



***cassandra***

# Banco de dados NoSQL - Cassandra (Introdução)

Prof. Gustavo Leitão

# AGENDA

- Introdução ao Cassandra
- Instalação com Docker
- Comandos básicos
- Operações com CQL



- Cassandra é um banco de dados open-source distribuído e descentralizado altamente escalável projetado para lidar com volume extremamente grande de dados
- Cassandra possui alta disponibilidade e não possui um ponto único de falha

# Características

- Horizontalmente escalável
- Orientado a coluna
- Arquitetura baseada no Amazon Dynamo e estrutura de dados baseada no Google Big Table
- Escrito em Java







# Quem usa?

Apple atualmente possui  
**75.000** nós Cassandra  
armazenando **10 petabytes**



# Quem usa?

Netflix utiliza Apache Cassandra para o stream de videos.  
Atualmente são **2.500** nós armazenando **420 terabytes** de dados e respondendo **1 trilhão** de requisições por dia



# Quem usa?

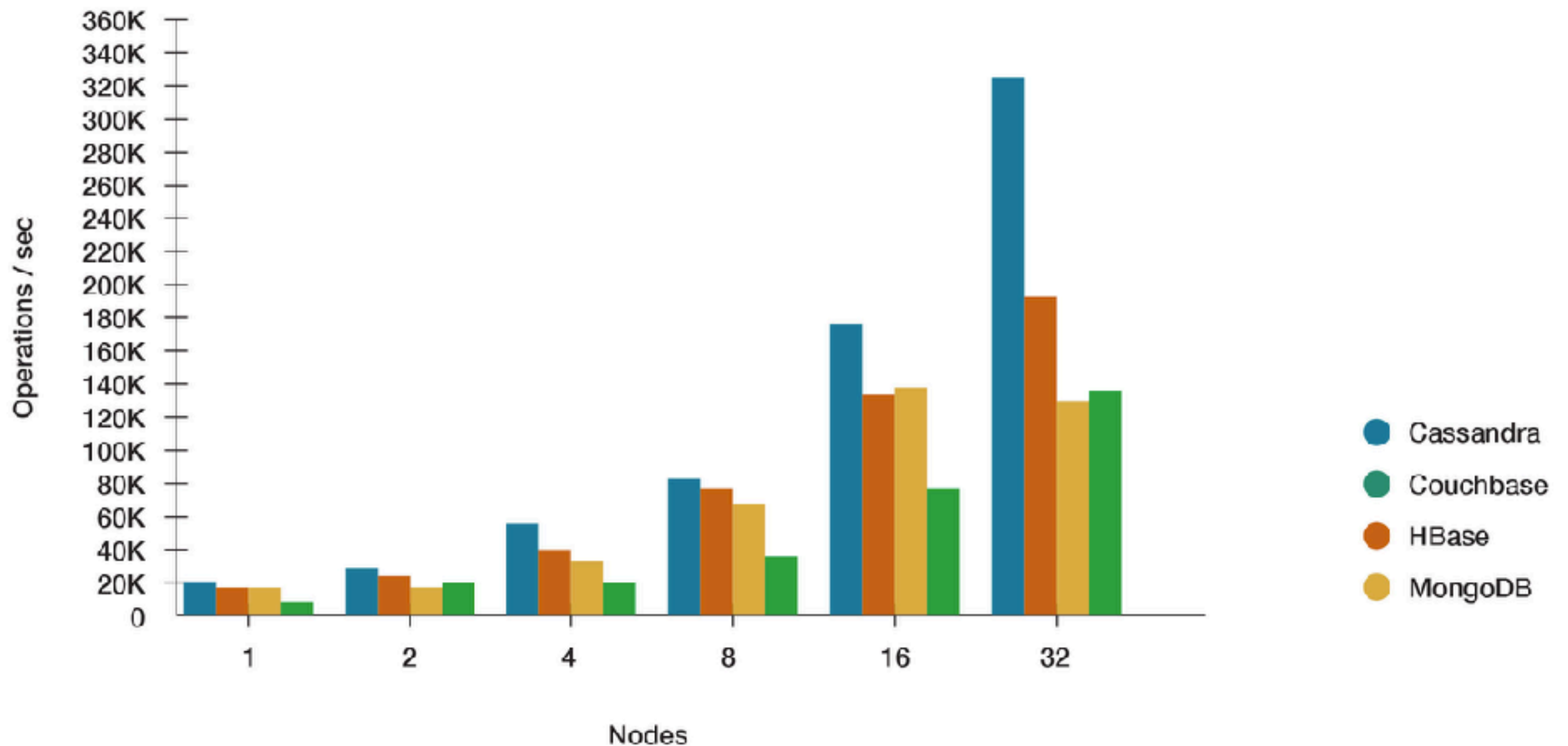
Ebay atualmente possui 100  
nós Apache Cassandra  
armazenando 250Tb de  
dados





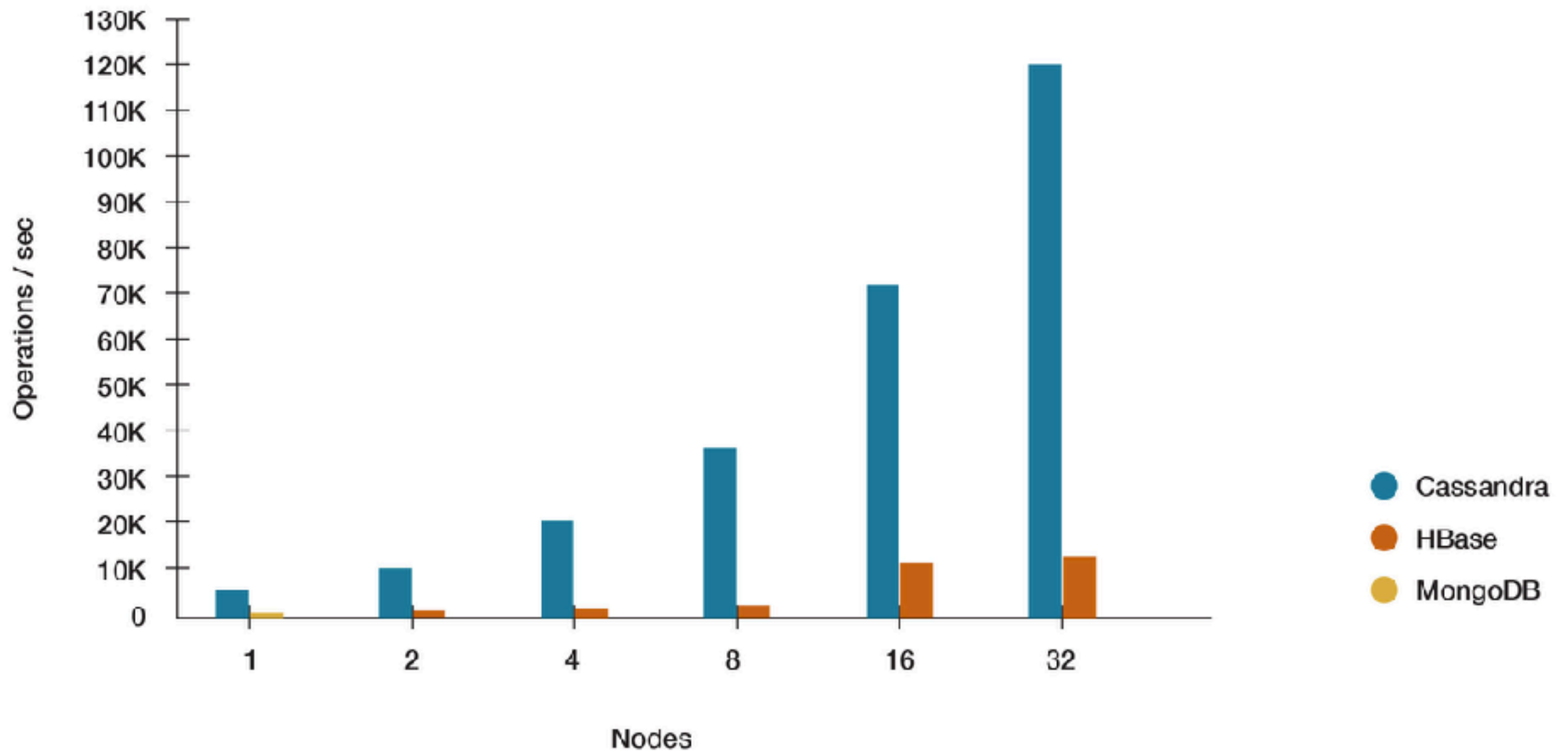
# Benchmarking

Inserting...



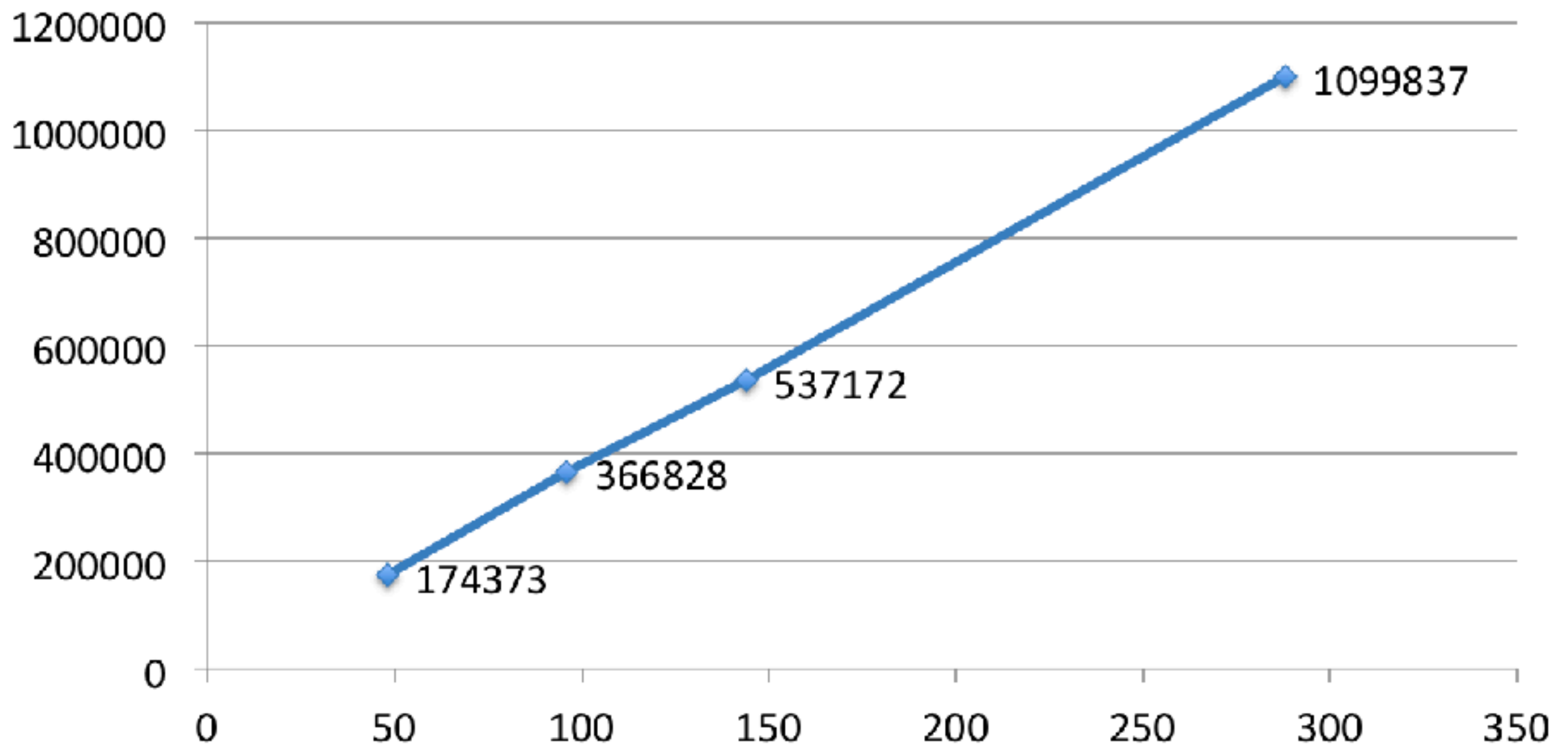
# Benchmarking

Operações diversas



# Scale-Up Linearity

**Client Writes/s by node count – Replication Factor = 3**



# Time is Money

	48 nodes	96 nodes	144 nodes	288 nodes
Writes Capacity	174373 w/s	366828 w/s	537172 w/s	1,099,837 w/s
Storage Capacity	12.8 TB	25.6 TB	38.4 TB	76.8 TB
Nodes Cost/hr	\$32.64	\$65.28	\$97.92	\$195.84
Test Driver Instances	10	20	30	60
Test Driver Cost/hr	\$20.00	\$40.00	\$60.00	\$120.00
Cross AZ Traffic	5 TB/hr	10 TB/hr	15 TB/hr	30 <sup>1</sup> TB/hr
Traffic Cost/10min	\$8.33	\$16.66	\$25.00	\$50.00
Setup Duration	15 minutes	22 minutes	31 minutes	66 <sup>2</sup> minutes
AWS Billed Duration	1hr	1hr	1 hr	2 hr
Total Test Cost	\$60.97	\$121.94	\$182.92	\$561.68

<sup>1</sup> Estimate two thirds of total network traffic

<sup>2</sup> Workaround for a tooling bug slowed setup

# Funcionalidades

- **Escalabilidade:** Cassandra é altamente escalável. Permite adicionar hardware para acomodar mais dados
- **Arquitetura descentralizada:** Cassandra não possui um ponto único de falha. Todos os nós tem as mesmas responsabilidades
- **Escalabilidade Linear:** Cassandra é linearmente escalável. Ou seja, aumenta a vazão de operações linearmente com a quantidade de nós na rede.
- **Estrutura flexível:** Salva diversos tipos de dados sejam eles estruturados ou não. Permite mudanças no modelo dinamicamente

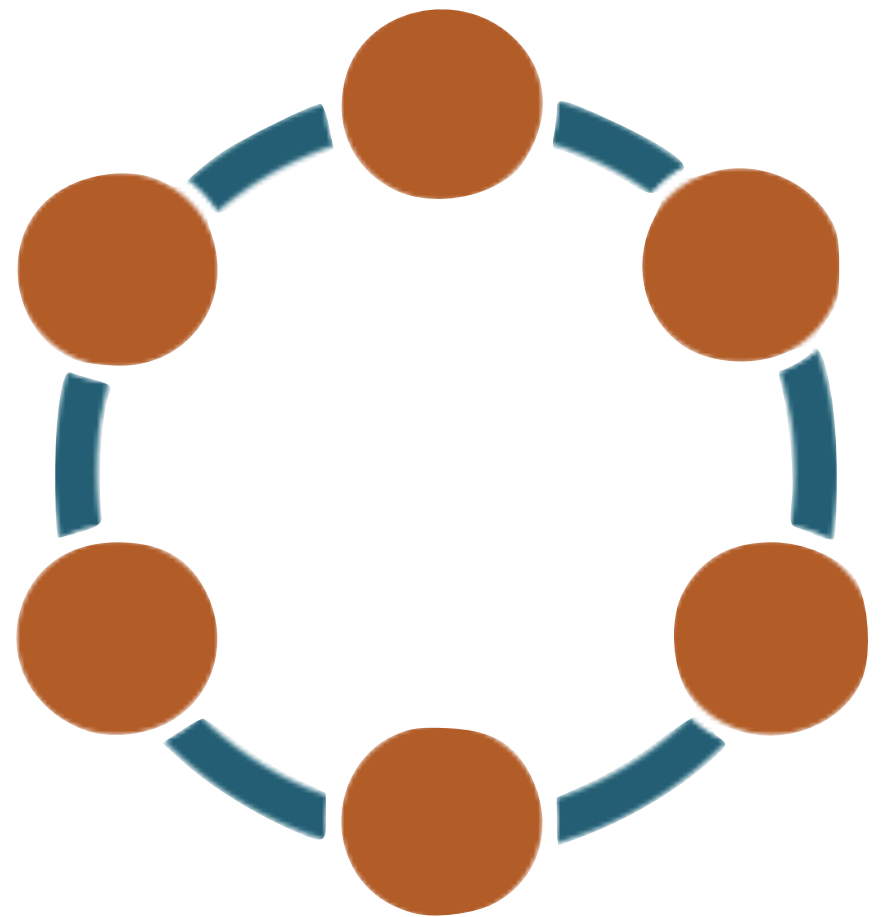
# Funcionalidades

- **Distribuição dos dados:** Os dados são replicados entre os nós do cluster através da definição do fator de replicação.
- **Suporte a transação:** Cassandra suporta a execução de transações (segue princípios ACID)
- **Escrita Rápida:** Cassandra possui escritas rápidas mesmo com hardwares simples sob milhares de dados.

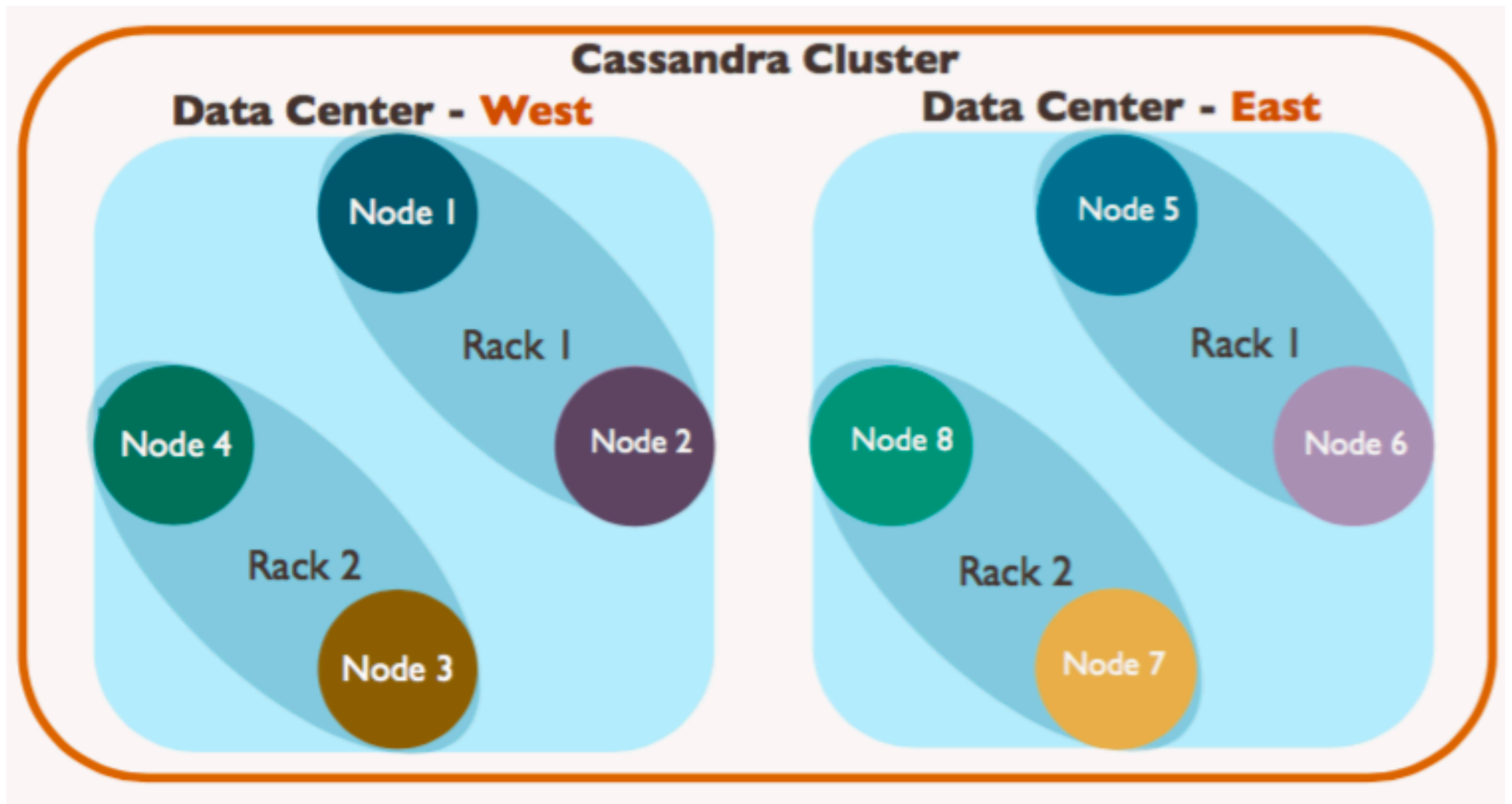


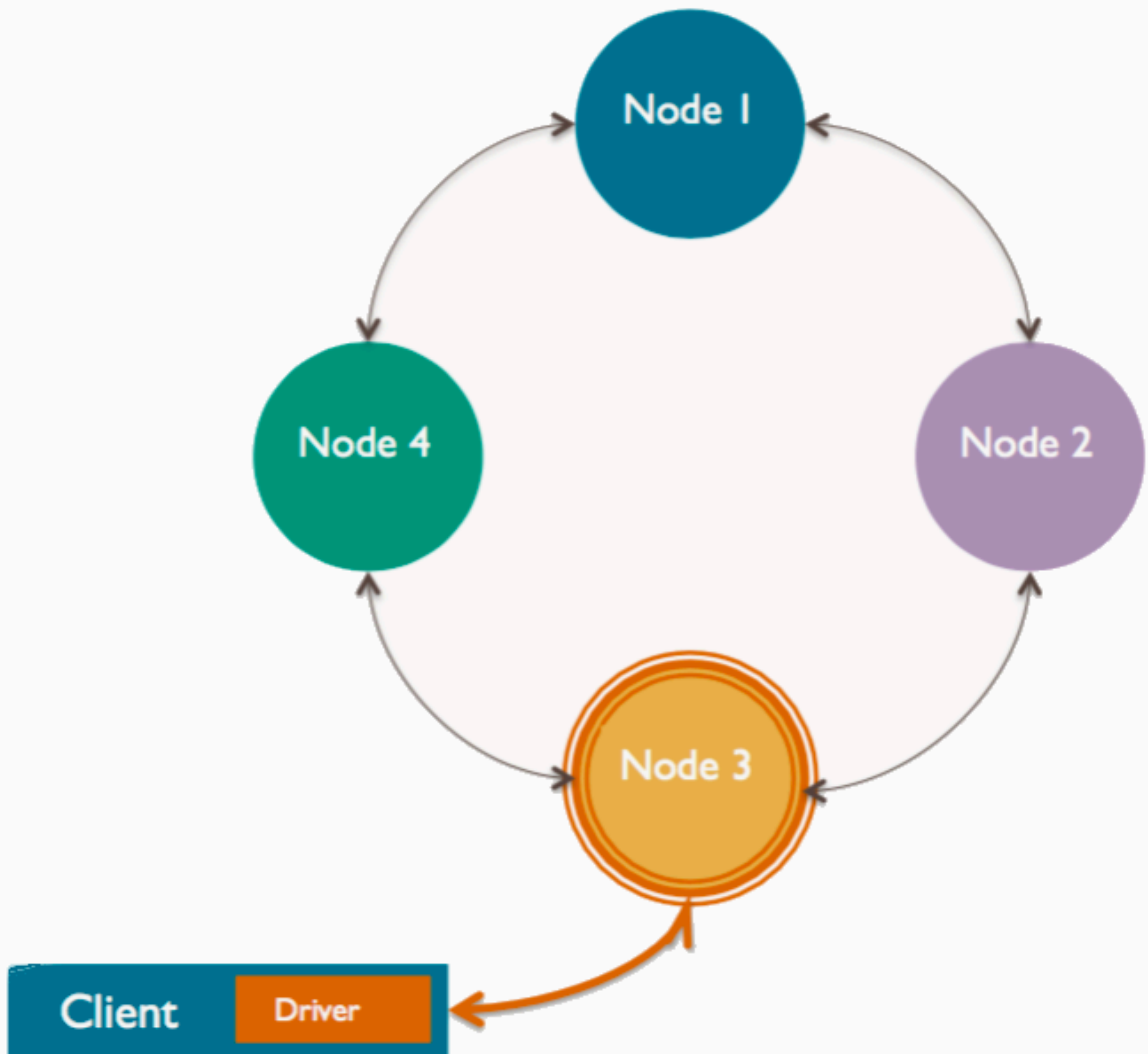
# Arquitetura

- Sistema distribuído p2p
- Todos os nós são iguais
- Os dados são particionados pelos nós
- Replicação de dados customizável para melhorar disponibilidade
- Leia/Escreva em qualquer lugar

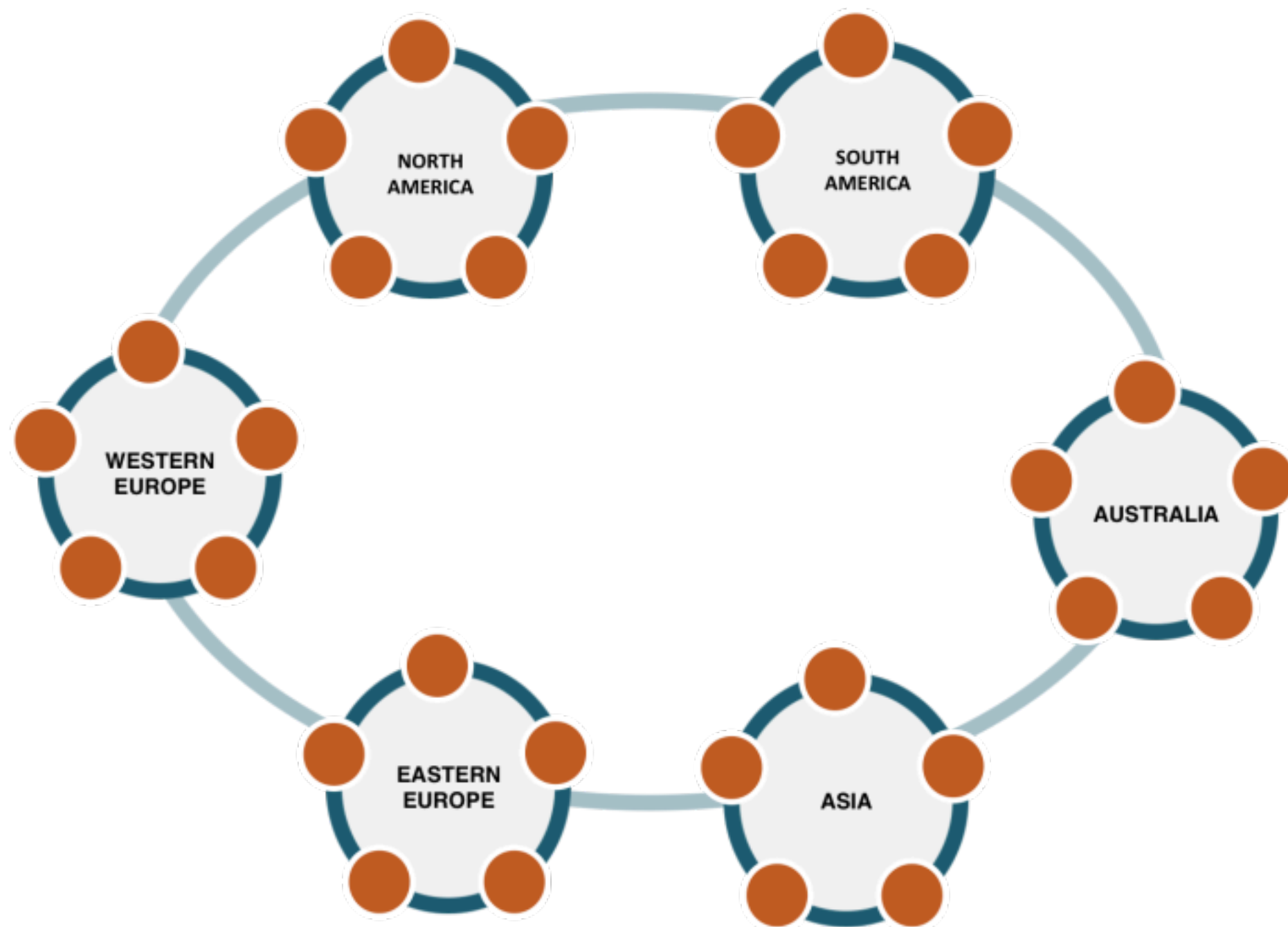


# Arquitetura

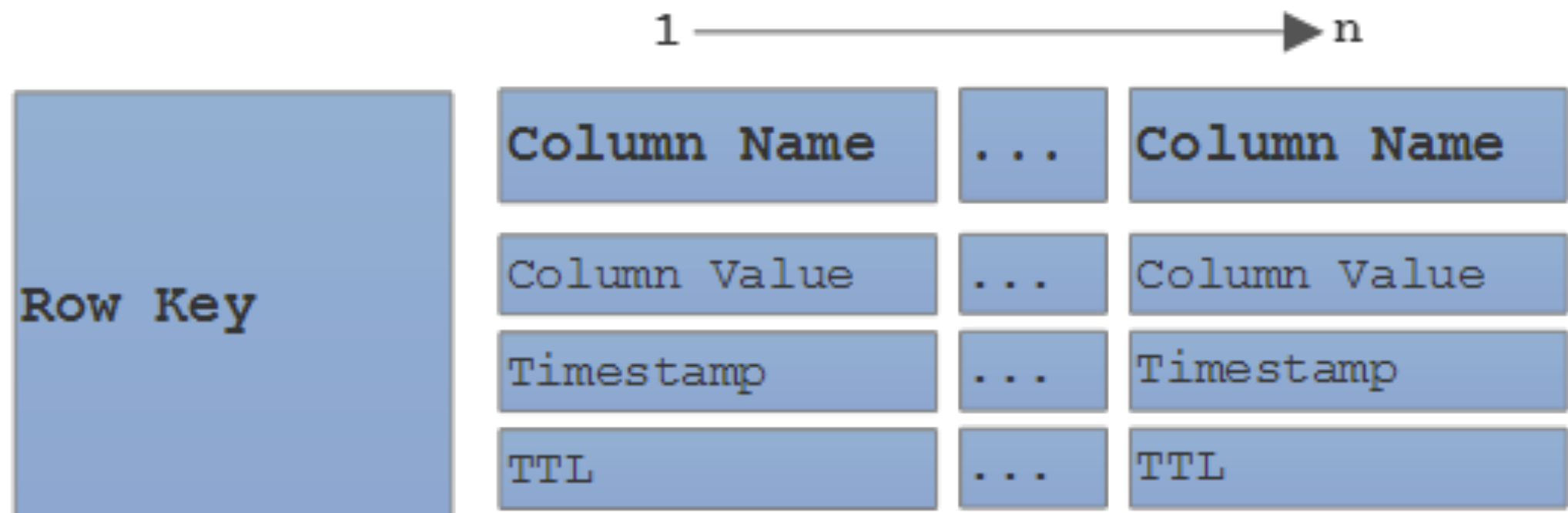




# Arquitetura

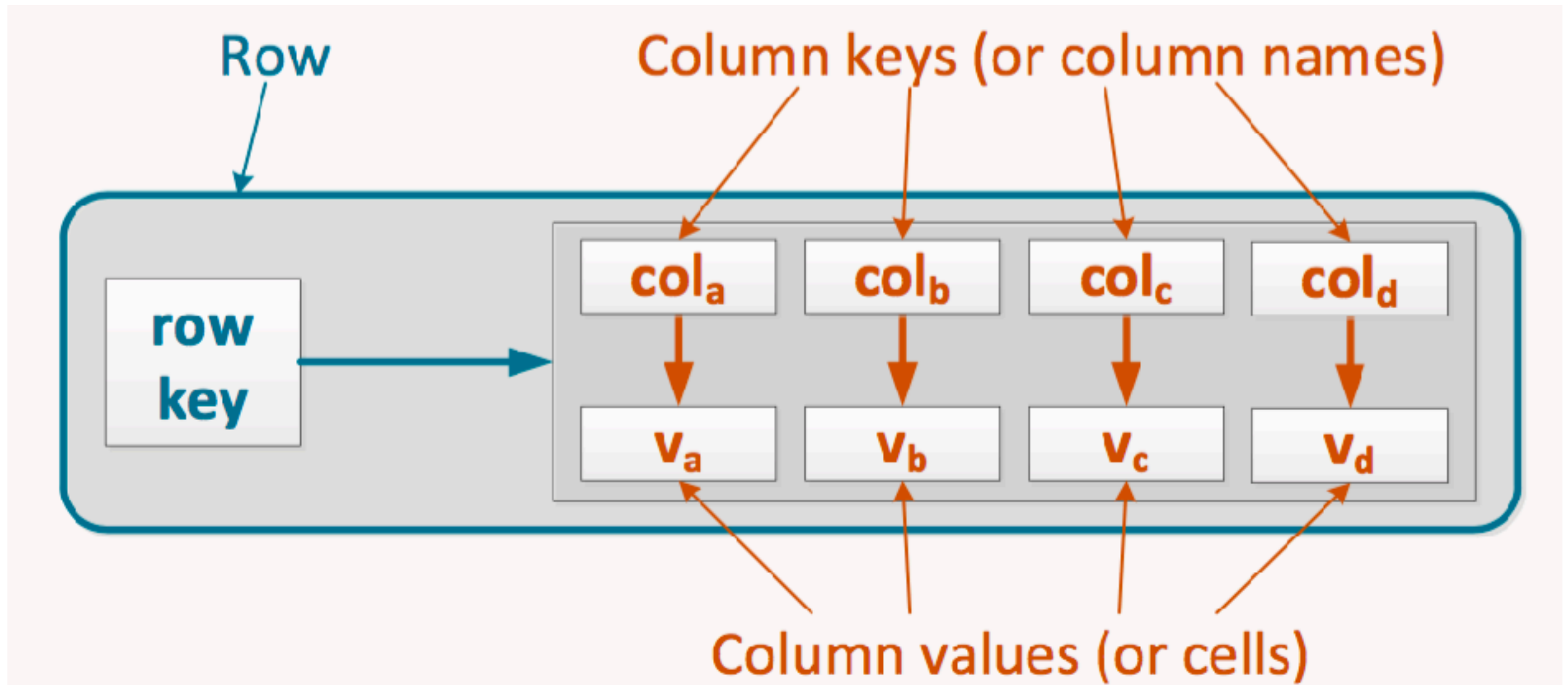


# Modelo de Dados



TTL (Time To Live) - Tempo de vida da coluna

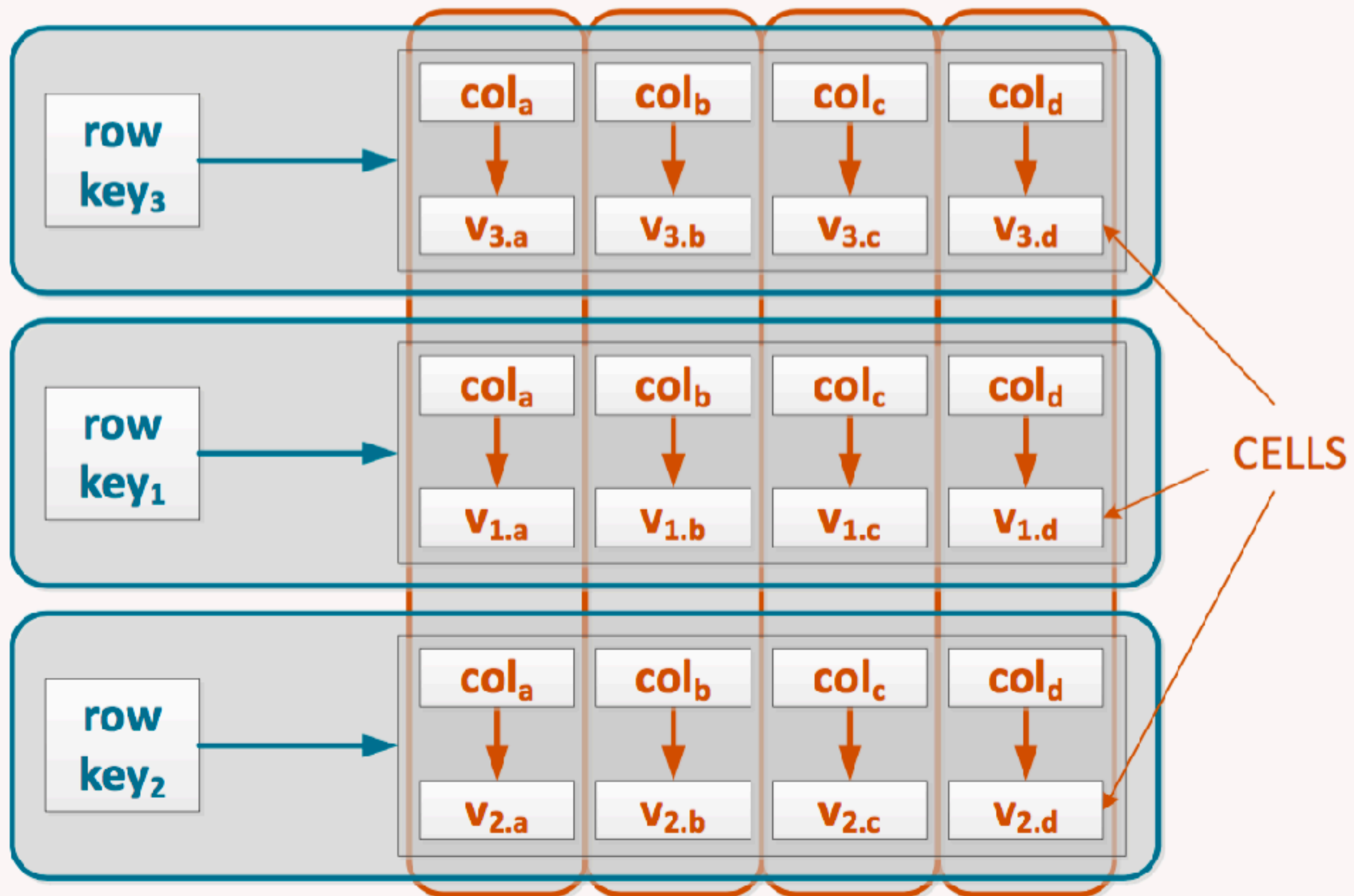
# Modelo de Dados





ROWS

COLUMNS





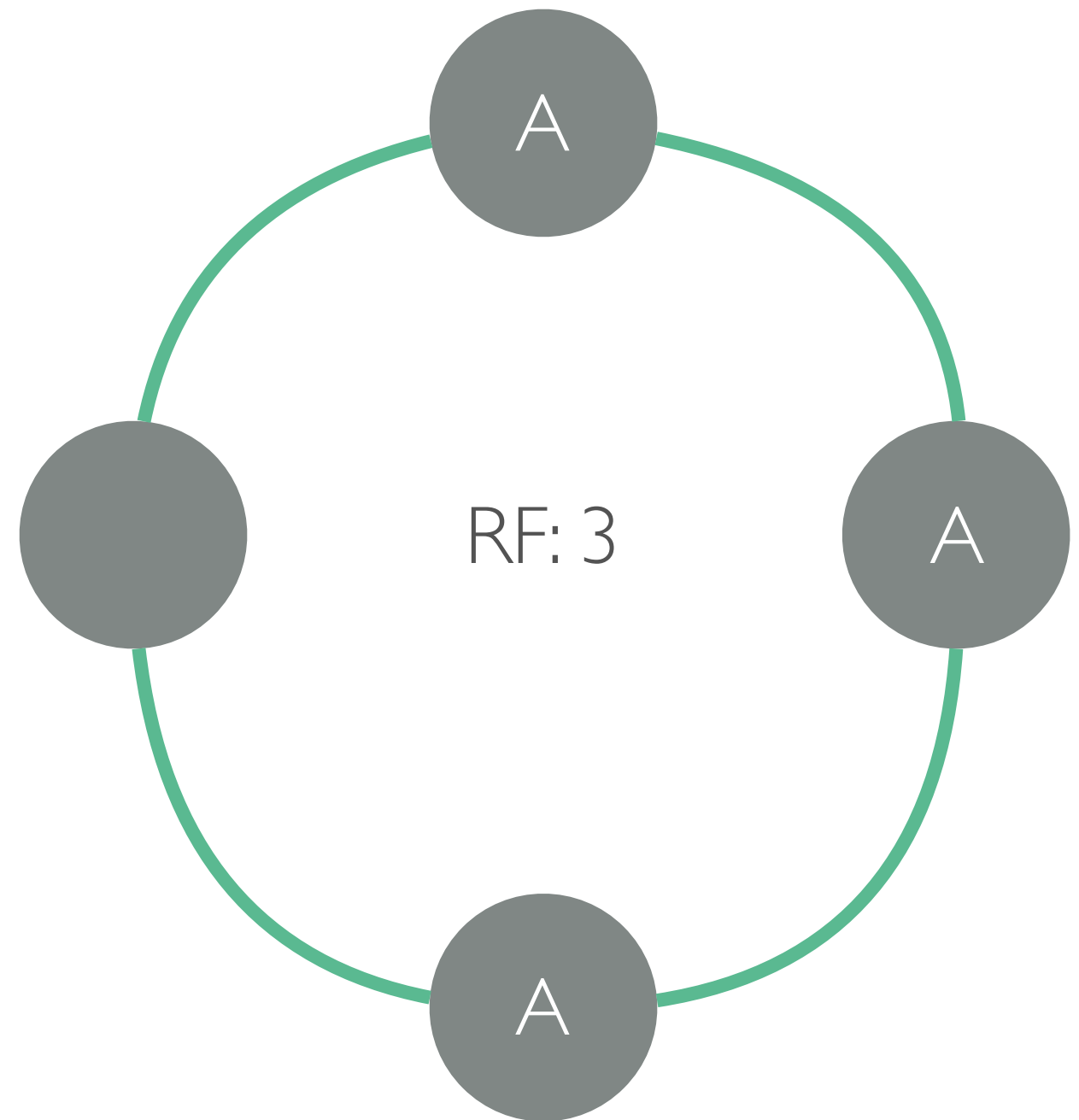
SGBD	Cassandra
Database	Keyspace
Table	Column Family
Linha	Linha
Primary Key	Row Key

# Por que cassandra?

- Altamente escalável
- Ganhos de desempenho linear através da adição de nós
- Nenhum ponto único de falha
- Fácil replicação / distribuição de dados
- Multi-data center e Cloud
- Não há necessidade de camada de cache separada
- Consistência de dados ajustáveis
- Esquema flexível do esquema
- Compressão de dados
- Linguagem CQL (como SQL)
- Não há necessidade de hardware ou software especial

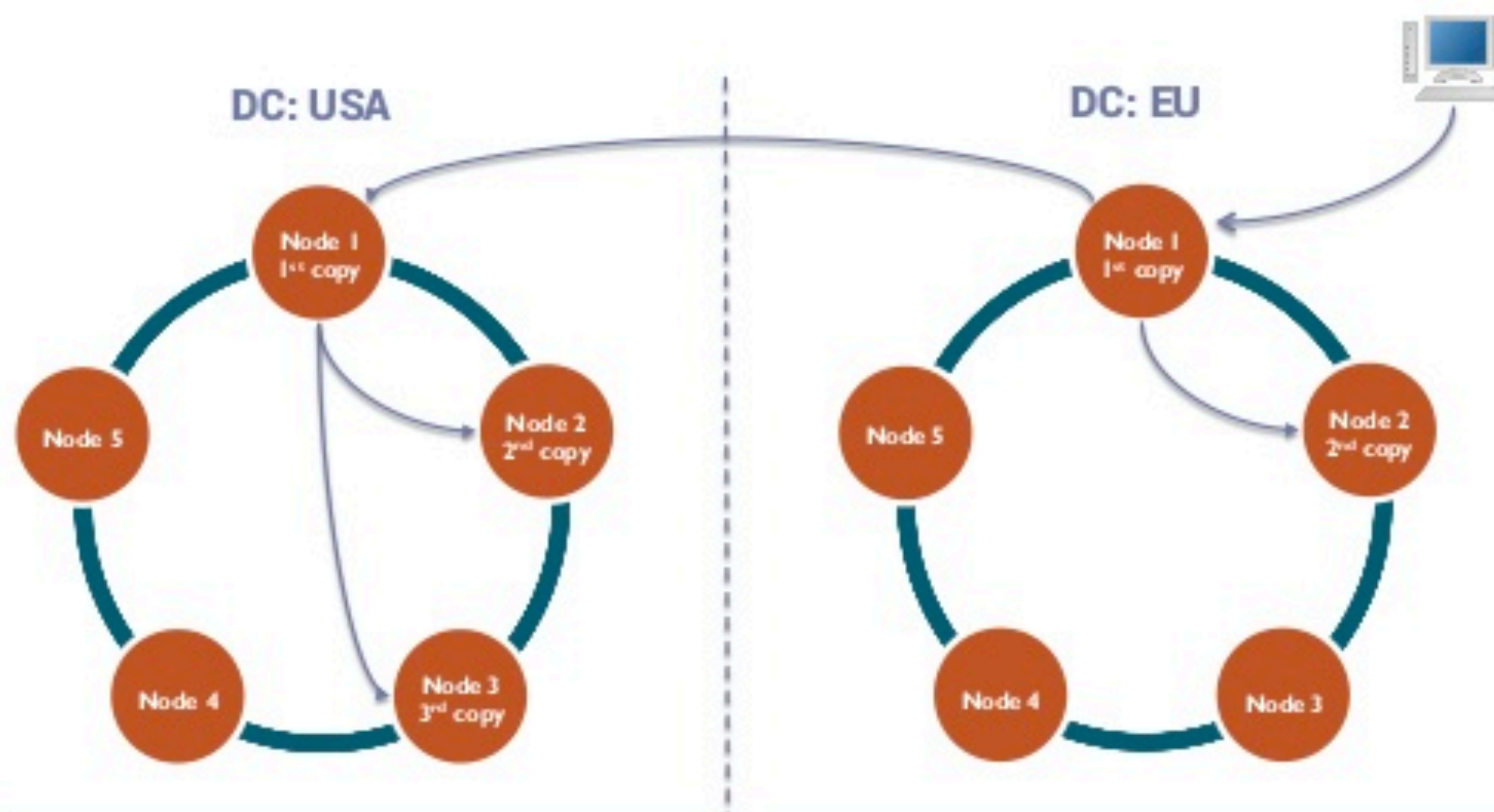
# Replicação

- Quantas cópias de cada parte do dado serão feitas?
- O fator de replicado pode ser definido por data center



# Creating A Keyspace

```
CREATE KEYSPACE johnny WITH REPLICATION =  
{'class':'NetworkTopologyStrategy', 'USA':3, 'EU': 2};
```



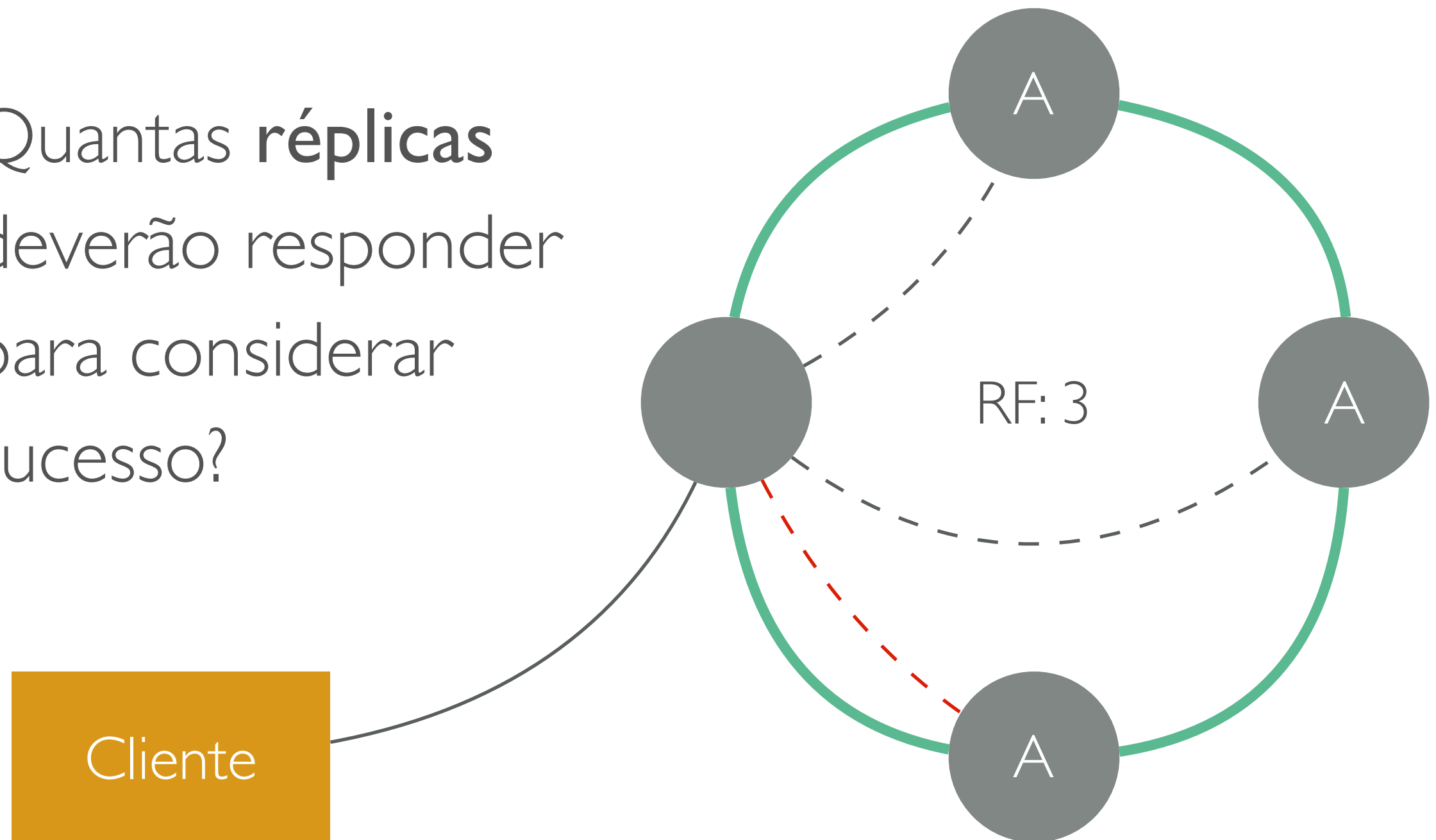
# Estratégias de Replicação

- **SimpleStrategy:** use apenas para um único datacenter e um rack. Se você planeja mais de um datacenter, use o NetworkTopologyStrategy.
- **NetworkTopologyStrategy:** altamente recomendado para a maioria das implantações, porque é muito mais fácil expandir para vários datacenters quando exigido por futuras expansões.



# Consistência

- Quantas **réplicas** deverão responder para considerar sucesso?



# Consistência Escrita

Consistência	Descrição
ANY	A escrita deve retornar sucesso se qualquer nó reportar sucesso
ONE	Pelo menos uma réplica deve retornar sucesso
LOCAL_ONE	Pelo menos uma réplica no mesmo DC deve retornar sucesso
TWO	Pelo menos duas réplicas devem retornar sucesso
THREE	Pelo menos três réplicas devem retornar sucesso
QUORUM	Deve retornar sucesso por um quorum de réplicas $((RF/2)+1)$
LOCAL_QUORUM	Deve retornar sucesso por um quorum de réplicas no mesmo DC
EACH_QUORUM	Deve retornar sucesso por um quorum de réplicas em cada DC
ALL	Todas as replicas devem retornar sucesso

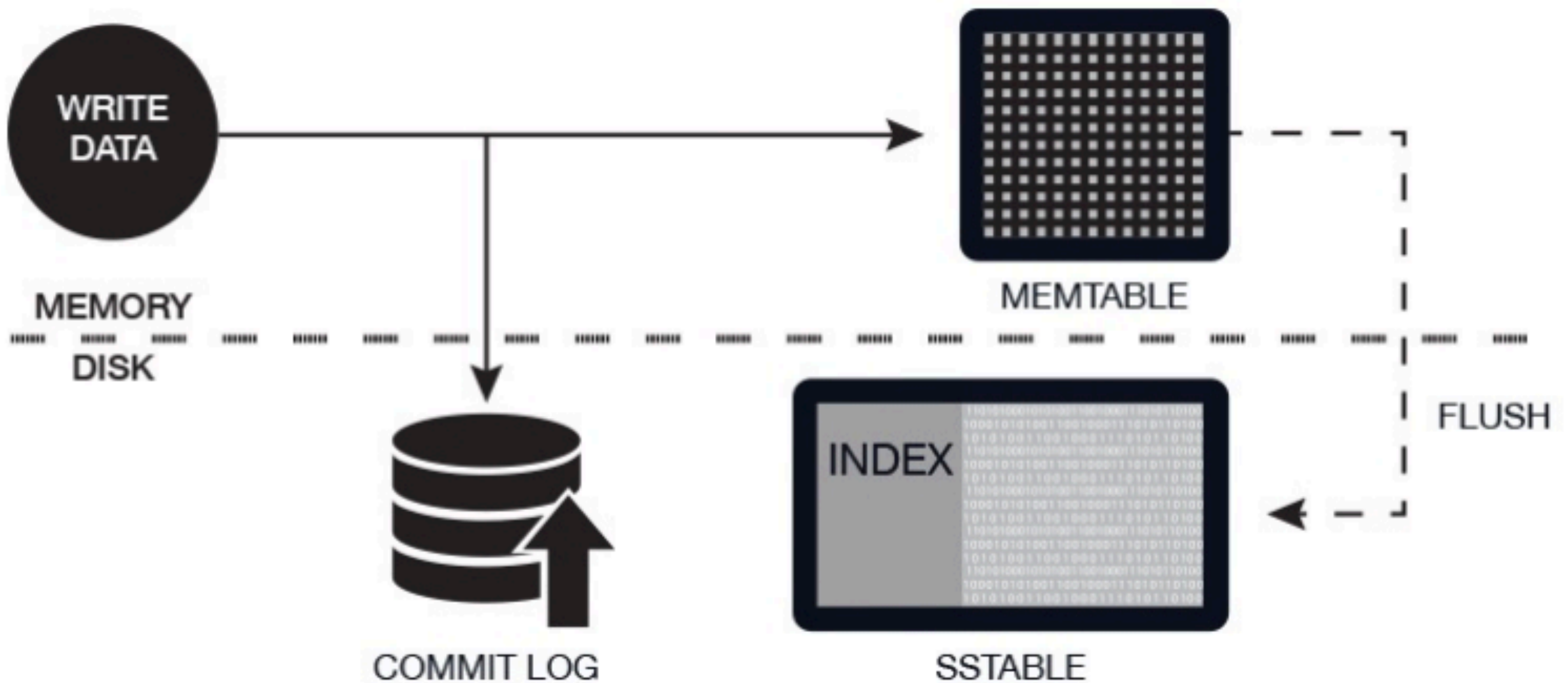
# Consistência Leitura

Consistência	Descrição
ONE	Retorna a resposta da réplica mais próxima (em seguida read-repair)
LOCAL_ONE	Retorna a resposta da réplica mais próxima no mesmo DC
TWO	Retorna o dado mais recente de duas das réplicas
THREE	Retorna o dado mais recente de três das réplicas
QUORUM	Retorna depois de obter resposta de um quorum de qualquer DC
LOCAL_QUORUM	Retorna depois de obter resposta de um quorum no DC
ALL	Retorna após todas as replicas responderem

# Componentes

- **Nó:** onde os dados são armazenados. Case da arquitetura.
- **Datacenter:** Coleção de nós relacionados
- **Cluster:** Contém um ou mais datacenters. Os datacenters podem estar fisicamente separados
- **Commit log:** Todos os dados são salvos inicialmente no commit log para garantir durabilidade. Depois os dados são salvos no SSTable para arquivamento.
- **SSTable:** Sorted String Table é um arquivo imutável onde o Cassandra armazena os dados periodicamente.

# Escrita



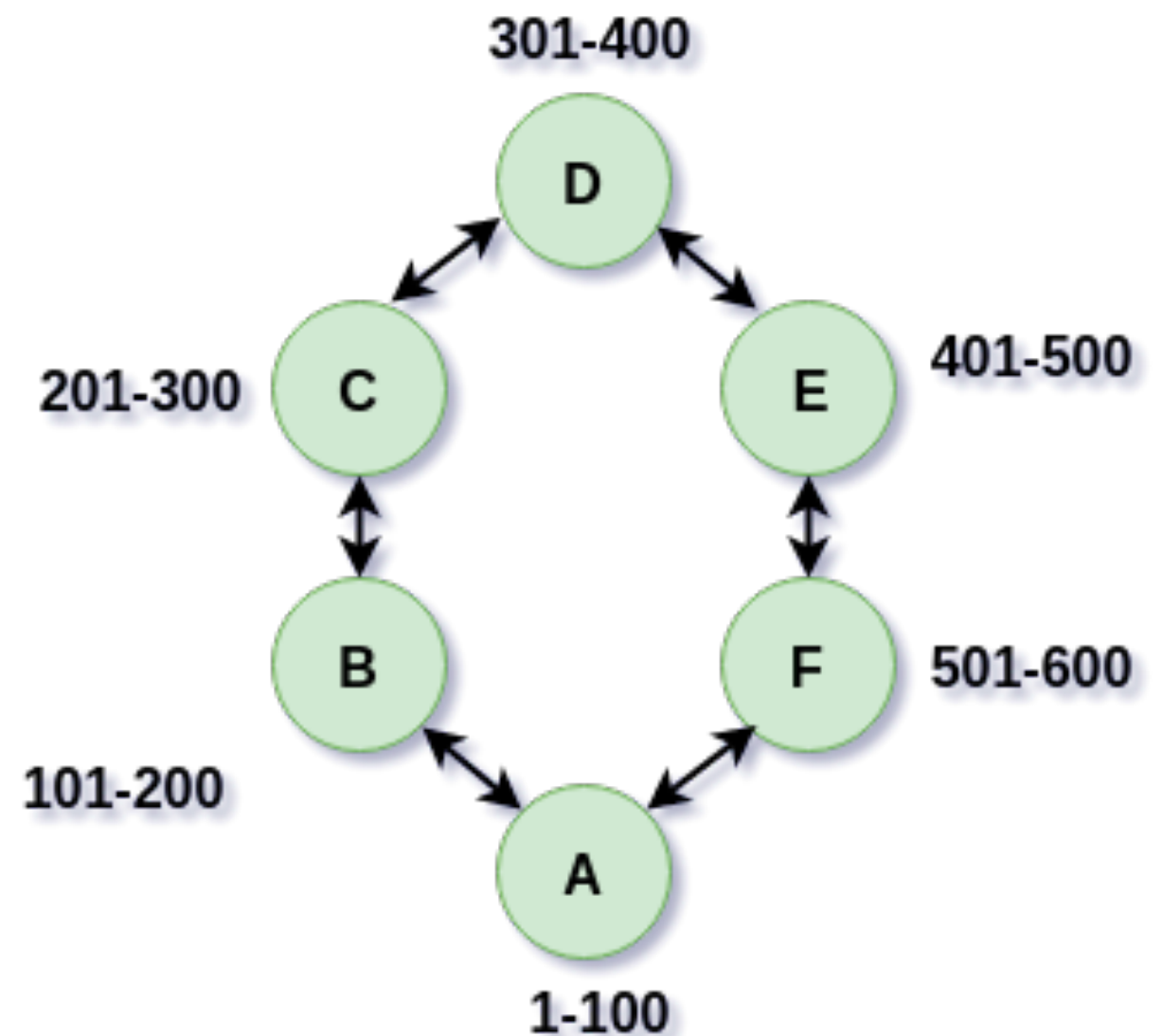
# CONCEITOS

- Partition Key —> Como os dados vão ser distribuídos. Cada partição residirá em um nó. Limite de 2 bilhões de colunas por partição.
- Cluster order —> Controla a ordem (ASC ou DESC) de como os dados dentro de uma partição serão salvos no disco.



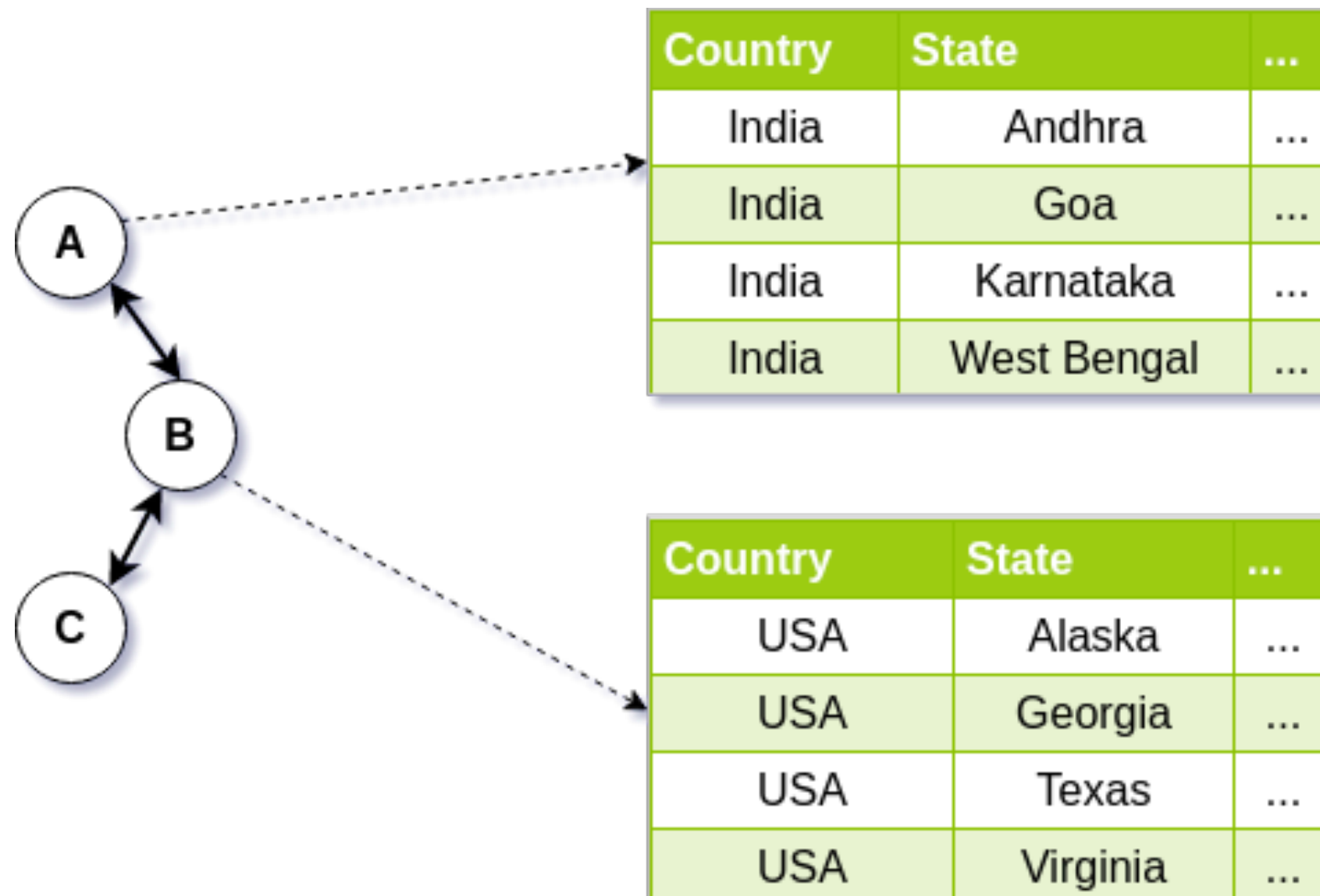
# PARTITION KEY

- Quando um dado é inserido, uma função hash da partition key é feita para decidir em qual nó o dado residirá.



```
CREATE TABLE store_by_location (  
  country text,  
  state text,  
  city text,  
  store_name text,  
  PRIMARY KEY (country, state)  
);
```

2



# Instalação



<https://www.docker.com/>

# INSTALAÇÃO

Crie uma rede

```
docker network create cassandra-net
```



# INSTALAÇÃO

Instalando o Cassandra (Instância simples)

```
docker run --name cassandra1 --network cassandra-net -d  
cassandra
```



# INSTALAÇÃO

Iniciando o prompt do cassandra CQLSH

```
$ docker run -it --network cassandra-net --rm  
cassandra cqlsh cassandra1
```



# INSTALAÇÃO

<https://www.datastax.com/get-started#two>



Solutions

Products

Partners

Resources

Company

Get Started

Support Academy Documentation

## BE THE HERO YOUR STACK DESERVES

Follow these simple steps to get your first query going now. Get up and running with DataStax today!



# BE THE HERO YOUR STACK DESERVES

Follow these simple steps to get your first query going now. Get up and running with DataStax today!

```
$ docker run -e DS_LICENSE=accept --name my-dse -d  
datastax/dse-server -g -s -k
```

```
$ docker run -e DS_LICENSE=accept --link my-dse -p  
9091:9091 --memory 1g --name my-studio -d datastax/  
dse-studio
```



## EDIT CONNECTION

**Name \***

**Username**

**Host/IP (comma delimited) \***

**Password**

**Port \***

☐ Use SSL

Save

Cancel

Test

# Primeiros Comandos (CQL)

# CQL

- CQL (Cassandra Query Language) é uma linguagem baseada no SQL para operações no Apache Cassandra

# CQL

## Criando um keyspace

```
CREATE KEYSPACE | SCHEMA IF NOT EXISTS keyspace_name  
WITH REPLICATION = map  
AND DURABLE_WRITES = true | false
```

Exemplo para estratégia simples

```
CREATE KEYSPACE Excelsior  
WITH REPLICATION = { 'class' : 'SimpleStrategy', 'replication_factor' :  
3 };
```

# CQL

## Criando um keyspace

Exemplo para estratégia de data center

```
CREATE KEYSPACE NTSkeyspace WITH REPLICATION = { 'class' : 'NetworkTopologyStrategy', 'datacenter1' : 1 };
```

Exemplo para estratégia de data center

```
CREATE KEYSPACE "Excalibur"  
  WITH REPLICATION = { 'class' : 'NetworkTopologyStrategy', 'dc1' : 3, 'dc2' : 2 };
```

# CQL

## Alterand um keyspace

```
ALTER KEYSPACE | SCHEMA keyspace_name  
WITH REPLICATION = map  
| WITH DURABLE_WRITES = true | false  
AND DURABLE_WRITES = true | false
```

Exemplo de alteração

```
ALTER KEYSPACE "Excalibur" WITH REPLICATION =  
{ 'class' : 'NetworkTopologyStrategy', 'datacenter1' : 3 };
```

# CQL

## Criando uma tabela

```
CREATE TABLE IF NOT EXISTS keyspace_name.table_name  
( column_definition, column_definition, ... )  
WITH property AND property ...
```

## Exemplo de criação de tabela

```
CREATE TABLE users (  
    user_name varchar PRIMARY KEY,  
    password varchar,  
    gender varchar,  
    session_token varchar,  
    state varchar,  
    birth_year bigint  
);
```



# CQL

## Criando uma tabela

Criando tabela com chave primária composta

```
CREATE TABLE emp (  
    empID int,  
    deptID int,  
    first_name varchar,  
    last_name varchar,  
    PRIMARY KEY (empID, deptID)  
);
```

# CQL

## Criando uma tabela

Criando tabela com chave primária composta

```
CREATE TABLE emp (  
  empID int,  
  deptID int,  
  first_name varchar,  
  last_name varchar,  
  PRIMARY KEY (empID, deptID)  
);
```

# CQL

## Criando uma tabela

Criando tabela com chave partition key composta

```
CREATE TABLE Cats (  
  block_id uuid,  
  breed text,  
  color text,  
  short_hair boolean,  
  PRIMARY KEY ((block_id, breed), color, short_hair)  
);
```

Partition key

# CQL

## Criando uma tabela

Criando tabela com clustering order

```
create table timeseries (  
    event_type text,  
    insertion_time timestamp,  
    event blob,  
    PRIMARY KEY (event_type, insertion_time)  
)  
WITH CLUSTERING ORDER BY (insertion_time DESC);
```

# CQL

## Inserindo Dados

```
INSERT INTO keyspace_name.table_name  
  ( identifier, column_name... )  
VALUES ( value, value ... ) IF NOT EXISTS  
USING option AND option
```

## Exemplo com TTL

```
INSERT INTO Hollywood.NerdMovies (user_uuid, fan)  
  VALUES (cfd66ccc-d857-4e90-b1e5-df98a3d40cd6, 'johndoe')  
USING TTL 86400;
```

TTL: tempo de vida em segundos

# CQL

## Inserindo Dados

Insere apenas se não existe

```
INSERT INTO cycling.cyclist_name (id,lastname,firstname)  
  VALUES( 'c4b65263-fe58-83e8-f0e1c13d518f', 'RATTO', 'Risselda') IF NOT E  
XISTS;
```

# CQL

## Atualizando Dados

Atualizando várias linhas

```
UPDATE users
SET state = 'TX'
WHERE user_uuid
IN (88b8fd18-b1ed-4e96-bf79-4280797cba80,
      06a8913c-c0d6-477c-937d-6c1b69a95d43,
      bc108776-7cb5-477f-917d-869c12dffa8);
```

Atualizando várias colunas de uma linha

```
UPDATE users
SET name = 'John Smith',
    email = 'jsmith@cassie.com'
WHERE user_uuid = 88b8fd18-b1ed-4e96-bf79-4280797cba80;
```

# CQL

## Removendo Dados

Deletando apenas algumas colunas

```
DELETE firstname, lastname FROM cycling.cyclist_name WHERE firstname = 'Alex';
```

Deletando toda a linha

```
DELETE FROM cycling.cyclist_name WHERE firstname IN ('Alex', 'Marianne');
```



# CQL

## Seleccionando Datos

Seleccionando datos

```
SELECT select_expression  
  FROM keyspace_name.table_name  
  WHERE relation AND relation ...  
  ORDER BY ( clustering_column ASC | DESC ... )  
  LIMIT n  
ALLOW FILTERING
```

# CQL

## Seleccionando Datos

Seleccionando todos registros:

```
SELECT * from People;
```

Contando número de elementos

```
SELECT COUNT(*) FROM users;
```

# CQL

## Seleccionando Datos

Seleccionando apenas algunas columnas:

```
SELECT event_id, dateOf(created_at), blobAsText(content) FROM timeline;
```

Limitando o resultado

```
SELECT COUNT(*) FROM big_table LIMIT 50000;  
SELECT COUNT(*) FROM big_table LIMIT 200000;
```



***cassandra***

## Banco de dados NoSQL - Cassandra (Introdução)

Prof. Gustavo Leitão