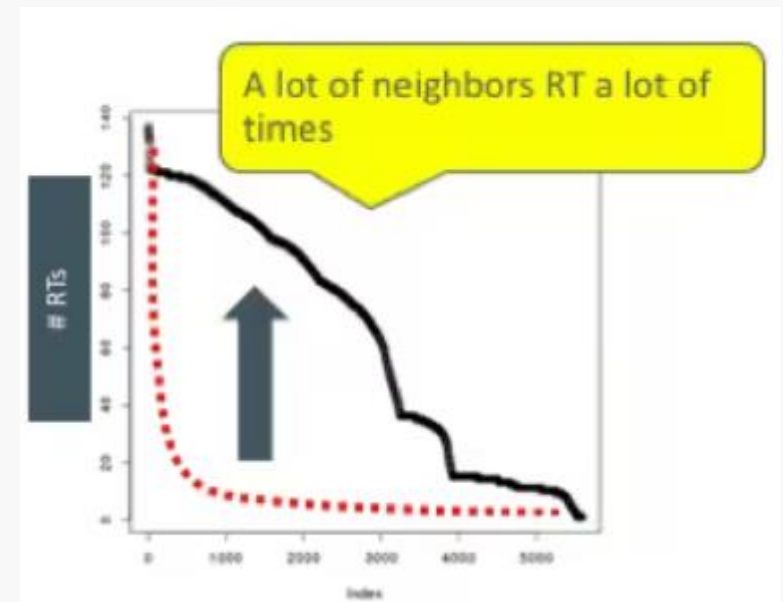
BD NoSQL - Grafo

- Em um caso de uso do mundo real, foi usado o Oracle Marketing Cloud para avaliar a publicidade e a atração nas mídias sociais, especificamente, para identificar contas de *bot* falsas (robôs que enviam mensagens) que distorcem os dados. Esses *bots* reenvia mensagens para as contas-alvo, aumentando artificialmente sua popularidade. Uma análise de padrão permitiu a contagem de reenvio de mensagens e densidade de conexões com os vizinhos. Uma avaliação comparativa nas redes sociais mostrou um padrão diferente entre contas reais e contas controladas por bot.

Reais

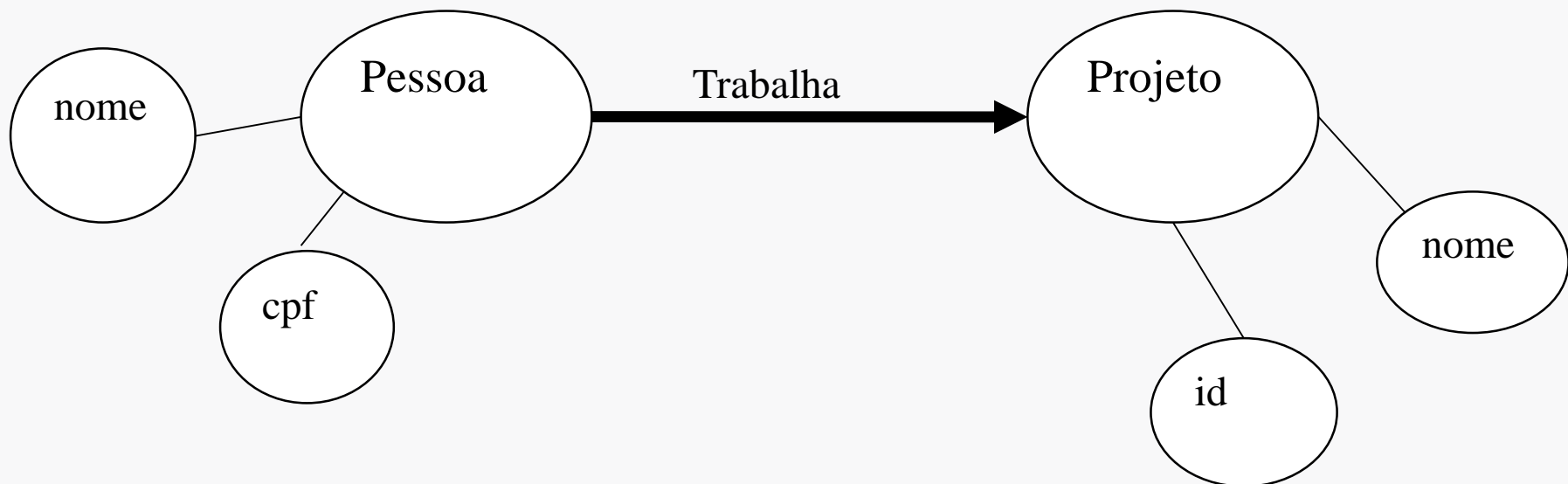
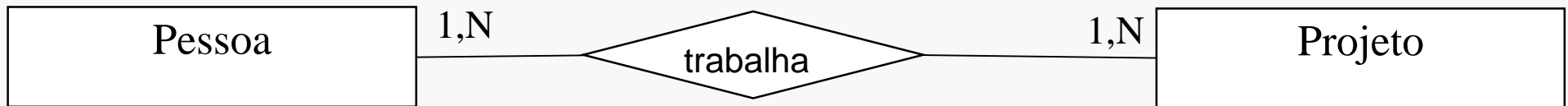


Bot



BD NoSQL – BD de Grafo com Neo4j

- Exemplo de modelagem conceitual e o correspondente modelo no **Neo4j**, um BD orientado à grafos.



BD NoSQL – BD de Grafo com Neo4j

- A linguagem de consulta usada pelo Neo4j é o Cypher.
- Para criar a estrutura de grafo referente ao modelo anterior usa-se a seguinte sintaxe:

```
CREATE (a:Pessoa {nome:"Andre"})
```

```
CREATE (p:projeto {sigla:"PNSB"})
```

```
CREATE (a)-[:trabalha {sigla:"PNSB"}]->(p)
```

- O Conjunto de comando devem ser executados numa única transação para que as variáveis "a" e "b" respectivamente dos nós de funcionário e projeto sejam usadas para criar o relacionamento trabalho de "Andre" para "PNSB".



BD NoSQL – BD de Grafo com Neo4j

- A linguagem de consulta usada pelo Neo4j é o Cypher.

```
CREATE (a:funcionario {nome:"Andre"})
```

O Neo4j usa parênteses, () ,para definir um nó e chaves, { }para definir uma coleção de variáveis separadas por vírgula, o símbolo de dois pontos, : , é usado para atribuição de valor. A sintaxe do comando *create* segue a seguinte definição para criação de um nó:

```
CREATE (variável de referencia: tipo de propriedade {parâmetro1: valor1,  
parâmetro2: valor2, ..., parâmetron: valorn})
```

```
CREATE (a)-[:trabalha {sigla:"PNSB"}]->(p)
```

O Neo4j usa colchetes, [], para criar um relacionamento. Veja que este relacionamento possui a propriedade sigla que recebe o valor " PNSB“. Os dois tracinhos antes e depois dos colchetes fazem a ligação entre dois nós. A seta de direcionamento, definida por tracinho e sinal de maior, indica a direção do grafo.

BD NoSQL – BD de Grafo com Neo4j

- Para realizar uma consulta nos dados, usa-se a seguinte sintaxe.
-- O comando *match* com *return* é o equivalente ao **SELECT** do SQL:
SELECT f.nome FROM Funcionario f.

```
match (f:funcionario)
```

```
return f.nome
```

<div>Table RAW</div>	
p.nome	
"Andre"	

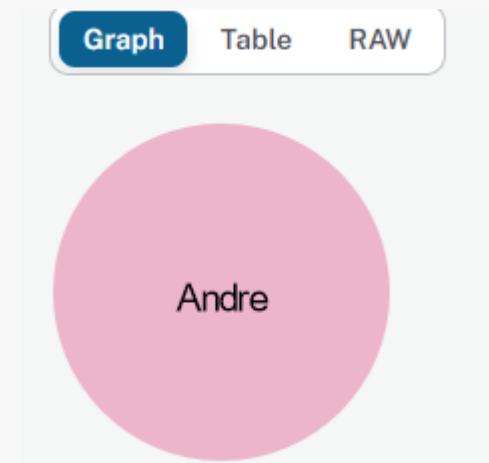
BD NoSQL – BD de Grafo com Neo4j

- Para realizar uma consulta nos dados, usa-se a seguinte sintaxe.
- O comando *match* com *return* e *where* é o equivalente ao usado no SELECT do SQL: SELECT f.nome FROM Funcionario f where f.nome='Andre'.

```
match (f:funcionario)
```

```
where f.nome='Andre'
```

```
return f
```



BD NoSQL – BD de Grafo com Neo4j

- Para realizar uma consulta neste BD de grafo, usa-se a seguinte sintaxe.

```
MATCH resultado=()-[:trabalha]->()
```

```
RETURN resultado LIMIT 2;
```

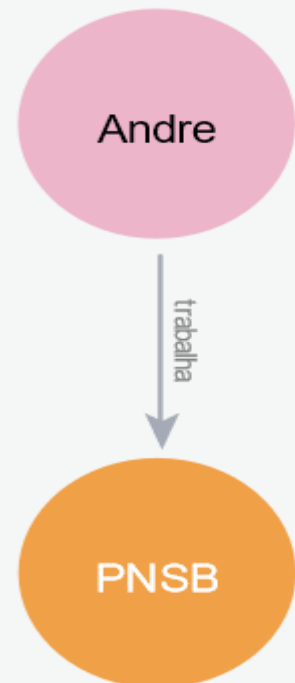
A variável resultado recebe o conteúdo retornado pela consulta. A sequência de caracteres define uma consulta que retorne um par de nós, onde há um relacionamento do tipo trabalho entre eles. O comando RETURN retorna o resultado para tela. A cláusula limite define que seja exibido no máximo 2 resultados.

Graph **Table** RAW

resultado

```
1 (:funcionario {nome: "Andre"})-[:trabalha {sigla: "PNSB"}]->(:projeto {sigla: "PNSB"})
```

Graph Table RAW



BD NoSQL – BD de Grafo com Neo4j

- Casos de uso:
 - Challenger Bank Current dos EUA usa tecnologia de gráfico Neo4j para construir serviços centrados em torno do relacionamento com clientes. Perspectiva de crescimento de US\$ 91.48 bilhões em 2023 para US\$ 1.232,83 bilhões até 2030. (<https://exactitudeconsultancy.com/pt/reports/37405/neo-and-challenger-bank-market/>)
 - Zurich Seguros, Suíça é uma seguradora que conectou os dados de todas as suas empresas em mais de 210 países para detecção de fraudes. Ela mantém um BD com 20 milhões de nós e 35 milhões de relacionamentos. Resultados recentes apontam elevação do lucro para US \$ 7,4 bilhões.
 - O eBay escolheu o Neo4j como o banco de dados de grafos nativos que contém os modelos probabilísticos que ajudam a entender o cenário de compras de conversação. Quando um comprador procura por “sacos castanhos”, por exemplo, o eBay App sabe quais detalhes perguntar em seguida, como tipo, estilo, marca, orçamento ou tamanho. No ramo de comércio eletrônico a eBay possui um capital de US \$ 24,10 bilhões. (<https://skilling.com/row/pt/markets/shares/ebay/>)

BD NoSQL - Documento

- Tipo de banco de dados NoSQL: **DOCUMENTO**
 - Cada registro fica armazenado em uma coleção específica, mas mesmo dentro de uma coleção, não existe um esquema fixo para os registros.
 - Sua estrutura costuma usar XML, JSON, BSON e outras formas.
 - Permitem tipos de documentos diferentes em um único armazenamento, permitem que os campos dentro deles sejam opcionais e geralmente permitem que eles sejam codificados usando sistemas diferentes de codificação, por exemplo: um documento pode estar no formato XML e outro no formato JSON.

BD NoSQL – MongoDB

- **MongoDB é um BD documento.**
- Os dados ficam armazenados em um formato muito semelhante ao JSON, chamado de BSON. Cada documento possui campos e não há um esquema predefinido de que todos documentos tenham os mesmos campos.
 - Não é projetado para uso de JOIN, usa documentos aninhados
 - Faz uso de PK.
 - Seus dados são agregados em estágios (pipeline)
 - É open-source não relacional
 - Funciona no Linux, MACOS e Windows
 - Possui interface de cliente para execução de comandos em linha.
 - Usa a linguagem JavaScript para executar *queries*.

BD NoSQL - MongoDB

- Cada documento individual é armazenado em uma coleção
- A operação `createCollection()` pode ser utilizada para criar uma coleção.
- Por exemplo, para criar a coleção “filmes”, equivalente a uma tabela:
`db.createCollection("Aluno")`
- Por exemplo, inserir um documento na coleção "Aluno", equivalente ao `INSERT INTO Aluno (matricula, nome, sexo, idade) VALUES (1234, "Rosane", "M",18)`:

```
db.Aluno.insert( {  
  "matricula" : 12345,  
  "nome" : "Rosane",  
  "sexo" : "M ",  
  "idade" : 18  })
```

BD NoSQL - MongoDB

- Exemplo de INSERÇÃO no MongoDB

```
db.Aluno.insert( {  
  "matricula" : 234567,  
  "nome" : "Aderbal",  
  "sexo" : "M",  
  "idade" : 23  
  "cadeira" : ["Estatística", "Português", "Algoritmos" ]  
})
```

BD NoSQL- Colunar

- Um banco de dados colunar persistem os dados por colunas criando um relacionamento através de um id.

Banco de dados Relacional

sku_produto	nome_produto	preco_produto
1	Camisa	100
2	Bola	50
3	Cafeteira	90
4	Colcha	200

Banco de dados Colunar

Sku_produto

id	valor
0	1
1	2
2	3
3	4

Nome_produto

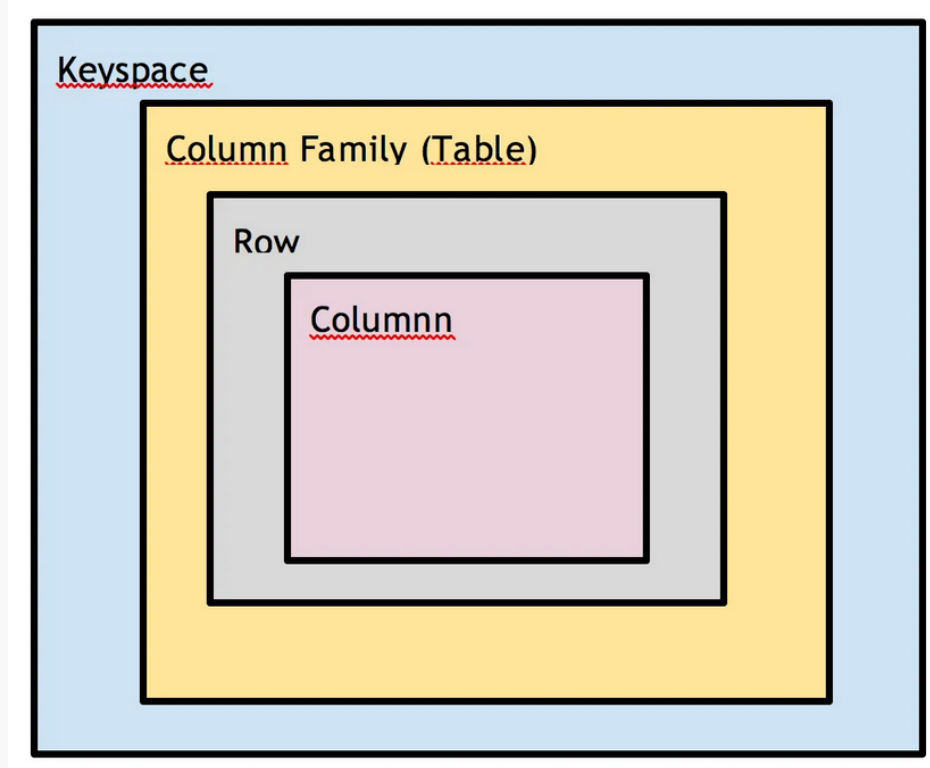
id	valor
0	Camisa
1	Bola
2	Cafeteira
3	Colcha

Preco_produto

id	valor
0	100
1	50
2	90
3	200

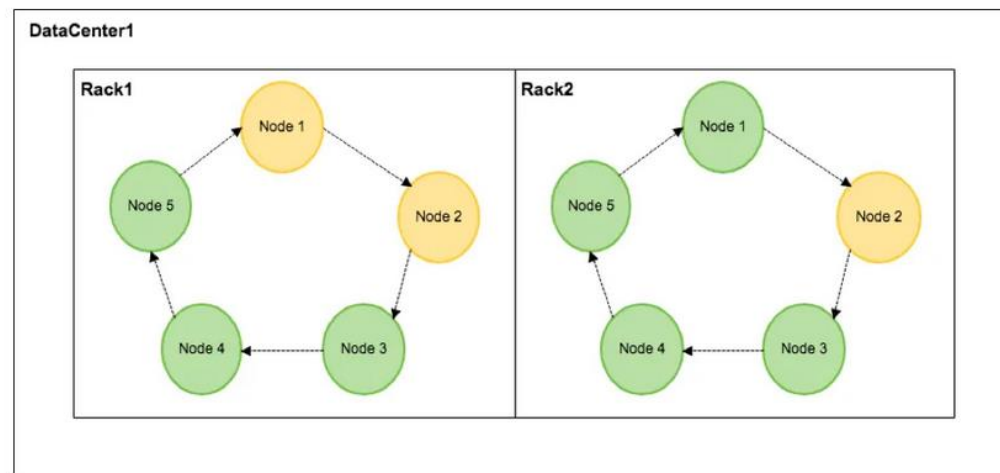
BD NoSQL - Cassandra

- É um BD colunar
- O BD relacional possui a hierarquia banco, tabela e coluna. O Cassandra usa uma solução semelhante estruturado em keyspaces, tables (column families), rows (linhas) e columns (colunas).



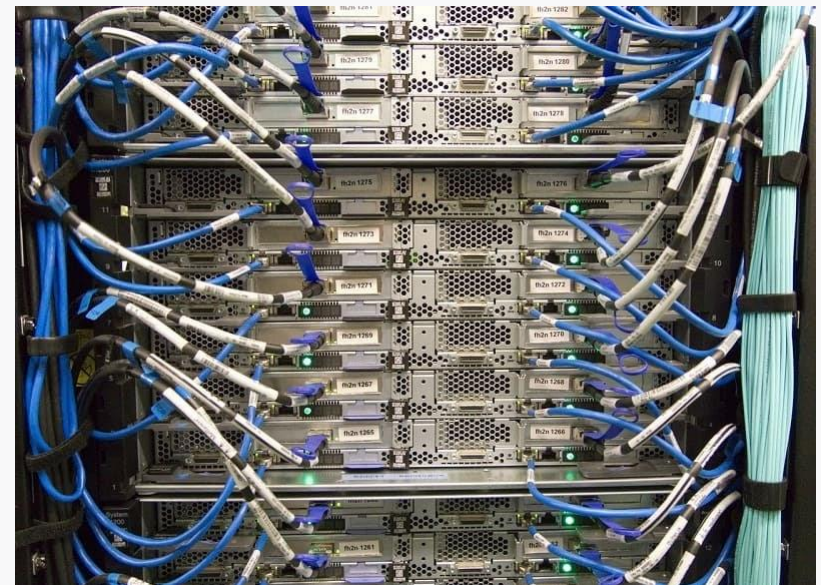
BD NoSQL - Cassandra

- Keyspace – agrega as tabelas do sistema. Trabalha com um fator de replicação (FP): Por exemplo, 5 máquinas e FP igual 2, A replicação será feita em pelo menos 2 máquinas.
- Permite dois tipos de estratégia de replicação:
 - i) simples (SimpleStrategy), replica os dados para o próximo nó do cluster.
 - ii) Topologia de rede (NetworkTopologyStrategy), a persistência dos dados será feita em mais de um datacenter. Um data center pode conter vários clusters organizados e racks diferentes.



Exemplo de Datacenter

Um datacenter corresponde a um local físico que armazena máquinas de computação e seus equipamentos de hardware relacionados. São locais seguros e adequados para a infraestrutura de TI (máquinas, clusters, dados, etc).



BD NoSQL - Cassandra

- **Tables** (column families) – é equivalente a uma tabela do mundo real com C colunas e L linhas.
 - As famílias de colunas possui os seguintes atributos:
 - Primary key: É composta de 2 partes:
 - (i) Partition key define qual partição será armazenado o dado.
 - (ii) Clustering key: É o restante da chave.
 - Columns: Representam as demais colunas da tabela.
 - Row: É composta pela primary key e um conjunto de colunas.
 - A coluna é composta pelos seguintes atributos:

Column Key: nome da coluna

Column value : É o valor que está sendo persistido.

Timestamp: O Cassandra utiliza esse campo para resolver conflitos e determinar qual é o valor mais atual.

BD NoSQL - Cassandra

- Exemplos de Rows (linhas)

ROW 1	<table><tr><td></td><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td><td></td></tr></table>																		
ROW 2	<table><tr><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr></table>																		
ROW 3	<table><tr><td></td><td></td></tr><tr><td></td><td></td></tr></table>																		

BD NoSQL - Cassandra

- **Transação:** O Cassandra possui um recurso (*feature*) de persistência atômica que permite executar um bloco de comandos de uma só vez. Assim, não se trata de uma transação, mas os dados somente serão dados como persistidos com sucesso se todos os comandos forem executados com sucesso.
- **Consistência:** Como o Cassandra leva alguns milissegundos para replicar os dados no cluster, suas queries podem não retornar a última versão dos registros, gerando um problema de consistência.
- Vantagens
 - AP (*availability* – disponibilidade; *partition tolerance* – Tolerante a falhas);
 - Escalabilidade vertical e linear: se precisa dobrar o poder de resposta basta dobrar a infraestrutura;
 - Suporta N datacenters
- Não é recomendado para aplicações que precisam de muita consistência.

BD NoSQL - Cassandra

- Permite instalação no Windows
- Necessita que o java esteja instalado na máquina
- Possui um interface para executar comandos em linha: cqlsh.

Como criar um banco de dados ?

```
> CREATE KEYSPACE blog
```

```
WITH
```

```
REPLICATION = {
```

```
  'class': 'SimpleStrategy',
```

```
  'replication_factor' : 1
```

```
};
```

- Como está configurado apenas 1 nó no cluster, o fator de replicação será 1.

BD NoSQL - Cassandra

- Como criar tabela?

Selecione o keyspace criado.

> Use blog

> CREATE TABLE posts (

tag varchar,

name varchar,

author varchar,

description text,

likes int,

PRIMARY KEY (tag, name)

);

- O campo tag é a partition key e a row key é composta por tag + name.

BD NoSQL - Cassandra

- **Como Inserir dados?**

> insert into blog.posts (tag, name, author, description, likes)

Values

('apache-cassandra','Cassandra post','Jose','post do cassandra',0);

Caso o INSERT de mesma chave seja executado duas vezes, o registro será alterado.

Exemplo:

insert into blog.posts (tag, name, author, description, likes)

values

('apache-cassandra','Cassandra post','Jose','post do cassandra',1);

O Cassandra vai identificar que se trata de uma mesma partition key e mesma row key, assim será feita uma alteração na row (linha), diferentemente do que ocorre no BD relacional.

BD NoSQL - Cassandra

- **Como atualizar dados?**

```
> update posts set likes = 2
```

```
where tag = 'apache-cassandra' and name = 'Cassandra post';
```

Atenção:

Quando se tenta realizar uma atualização do registro cujo a chave ainda não exista, o Cassandra vai incluir um novo registro.

```
update posts set likes = 2
```

```
where tag = 'apache-kafka' and name = 'Kafka post';
```

BD NoSQL - Cassandra

- Como realizar consulta básica?

> Select * from posts;

tag	name	author	description	likes
apache-cassandra	Cassandra post	Jose	post do cassandra	2
apache-kafka	Kafka post	null	null	2

(2 rows)

Consulta com condição ?

É fundamental informar a partition key na condição.

select * from posts where **tag** = 'apache-Cassandra';

tag	name	author	description	likes
apache-cassandra	Cassandra post	Jose	post do cassandra	2

(1 rows)

BD NoSQL - Cassandra

- Consulta sem partition key?

```
> select * from posts where likes = 1;
```

```
InvalidRequest: Error from server: code=2200 [Invalid query]
message="Cannot execute this query as it might involve data filtering
and thus may have unpredictable performance. If you want to execute
this query despite the performance unpredictability, use ALLOW
FILTERING"
```

É possível fazer essa query utilizando a cláusula “ALLOW FILTERING”. Contudo perde performance e perde a capacidade de tolerância a falhas.

```
> select * from posts where likes = 2 allow filtering;
```

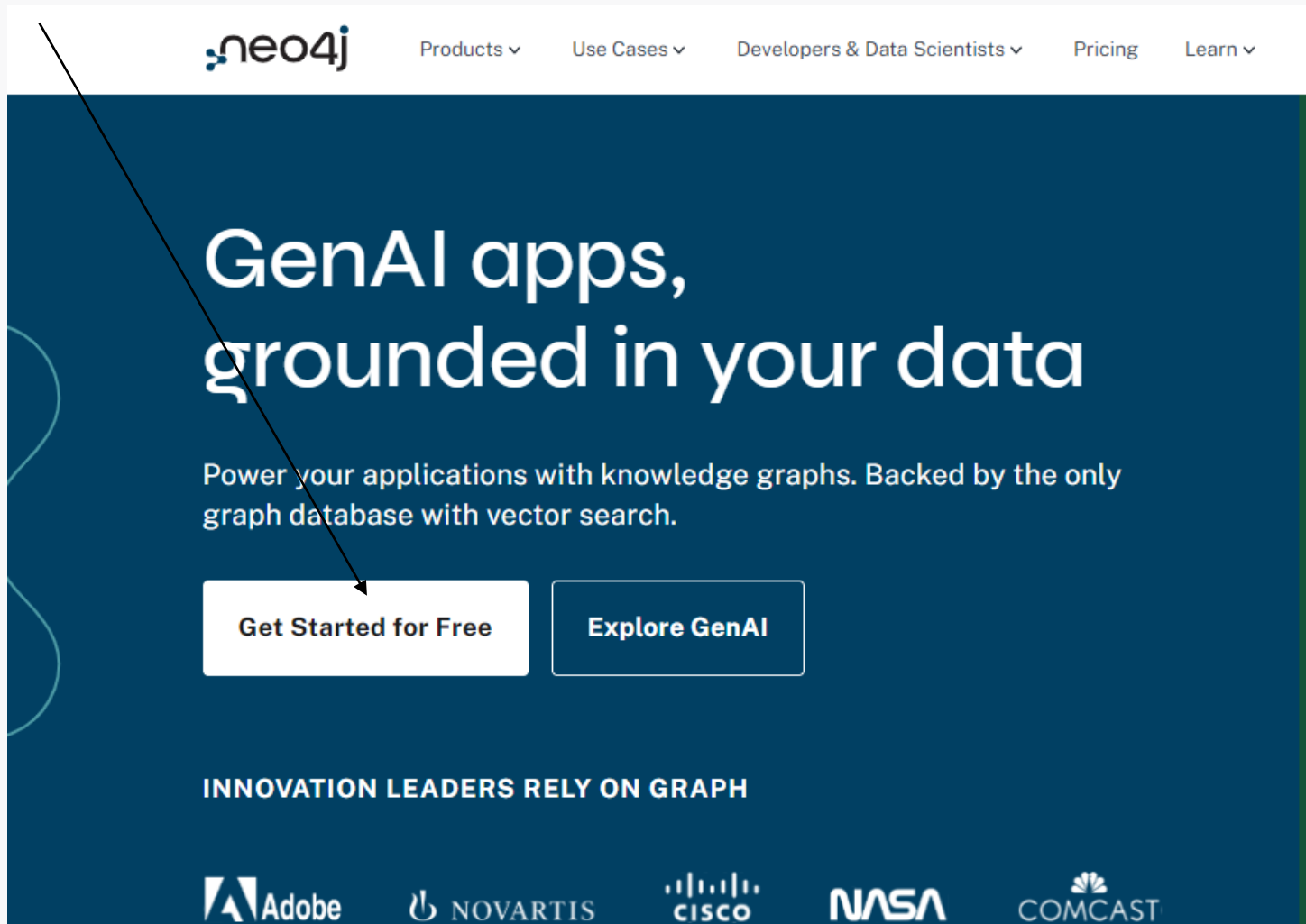
tag	name	author	description	likes
apache-cassandra	Cassandra post	Jose	post do cassandra	2
apache-kafka	Kafka post	null	null	2

(2 rows)

Exercícios (Para casa)

- Para executar consultas no Neo4j (<https://neo4j.com/>) use este endereço de internet para acessar a página para criar uma conta gratuita.

Passo 1: Clique aqui



Exercícios (Para casa)

- Para executar consultas no Neo4j (<https://neo4j.com/>) use este endereço de internet para acessar a página para criar uma conta gratuita.

Passo 2: Clique aqui

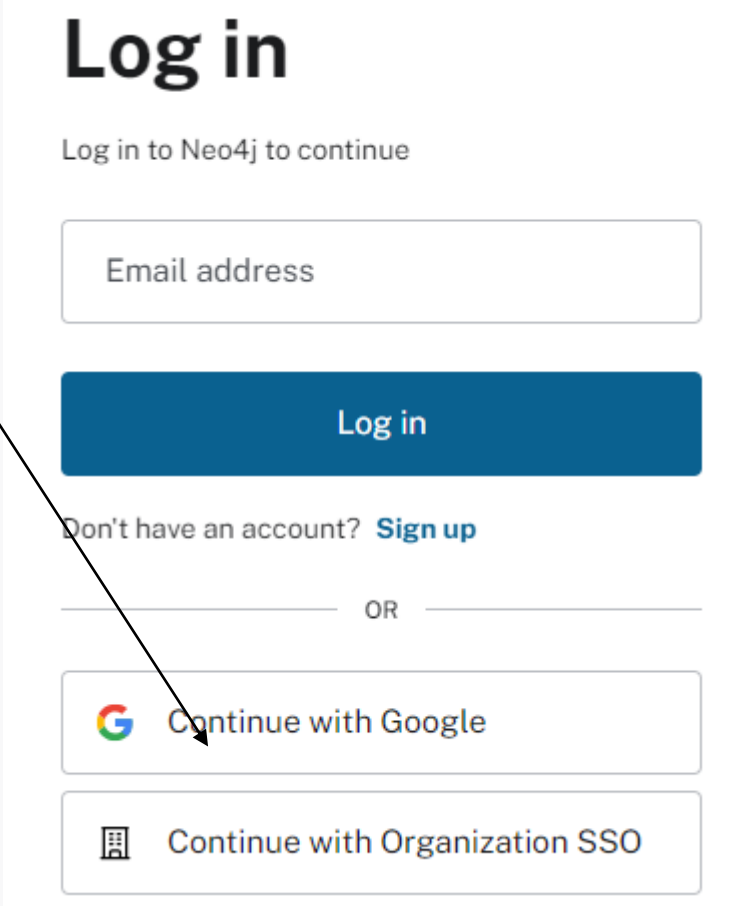


Exercícios (Para casa)

- Para executar consultas no Neo4j (<https://neo4j.com/>) use este endereço de internet para acessar a página para criar uma conta gratuita.

Passo 3: clique em Continue with Google

Ou informe seu e-mail e clique em login



The image shows a screenshot of the Neo4j login page. At the top, it says 'Log in' in a large, bold font. Below that, it says 'Log in to Neo4j to continue'. There is a text input field labeled 'Email address'. Below the input field is a blue button labeled 'Log in'. Underneath the button, it says 'Don't have an account? Sign up' with 'Sign up' in blue. Below this is a horizontal line with 'OR' in the center. Under the line are two buttons: 'Continue with Google' (with the Google logo) and 'Continue with Organization SSO' (with a building icon). A black arrow points from the text 'Continue with Google' in the instructions to the 'Continue with Google' button on the page.


Exercícios (Para casa)

- Para executar consultas no Neo4j (<https://neo4j.com/>) use este endereço de internet para acessar a página para criar uma conta gratuita.

Passo 5: informe seu e-mail do gmail

Clique em avançar

E informe sua senha

 Fazer login com o Google

Fazer login

Prosseguir para neo4j.com

[Esqueceu seu e-mail?](#)

Para continuar, o Google compartilhará com o app neo4j.com seu nome, endereço de e-mail, idioma preferido e sua foto do perfil. Consulte a [Política de Privacidade](#) e os [Termos de Serviço](#) do app neo4j.com antes de usá-lo.

[Criar conta](#)

[Avançar](#)

Exercícios (Para casa)

- Para executar consultas no Neo4j (<https://neo4j.com/>) use este endereço de internet para acessar a página para criar uma conta gratuita.

Passo 6: Clique em New Instance

The screenshot shows the Neo4j Aura web interface. On the left is a sidebar with navigation links: 'AuraDB', 'Instances' (highlighted in light blue), 'Connect', 'AuraDS', and 'Getting Started'. The main content area is titled 'Instances' and features a prominent blue 'New Instance' button. An arrow points from the 'Passo 6: Clique em New Instance' text to this button. Below the button, a card displays details for 'Instance01', which is a 'Free' tier instance with ID 'f1fc4f64' and a 'Running' status. It lists specifications: 'Neo4j version 5', 'Nodes 6 / 200000 (0%)', 'Relationships 3 / 400000 (0%)', and 'Region Iowa, USA (us-central1)'. The 'Connection URI' is 'neo4j+s://f1fc4f64.databases.neo4j.io'. Action buttons for 'Open', delete, and more options are visible at the bottom right of the instance card.

Exercícios (Para casa)

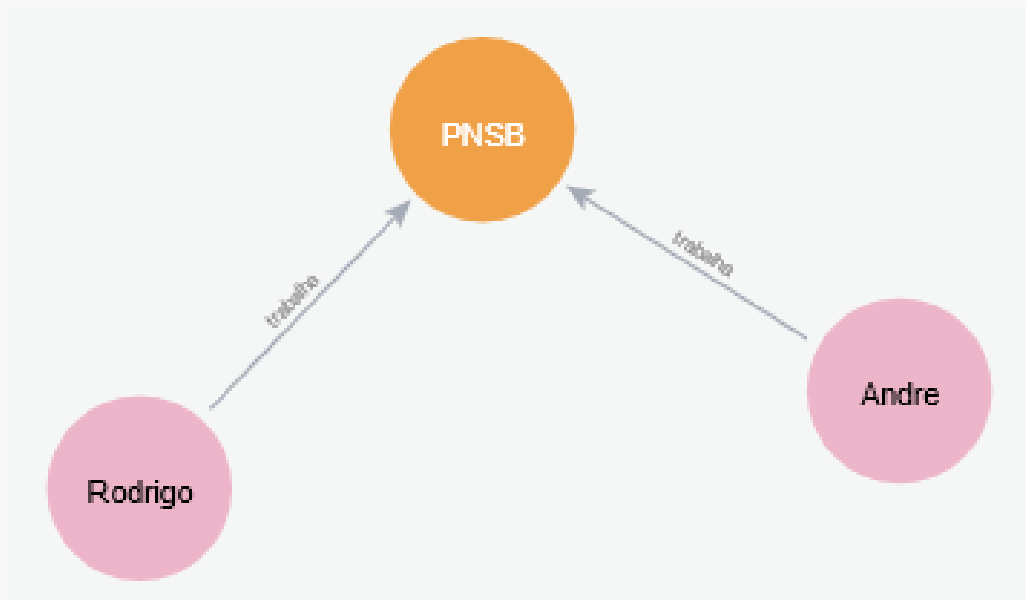
1) Criar uma conta no Neo4j (<https://neo4j.com/>) execute os seguintes conjunto de comando abaixo para criar as duas associações.

```
CREATE (a:funcionario {nome:"Andre"})
```

```
CREATE (r:funcionario {nome:"Rodrigo"})
```

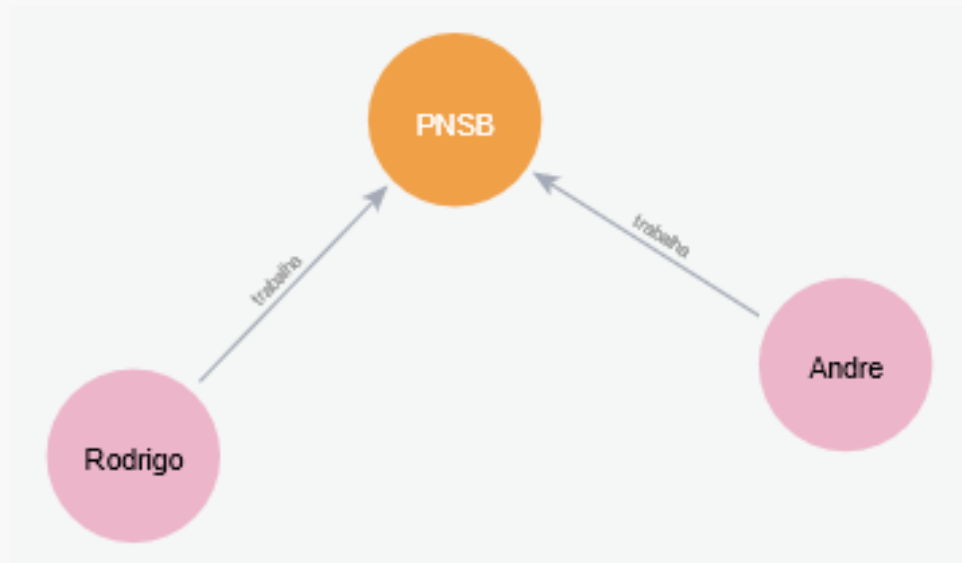
```
CREATE (p:projeto {sigla:"PNSB"})
```

```
CREATE (a)-[:trabalha {sigla:"PNSB"}]->(p)CREATE (r)-[:trabalha {sigla:"PNSB"}]->(p)
```



Exercícios

- 2) Construa uma consulta para retornar o funcionário Rodrigo.
- 3) Construa uma consulta para retorna o funcionário André.
- 4) Construa um consulta para retornar o grafo abaixo.



Fonte de consultas:

- Fontes:
 - <https://redis.io/docs/about/>,
 - <https://docs.riak.com/riak/ts/latest/>
 - <https://aws.amazon.com/pt/nosql/key-value/>
 - <https://www.oracle.com/br/autonomous-database/what-is-graph-database/#money-laundering>
 - <https://medium.com/codigorefinado/amazon-dynamodb-6eb351bb3107>
 - <https://oieduardorabelo.medium.com/amazon-dynamodb-o-qu%C3%AA-por-que-e-quando-usar-o-design-de-tabela-%C3%BA-nica-com-dynamodb-556f5d8c474d>
 - <https://medium.com/nstech/apache-cassandra-8250e9f30942>

Bibliografias

- Armazenando dados com Redis. Casa do código. Rodrigo Lazoti. Data publicação: 05/2014. Atualizado em 07/2020
- EBOOK. Guia Definitivo de Bancos de Dados Grafos: Para Desenvolvedores RDBMS. Por Michael Hunger, Ryan Boyd e William Lyon. 2021, Neo4j.
- Armazenando dados com Redis, casa do código. Rodrigo Lazoti.
- Guia Definitivo de Bancos de Dados Grafos: Para Desenvolvedores RDBMS
- Por Michael Hunger, Ryan Boyd e William Lyon

Obrigado