

# Seminário sobre Banco de Dados NoSQL Distribuído: Apache Cassandra

Felipe Erik

Pedro Vitor

# Origem e Contexto Histórico

- 1** — 2008: Nasce no Facebook  
Criado para gerenciar a busca na caixa de entrada, inspirando-se em **DynamoDB da Amazon** e **BigTable do Google**.
- 2** — 2010: Open Source  
Torna-se um projeto **open source** sob a Apache Software Foundation, ganhando a comunidade.
- 3** — Nome Mítico  
Seu nome é uma homenagem à profetisa **Cassandra da mitologia**, simbolizando a capacidade de prever e superar desafios complexos de dados.
- 4** — Evolução Contínua  
Desde então, Cassandra tem evoluído, tornando-se um dos bancos de dados distribuídos mais resilientes e amplamente adotados.



Caseanbaray dot

# O Que é Apache Cassandra?



## Banco de Dados NoSQL

Um sistema de gerenciamento de banco de dados NoSQL, distribuído, **orientado a colunas** e altamente escalável.



## Arquitetura Distribuída

Projetado para operar em **clusters com múltiplos nós**, eliminando pontos únicos de falha e garantindo resiliência.



## Open Source e Adotado Globalmente

Código aberto, utilizado por gigantes como **Apple, Netflix, Twitter e eBay** para suas operações críticas.



## Replicação Multi Data Center

Suporta **replicação entre múltiplos data centers** para alta disponibilidade e tolerância a falhas, essencial para negócios globais.

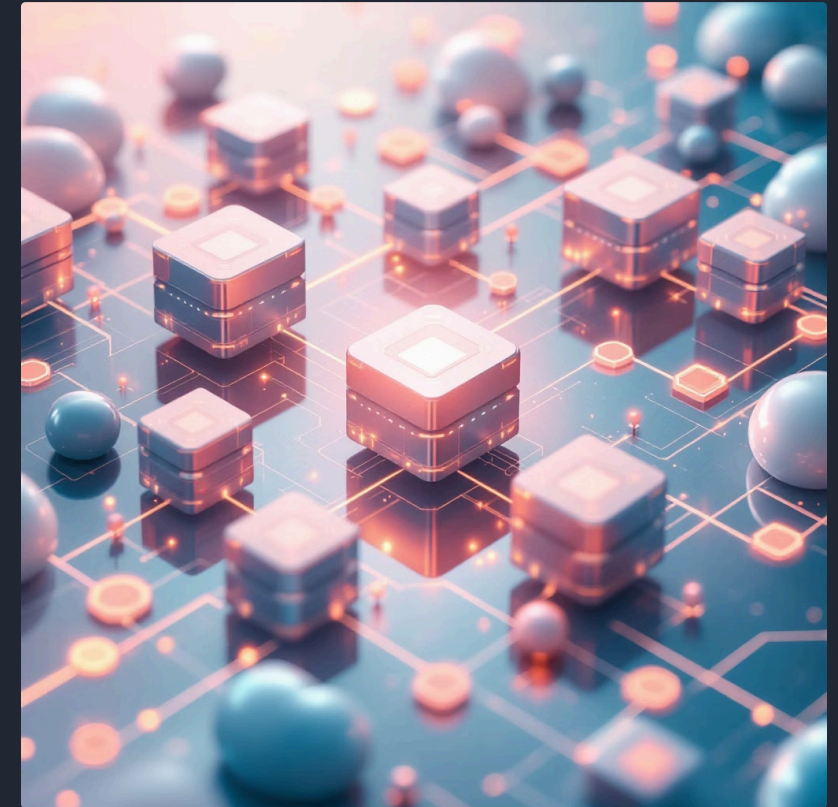




# Arquitetura Distribuída e Escalabilidade

O coração do Cassandra é sua **arquitetura distribuída**. Um cluster é formado por nós iguais, operando em um modelo **peer-to-peer**, o que significa que não há um "nó mestre" único, eliminando pontos de falha isolados. Os dados são automaticamente particionados e replicados entre os nós, garantindo que mesmo se um nó falhar, o serviço continue ininterrupto.

Sua **escalabilidade elástica** é um diferencial: adicione ou remova nós conforme a demanda sem nenhuma interrupção do serviço. Imagine uma empresa de e-commerce que precisa suportar um aumento de 10x no volume de transações durante a **Black Friday** sem qualquer degradação de performance – Cassandra torna isso possível.



## Escalabilidade Elástica

Adicione ou remova nós sem **interrupção do serviço**.

# Modelo de Dados e Linguagem CQL



## Modelo Orientado a Colunas

Cassandra utiliza um modelo de dados baseado em **linhas particionadas com famílias de colunas flexíveis**. Ao contrário dos bancos relacionais, cada linha pode ter um conjunto diferente de colunas.



## CQL: SQL-like

A **Cassandra Query Language (CQL)** possui uma sintaxe muito similar ao SQL, facilitando a curva de aprendizado para desenvolvedores familiarizados com bancos de dados relacionais.

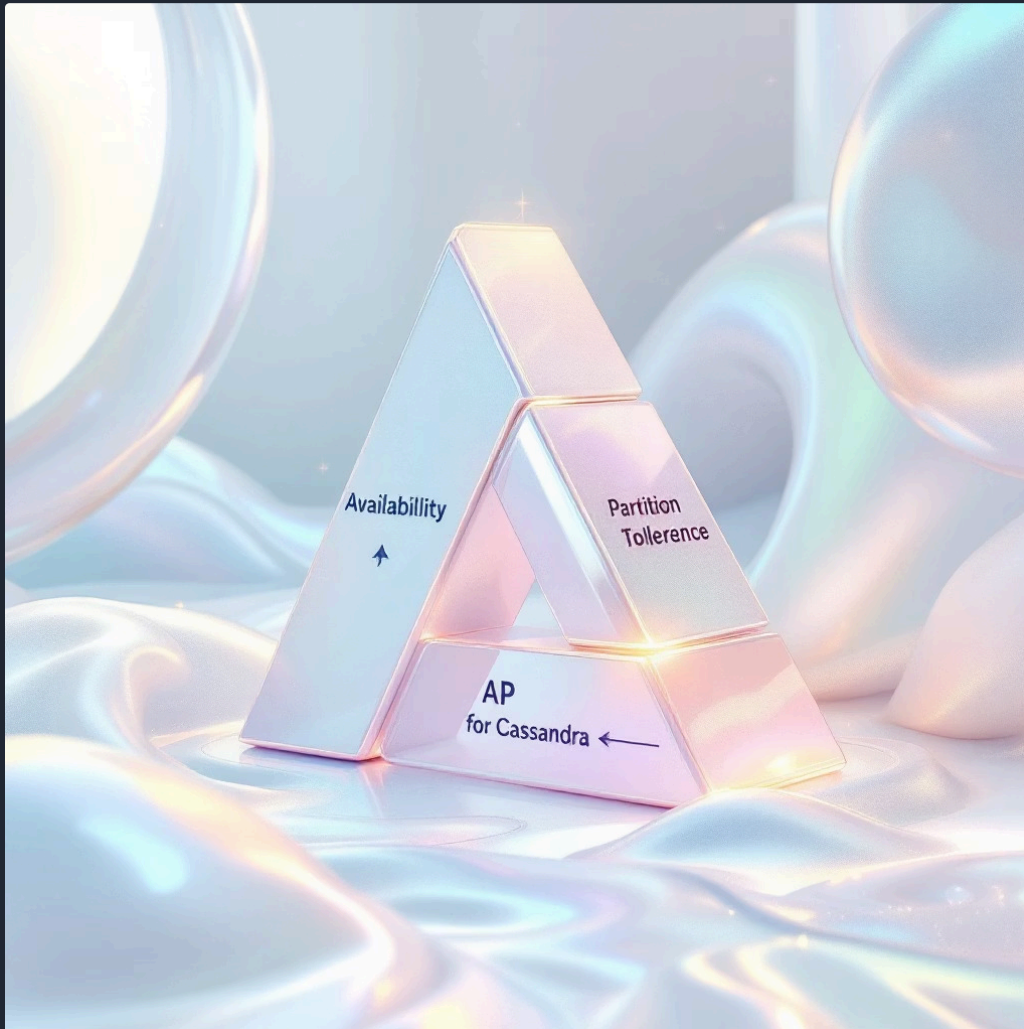


## Consultas Otimizadas

Este modelo permite uma **modelagem eficiente** para realizar consultas rápidas e precisas em volumes massivos de dados, otimizado para cenários de Big Data.

```
{(
flexible column families;
& flexible column languages)
flexible column families.
flexible communities;
flexible columnial);
onferencia enery,
tables;
)+
```

# Consistência e Teorema CAP



No contexto do **Teorema CAP**, Cassandra prioriza **Disponibilidade (A) e Tolerância a Partições (P)**. Isso significa que o sistema permanece operacional mesmo em falhas de rede, sacrificando a consistência imediata em favor da acessibilidade contínua.

## Consistência Ajustável

Cassandra oferece **consistência ajustável (tuneable consistency)**, permitindo que os desenvolvedores equilibrem performance e integridade dos dados conforme a necessidade da aplicação.

Com o **isolamento em nível de linha**, Cassandra minimiza inconsistências em atualizações simultâneas, tornando-o ideal para aplicações que demandam **alta disponibilidade**, mesmo em cenários de falhas de rede.

📌 Perfeito para sistemas que não podem parar!



# Casos de Uso Reais e Impacto no Mercado



## Netflix: Streaming Global

Gerencia milhões de usuários simultâneos e dados de streaming em tempo real, garantindo uma experiência contínua.



## Apple e eBay: Dados Distribuídos

Essencial para o gerenciamento de grandes volumes de dados distribuídos globalmente e alta taxa de transações.



## IoT, E-commerce e Finanças

Aplicado em IoT, e-commerce, análise em tempo real e sistemas financeiros, onde a velocidade e a resiliência são cruciais.

# Desafios e Considerações Técnicas

## Modelagem de Dados

Exige planejamento cuidadoso para evitar consultas ineficientes e garantir performance em Big Data.



## Gerenciamento de Tombstones

A gestão de "tombstones" (registros excluídos logicamente) é crucial para evitar degradação de performance ao longo do tempo.



## Monitoramento Constante

A saúde do cluster depende de monitoramento proativo para identificar e resolver problemas rapidamente.



## Curva de Aprendizado

Dominar a consistência ajustável e as políticas de replicação requer tempo e experiência.



# Principais conceitos

## Cluster

Um cluster é o conjunto completo de nós (servidores) que formam o Cassandra.  
Ele é o "todo" do banco distribuído.

## Nó (Node)

É uma instância do Cassandra.  
Cada nó armazena parte dos dados e se comunica com os demais.

## Keyspace

É o equivalente ao schema do SQL.  
Define o "escopo" de tabelas e políticas de replicação.

## Tabela

Similar às tabelas SQL, mas com estrutura flexível de colunas.

## Partição

É a unidade principal de organização dos dados.  
Cada partição é identificada pela partition key, que define em qual nó os dados serão armazenados no cluster.

# Prática

`docker compose up -d` (Subir o cassandra)

`docker logs -f cassandra` (Verificar se está rodando)

`docker exec -it cassandra cqlsh` (Acessando o cqlsh - Shell do Cassandra)

`CREATE KEYSPACE loja WITH replication = { 'class': 'SimpleStrategy',  
'replication_factor': 1 };` (Criar um KeySpace)

`USE loja;` (Usar KeySpace)

`CREATE TABLE produtos ( id_produto UUID PRIMARY KEY, nome TEXT,  
categoria TEXT, preco DECIMAL );` (Criar uma tabela)

`INSERT INTO produtos (id_produto, nome, categoria, preco) VALUES (uuid(),  
'Notebook', 'Eletrônicos', 3500.00);` (Inserir os dados)

`SELECT * FROM produtos;` (Consultar produtos)

```
1  version: '3.8'
2  services:
3    cassandra:
4      image: cassandra:4.1
5      container_name: cassandra
6      ports:
7        - "9042:9042"
8      environment:
9        CASSANDRA_CLUSTER_NAME: "Me
        uCluster"
10       CASSANDRA_NUM_TOKENS: 256
11       CASSANDRA_DC: "DC1"
12       CASSANDRA_RACK: "Rack1"
13     volumes:
14       - ./data:/var/lib/cassandra
15
```