



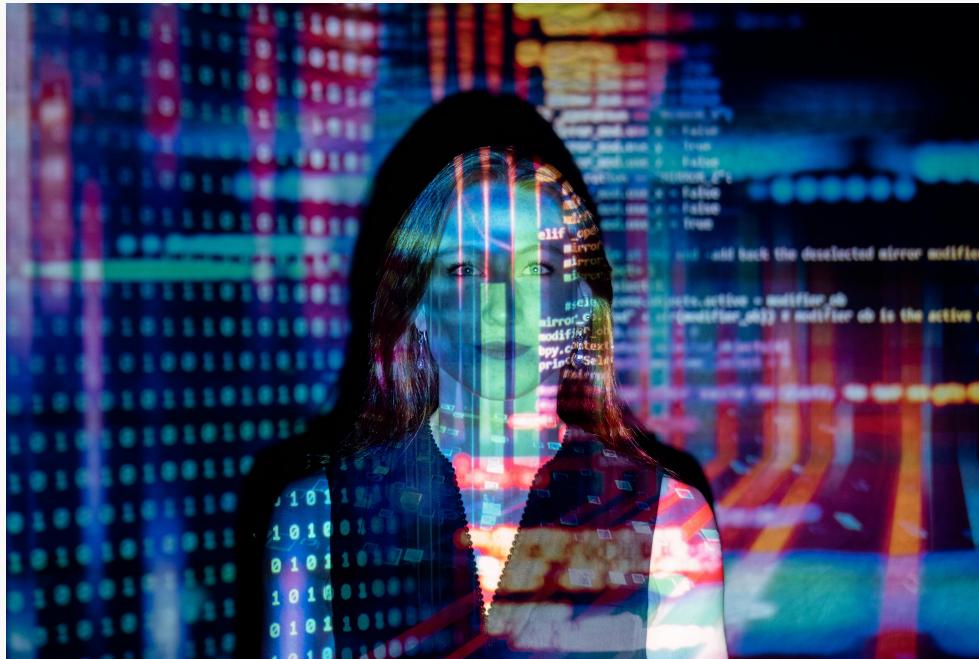
Frameworks de Big Data

UNIDADE 03

Explorando Frameworks que Utilizam Sharding

Nesta semana, vamos explorar como os frameworks que utilizam *sharding* e estão revolucionando a maneira como lidamos com grandes volumes de dados. A exemplo temos o estudo de caso sobre as férias em família nos parques temáticos da Disney. O uso intensivo de dados, como demonstrado no estudo de caso, demanda soluções eficazes para lidar com o volume massivo de informações geradas. É aqui que entram os frameworks que empregam *sharding*, como veremos na videoaula sobre o MongoDB. Vamos compreender os conceitos fundamentais por trás do MongoDB e como ele utiliza a técnica de sharding para

distribuir e gerenciar os dados de forma eficiente em ambientes distribuídos. Preparem-se para uma semana de aprendizado fascinante, onde vamos explorar como o Big Data está transformando diversas áreas, incluindo nossas experiências de férias em família, e como os frameworks que utilizam sharding desempenham um papel fundamental nesse cenário.



Fonte: ThisIsEngineering/Banco de imagens Pexels.

Bem-vindo à semana de estudo sobre frameworks que utilizam sharding! Sharding é uma técnica de distribuição de dados que divide conjuntos de dados em pequenas partes chamadas shards e as distribui em diferentes servidores. Essa abordagem é crucial para lidar com grandes volumes de dados e garantir escalabilidade e desempenho em sistemas distribuídos. Nesta aula, exploraremos alguns dos principais frameworks que implementam essa técnica e discutiremos suas características, benefícios e casos de uso.

Desvendando Crimes Ambientais com Sharding: Uma História Real



REFLEXÃO

O Agente Bruno, defensor ferrenho do meio ambiente, investiga um caso de desmatamento ilegal na Amazônia. Imagens de satélite revelam áreas devastadas, mas identificar os responsáveis é como procurar uma agulha no palheiro. A quantidade de dados é colossal: imagens de satélite, registros de propriedades, atividades de empresas madeireiras e até mesmo dados climáticos. A análise tradicional é lenta e trabalhosa, atrasando a justiça ambiental. O Agente Bruno decide usar o Sharding para acelerar a investigação. Essa tecnologia inovadora divide os dados em categorias, como imagens, registros e dados climáticos, armazenando-os em servidores distintos. No contexto de uma solução de Big Data por que usar a técnica de sharding pode tornar o trabalho do agente mais eficiente?

| Conceitos Fundamentais de Sharding

Definição de Sharding

Sharding é uma técnica de design de banco de dados e distribuição de dados em sistemas distribuídos. Consiste na divisão horizontal ou vertical de dados em múltiplos servidores de banco de dados, conhecidos como shards. O objetivo é distribuir a carga de trabalho e melhorar o desempenho, escalabilidade e disponibilidade do sistema. Imagine um enorme banco de dados como um quebra-cabeça gigante. O Sharding divide esse quebra-cabeça em peças menores, distribuindo-as em diferentes servidores, conforme apresentado na Figura 1. Cada servidor armazena um conjunto específico de dados, tornando o acesso e o processamento mais rápidos e eficientes. Em sistemas distribuídos, onde a demanda por armazenamento e processamento de dados pode ser imensa, o sharding desempenha um papel crucial. Ao distribuir os dados entre vários servidores, é possível aumentar a capacidade de armazenamento e processamento, reduzir gargalos de desempenho e melhorar a tolerância a falhas. Isso é especialmente importante em ambientes de grande escala, como redes sociais, serviços de e-commerce e sistemas de Big Data.

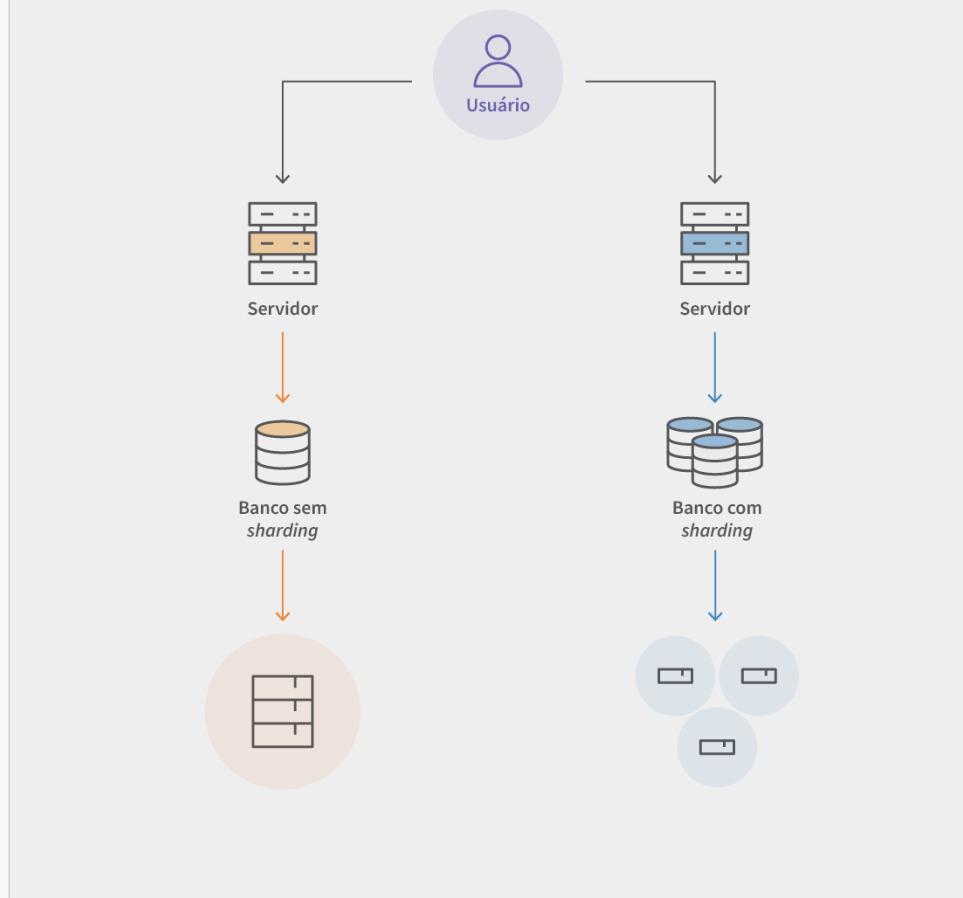


Figura 1: Comparação entre Banco de Dados com e sem aplicação de técnica *sharding*. Fonte: Adaptado de PEREIRA (2020).

Tipos de Sharding

De acordo com PEREIRA (2020), os dados precisam estar em segurança e devem ter uma estrutura planejada para a expansão. Já que um dos grandes desafios em aplicações de Big Data é a escalabilidade e para garantir a mesma os dados podem ser armazenados em dimensões. No sharding horizontal, os dados são divididos em registros ou linhas, distribuindo esses fragmentos de dados entre diferentes servidores. Cada shard contém uma parte dos dados completos do

conjunto de dados global. Esse método é eficaz quando há uma grande quantidade de dados que podem ser facilmente divididos em partes iguais ou aproximadamente iguais.

Já no sharding vertical, os dados são divididos com base nos atributos ou colunas em vez de linhas completas. Isso significa que diferentes servidores de banco de dados podem armazenar diferentes conjuntos de atributos para a mesma linha de dados. O sharding vertical é útil quando há diferentes tipos de dados ou quando certos atributos são acessados com mais frequência do que outros. Na Figura 2 é mostrado os tipos de sharding horizontal e vertical.

Tabela original

ID Cliente	Primeiro nome	Sobrenome	Cidade
1	Alice	Anderson	Austin
2	Bob	Best	Boston
3	Carrie	Conway	Chicago
4	David	Doe	Denver

Shards vertical

VS1

ID Cliente	Primeiro nome	Sobrenome
1	Alice	Anderson
2	Bob	Best
3	Carrie	Conway
4	David	Doe

VS2

ID Cliente	Cidade
1	Austin
2	Boston
3	Chicago
4	Denver

Shards horizontal

HS1

ID Cliente	Primeiro nome	Sobrenome	Cidade
1	Alice	Anderson	Austin
2	Bob	Best	Boston

n32

ID Cliente	Primeiro nome	Sobrenome	Cidade
3	Carrie	Conway	Chicago
4	David	Doe	Denver

Figura 2: Tipos de *sharding*. Fonte: Adaptado de SHARMA (2023).

Arquitetura do sharding

Para SHARMA (2023), após tomar a decisão de particionar seu banco de dados, o próximo passo crucial é definir a estratégia de implementação. Ao executar consultas ou distribuir dados recebidos para as tabelas ou bancos de dados particionados, é fundamental garantir que eles sejam direcionados ao shard correto. Falhas nesse direcionamento podem resultar em perda de dados ou lentidão nas consultas. Nos itens a seguir temos os tipos de estratégias para implementar a arquitetura de sharding.

+ Chave Hash

O sharding baseado em chave pode ser visto como uma forma de sharding baseado em hash, onde uma chave é usada para determinar o valor ou ID do shard ao qual ela pertence. Isso é feito por meio do uso de uma função de hash na chave. Na Figura 3 temos um exemplo da aplicação utilizando sharding com chave hash.

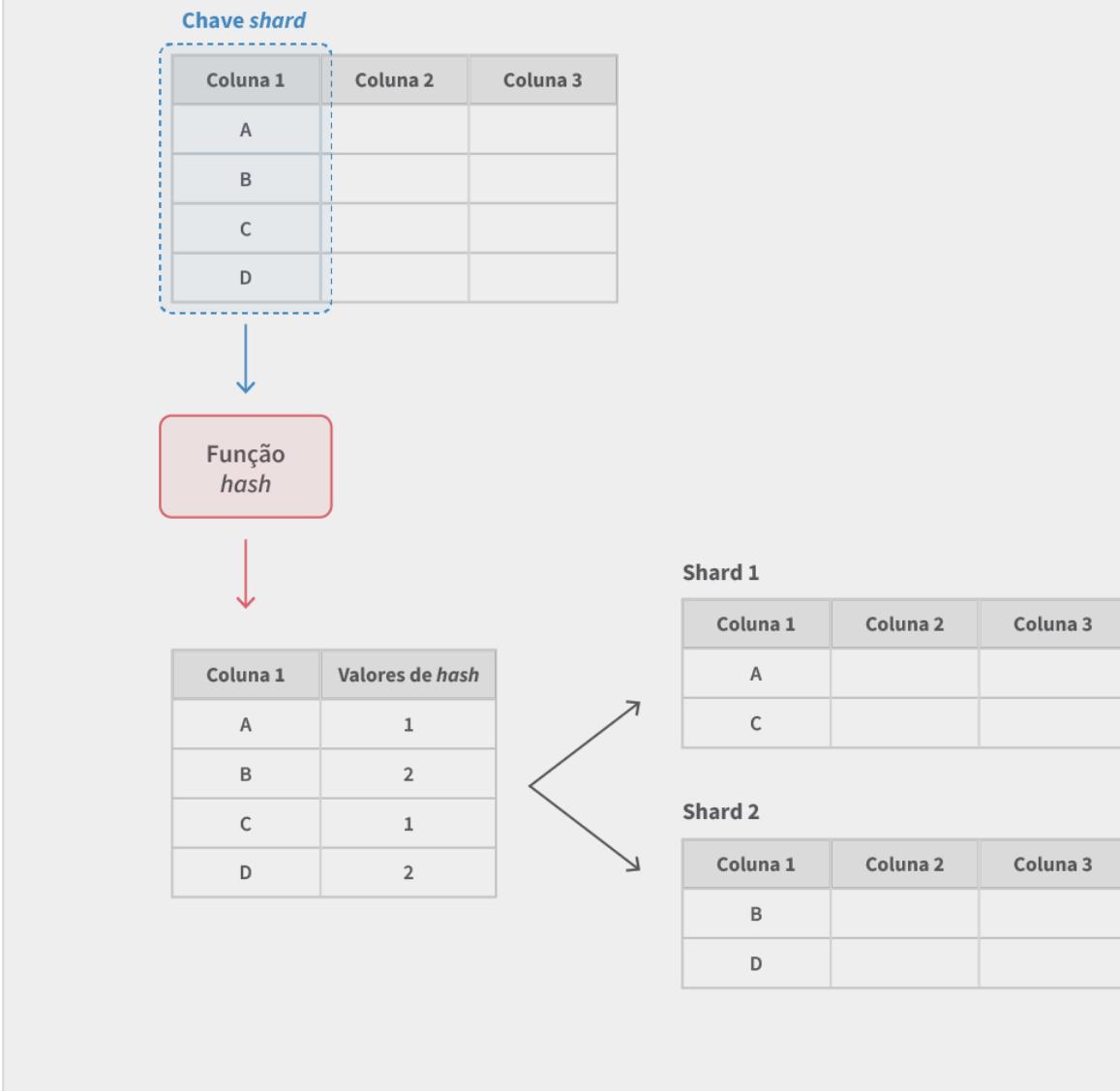


Figura 3: *sharding Hash*. Fonte: Adaptado de SHARMA (2023).

+ Range

O sharding por intervalo, também conhecido como *sharding* baseado em range, é uma técnica de particionamento de dados que os organiza em segmentos específicos, conforme Figura 4.

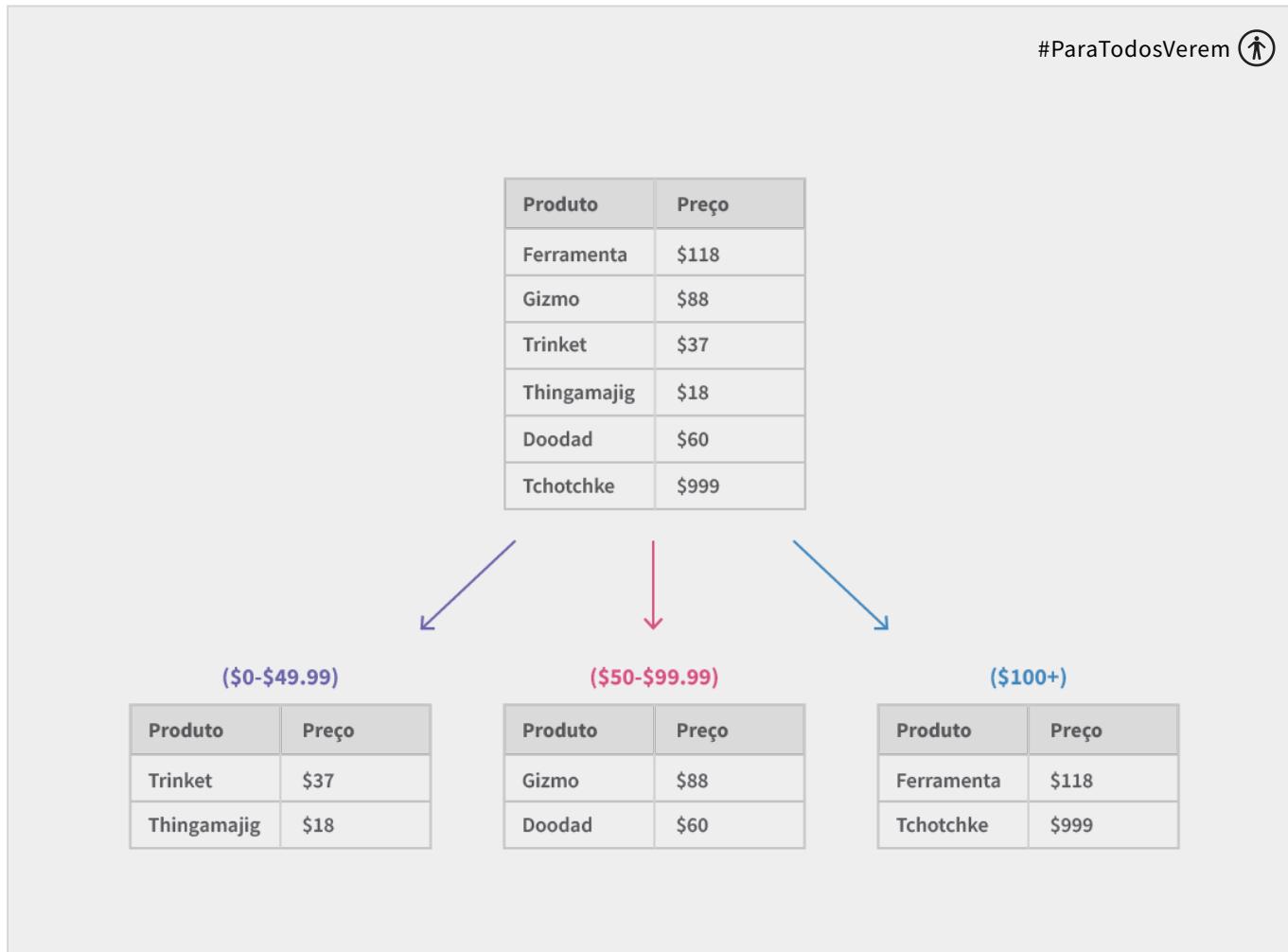


Figura 4: *sharding range*. Fonte: Adaptado de SHARMA(2023).

+ Diretório

O *sharding* baseado em diretório se destaca por sua flexibilidade e capacidade de se adaptar às suas necessidades específicas de particionamento de dados. Ao contrário de outros métodos que se restringem ao uso de chaves ou intervalos fixos, o *sharding* baseado em diretório oferece a liberdade de utilizar qualquer sistema ou algoritmo para distribuir seus dados entre os *shards*, de acordo com a Figura 5.

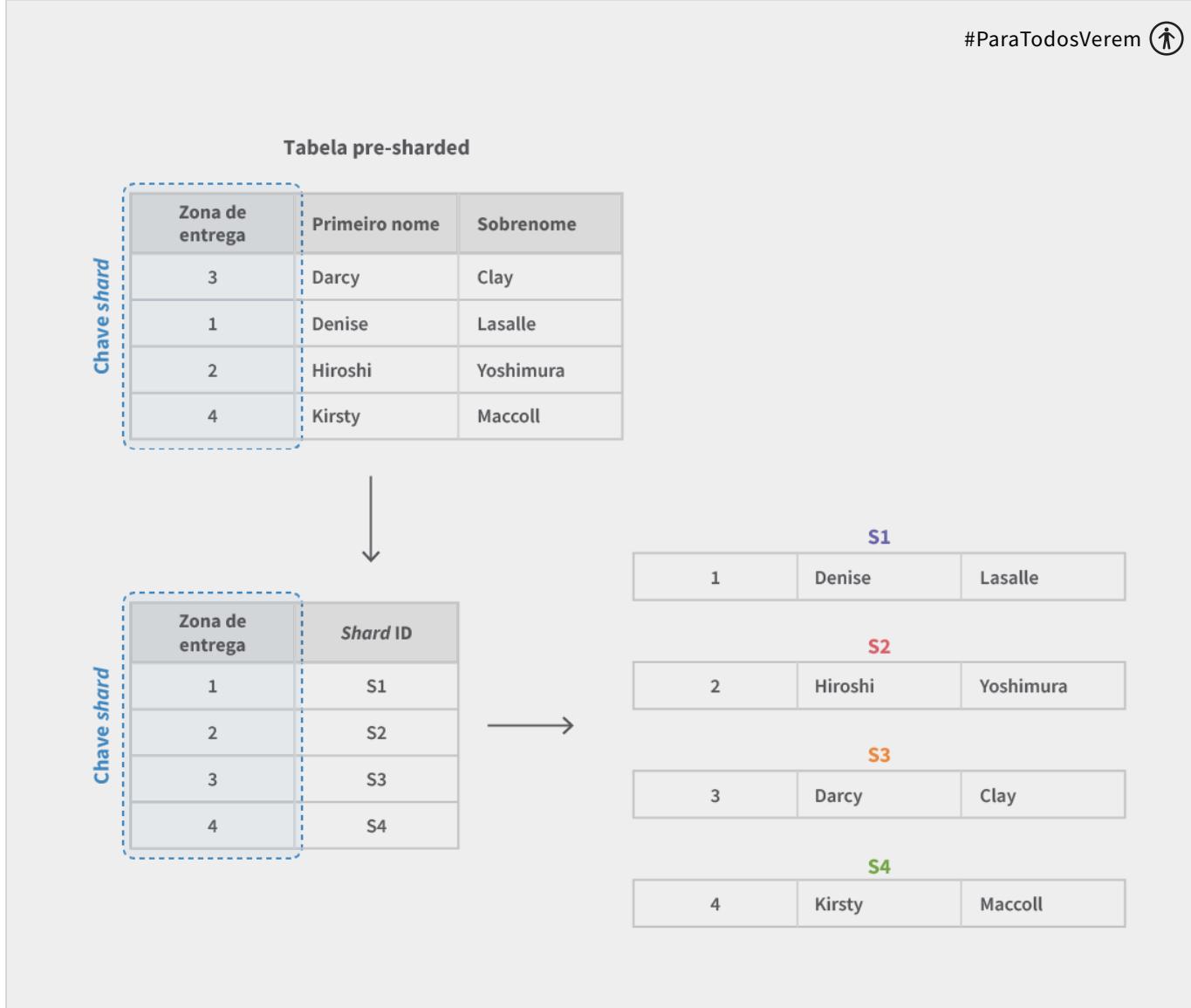


Figura 5: sharding por diretório. Fonte: Adaptado de SHARMA (2023).

SWE (2023) traz uma ideia sobre benefícios significativos em termos de escalabilidade e desempenho em sistemas distribuídos ao implementar sharding, conforme apresentado no Quadro 1.

Aspecto	Descrição
Distribuição de Dados	Em um banco de dados particionado, os dados são divididos em subconjuntos lógicos chamados shards, usando um critério de sharding predefinido.
Shards Independentes	Cada shard opera como um banco de dados independente, com seu próprio armazenamento, recursos de processamento e possivelmente seu próprio esquema.
Escalabilidade	O sharding permite alcançar escalabilidade horizontal, possibilitando ao sistema de banco de dados lidar com volumes maiores de dados e cargas de trabalho mais pesadas.
Balanceamento de Carga	O sharding possibilita o balanceamento de carga entre vários shards, garantindo que nenhum shard individual se torne um gargalo de desempenho.
Desempenho Aprimorado	Ao distribuir dados e carga de consulta, o sharding pode melhorar significativamente o desempenho do banco de dados.
Tolerância a Falhas	O sharding pode aprimorar a tolerância a falhas e a disponibilidade.
Consistência de Dados	Manten a consistência dos dados entre shards pode ser um desafio.
Complexidade	O sharding introduz complexidade em termos de distribuição de dados, roteamento de consultas e coordenação entre várias instâncias de banco de dados.
Distribuição Global	O sharding é comumente usado em sistemas globalmente distribuídos para armazenar dados mais perto dos usuários em diferentes regiões geográficas, reduzindo a latência e melhorando o desempenho para aplicativos globais.

O sharding se destaca como uma ferramenta poderosa para o futuro dos bancos de dados. Ao dominar essa técnica, as organizações podem construir sistemas robustos e escaláveis, prontos para enfrentar os desafios da era da informação.

| Frameworks que Utilizam Sharding

Para SOUZA (2017), ao planejar a escalabilidade por meio do sharding, a disponibilidade via replicação e a elasticidade, juntamente com o modelo de dados, é crucial examinar como a solução de SGBD NoSQL projeta sua topologia de arquitetura distribuída. Isso permite ao administrador de banco de dados selecionar a opção que ofereça os melhores benefícios de desempenho, considerando o tipo de dados a serem manipulados. Dessa forma, os Frameworks que utilizam sharding são essenciais para lidar com grandes volumes de dados e garantir a escalabilidade horizontal em sistemas distribuídos.

Apache Cassandra:

O Cassandra representa um dos primeiros bancos de dados NoSQL, caracterizado por um design híbrido entre armazenamento tabular e de chave-valor. Ele foi especificamente concebido para armazenar dados destinados a aplicações que requerem alto desempenho tanto em operações de leitura quanto de escrita. Segundo SOUZA (2017), esse SGBD possui uma topologia nativa configurada como um cluster, onde os nós deste cluster são organizados em uma topologia distribuída em forma de anel, como exemplificado na *Figura 6*. Além disso, o Cassandra pode ser configurado no modo sharding, conforme mostrado na Figura 6B, onde os dados são fragmentados e distribuídos entre as réplicas associadas a cada fragmento, mantendo as regras de replicação intactas. Cada nó troca informações regularmente sobre o seu estado e o estado dos outros nós no cluster, utilizando o protocolo denominado "gossip". Devido a esse protocolo, o Cassandra implementa a propriedade da elasticidade.

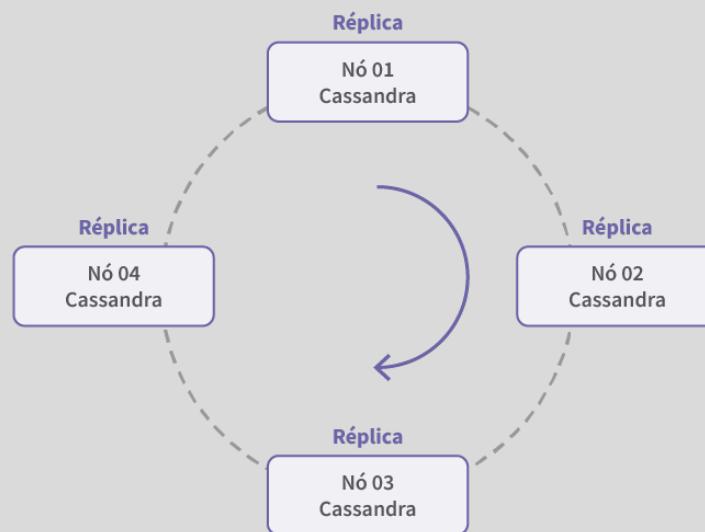
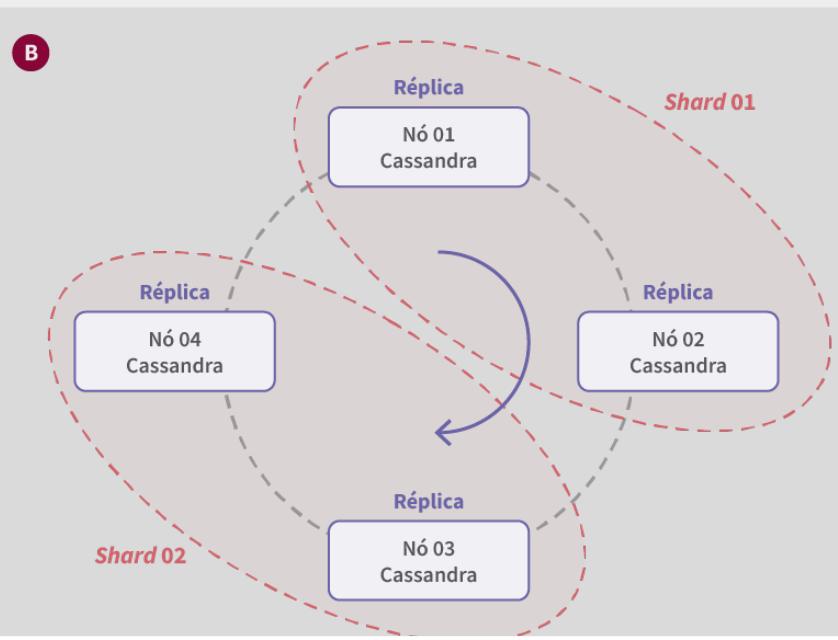
A**B**



Figura 6: Arquitetura Cassandra. Fonte: Adaptado de SOUZA (2017).

MongoDB:

Segundo PEREIRA (2020), o MongoDB, conforme apresentado na *Figura 7*, é descrito como um banco de dados distribuído, orientado a documentos e de uso geral, projetado para atender às necessidades dos desenvolvedores de aplicativos modernos na era da computação em nuvem. Operacionalmente, o MongoDB automatiza o equilíbrio de dados entre as partições (shards) e facilita a adição e remoção de máquinas para ajustar a capacidade de armazenamento. Para uma implementação completa do MongoDB, são necessários três componentes principais: shards, roteadores de consulta e servidores de configuração.

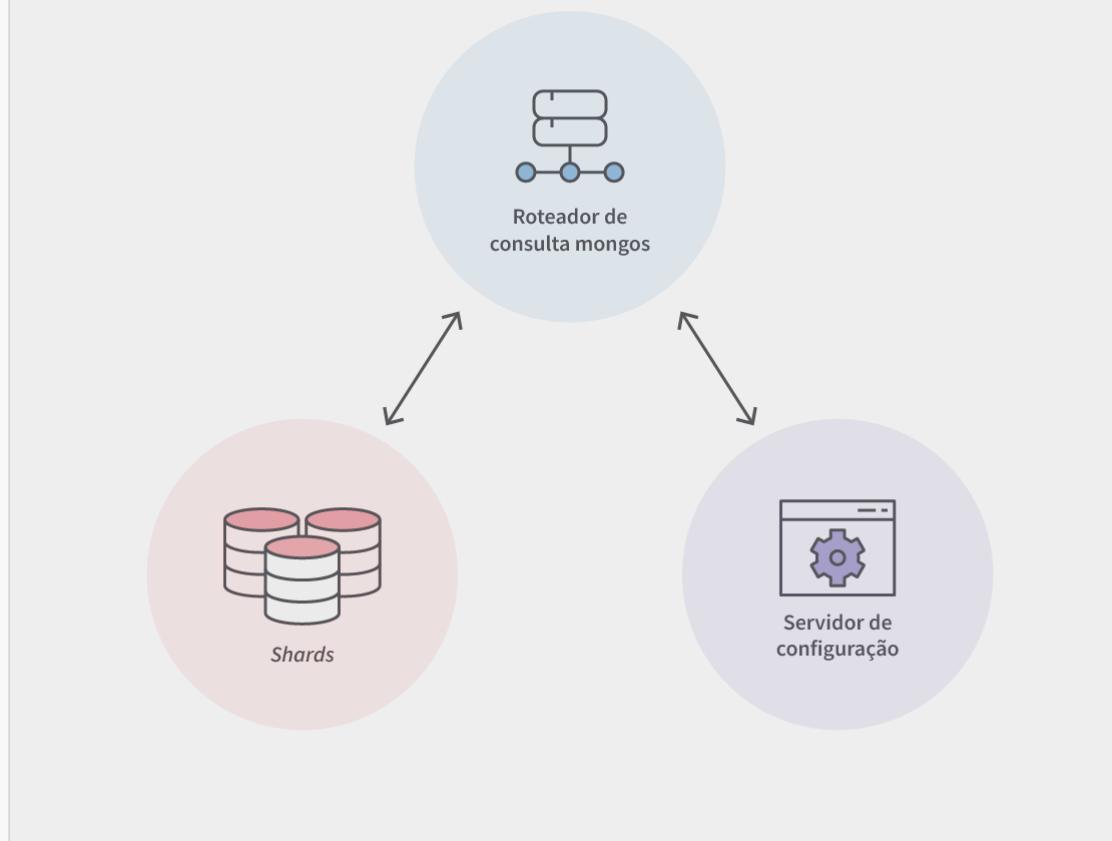


Figura 7: Arquitetura MongoDB. Fonte: Adaptado de PEREIRA (2020).

O MongoDB oferece suporte a duas estratégias de fragmentação, chave/hash e intervalo, ambas exigindo uma seleção cuidadosa antes da fragmentação. Isso se deve ao fato de que a escolha do campo para gerar a chave de fragmentação é definitiva. A seleção desta chave pode ter um impacto significativo no desempenho, eficiência e escalabilidade de um cluster fragmentado, independentemente da qualidade do hardware e da infraestrutura. Além disso, a escolha da chave de fragmentação e seu índice de suporte podem influenciar a estratégia que pode ser utilizada.

| Comparação entre Frameworks

No Quadro 2 apresentamos um comparativo entre as principais diferenças entre a arquitetura do MongoDB e do Cassandra

Característica	Cassandra	MongoDB
Arquitetura e Escalabilidade	Replicação em anel sem mestre, alta escalabilidade	Arquitetura mestre-escravo, dimensionamento horizontal e vertical
Linguagem de Consulta	CQL (similar ao SQL)	Fragments JSON e MQL
Modelo de Dados	Família de colunas (tabular)	Documentos (esquema flexível)
Linguagens de Programação	C#, C++, JavaScript, PHP, Python, etc.	C, C#, C++, JavaScript, PHP, Python, etc. (maior variedade)
Esquema	Esquema predefinido obrigatório	Esquema flexível (opcional)
Segurança	Controle de acesso baseado em função, segurança de transporte, rastreamento de auditoria	TLS/SSL, criptografia de nível de campo no lado do cliente (requer licença empresarial)
Desempenho de Leitura e Gravação	Gravações mais rápidas (várias gravações ao mesmo tempo)	Gravações mais lentas (único nó primário gravável)
Desempenho de Leitura	Índices secundários podem não ser totalmente utilizados	Índices secundários podem ser usados para leituras eficientes
Agregação	Sem estrutura interna, requer ferramentas externas	Estrutura de agregação integrada
Sharding	Sharding rígido (difícil de mudar)	Sharding flexível (pode ser mudado em tempo real)

Fonte: Adaptado de Amazon (2023).

Após analisarmos o comparativo entre as arquiteturas do MongoDB e do Cassandra, podemos concluir que cada uma possui suas características distintas, adequadas para diferentes necessidades e cenários de aplicação. Enquanto o Cassandra oferece uma abordagem de replicação em anel sem mestre, proporcionando alta escalabilidade e desempenho para ambientes distribuídos, o MongoDB destaca-se por sua flexibilidade e esquema dinâmico, ideal para casos onde a estrutura dos dados é menos definida.

Conclusão

Ao final desta aula, é importante recapitular o problema inicial enfrentado pelo Agente Bruno, que estava lidando com um desafio complexo relacionado ao desmatamento ilegal na Amazônia. As vastas quantidades de dados disponíveis tornavam a análise tradicional lenta e ineficiente, dificultando a identificação dos responsáveis pelos crimes ambientais. No entanto, ao adotar a técnica de sharding, o Agente Bruno e sua equipe puderam dividir esses dados em categorias específicas e distribuí-los em servidores distintos, permitindo uma análise mais rápida e eficiente. Isso possibilitou que equipes especializadas trabalhassem em paralelo, examinando imagens de satélite, registros de propriedades e dados climáticos simultaneamente, facilitando a identificação de padrões e correlações entre os diferentes conjuntos de dados.

O uso do sharding não apenas acelerou o processo de investigação, mas também tornou-o mais dinâmico e adaptável às mudanças e descobertas ao longo do caminho. Com essa abordagem, o Agente Bruno conseguiu identificar os responsáveis pelo desmatamento ilegal, garantindo a punição dos culpados e protegendo a floresta Amazônica. Portanto, podemos concluir que o sharding se mostrou uma ferramenta poderosa e eficaz no contexto de soluções de Big Data, permitindo lidar com grandes volumes de dados de forma mais eficiente e promovendo ações mais ágeis e assertivas na preservação do meio ambiente e na busca por justiça.

| Referências Bibliográficas

AMAZON WEB SERVICES (AWS). **The Difference Between Cassandra and MongoDB**. AWS, Inc., 2023. Disponível em: <https://aws.amazon.com/pt/compare/the-difference-between-cassandra-and-mongodb/>. Acesso em: 08 mar. 2024.

MONGODB, Inc. Sharding. MongoDB, Inc., 2023. Disponível em: <https://www.mongodb.com/docs/manual/sharding/>. Acesso em: 08 mar. 2024.

PEREIRA, M. A.; NEUMANN, F. B.; MILANI, A. M. P.; et al. **Framework de Big Data**. Porto Alegre: Grupo A, 2020. E-book. ISBN 9786556900803. Disponível em: <https://integrada.minhabiblioteca.com.br/#/books/9786556900803/>. Acesso em: 29 fev. 2024.

SHARMA, A. **Sharding a database server: scaling technique**. Medium, 2023. Disponível em: <https://medium.com/@akshatsharma0610/sharding-a-database-server-scaling-technique-88c3b7067930>. Acesso em: 08 mar. 2024.

SWE, S. Database Sharding Implementation. Medium, 14 fev. 2023. Disponível em: <https://medium.com/@sujoy.swe/database-sharding-implementation-e11cbba0e745>. Acesso em: 14 mar. 2024.

SOUZA JUNIOR, J. B. **NoSQL ClusterAdmin**: uma ferramenta para configuração, gerenciamento e monitoramento de SGBD NoSQL fragmentados e replicados / João Bosco de Souza Junior. – 2017.



© PUCPR - Todos os direitos reservados.