



PUC Minas  
Virtual

# Arquiteturas e Serviços de Data Lakes e Data Warehousing



PUC Minas  
Virtual

---

# Unidade 4 – Arquitetura de Dados

---

# Arquiteturas com Alta Disponibilidade (HA)

# Alta Disponibilidade

- Alta disponibilidade: Refere-se a um conjunto de tecnologias que minimiza as interrupções de TI proporcionando a continuidade dos negócios de serviços de TI por meio de componentes redundantes e tolerantes a falhas ou protegidos contra failover no mesmo data center.

# Sistemas Distribuídos

Os sistemas distribuídos são recursos computacionais compartilhados em uma rede permitindo um aumento no desempenho, tolerância a falhas e escalabilidade do sistema.

A distinção fundamental é que em um sistema distribuído, uma coleção de computadores independentes mostra-se aos usuários como sendo um sistema único.

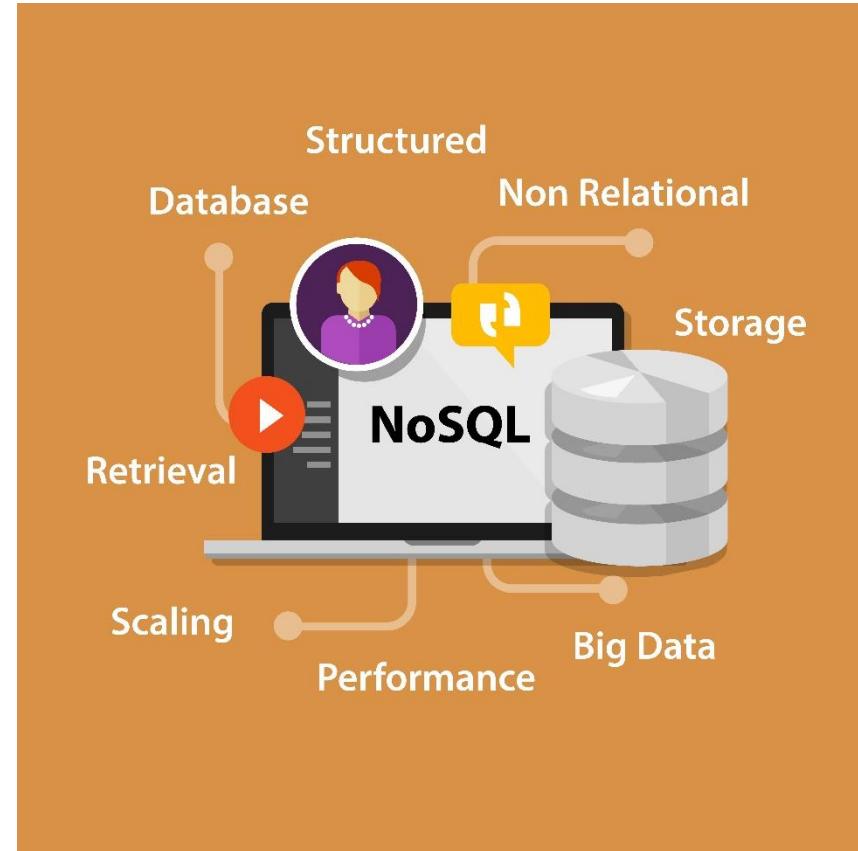
# Sistemas de Arquivos Distribuídos

Sistema de arquivo distribuído (SAD) permite os programas armazenar e acessar arquivos remotos exatamente como se fosse um acesso local, permitindo o acesso a partir de qualquer computador em uma rede.

A **replicação** é o segredo da eficácia dos sistemas distribuídos, pois ela é a chave para prover melhor desempenho, alta disponibilidade e tolerância a falhas.

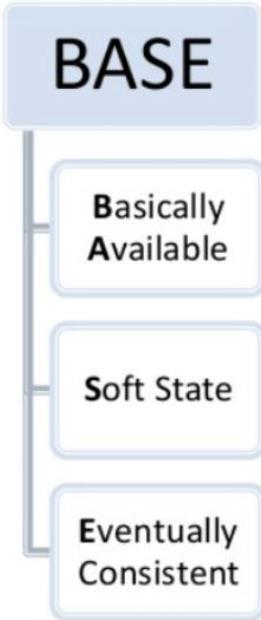
# Banco de Dados NoSQL

- Escalabilidade horizontal.
- Ausência de esquema ou esquema flexível.
- Suporte à replicação.
- API simples.
- Nem sempre prima pela consistência.



# Propriedades BASE

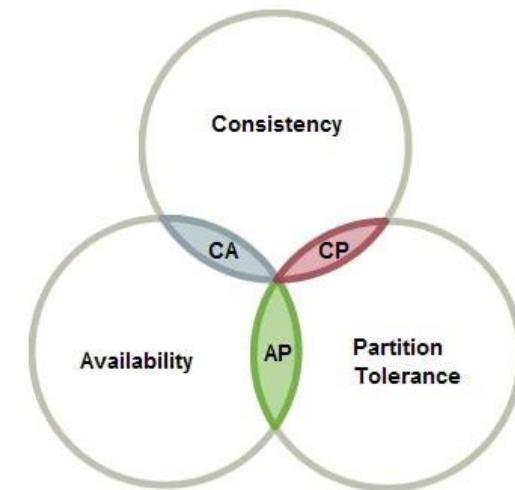
- **Basically Available** – Basicamente Disponível.
- **Soft-State** – Estado Leve.
- **Eventually Consistent** – Eventualmente Consistente.
- Uma aplicação funciona basicamente todo o tempo (Basicamente Disponível), não tem que ser consistente todo o tempo (Estado Leve) e o sistema torna-se consistente no momento devido (Eventualmente Consistente).



- **C**onsistency – Consistência.
- **A**vailability – Disponibilidade.
- **P**artition Tolerance – Tolerância ao Particionamento.

# Teorema CAP

- De acordo com o teorema **CAP**, *um sistema distribuído de bancos de dados somente pode operar com dois desses comportamentos ao mesmo tempo, mas jamais com os três simultaneamente.*



# Arquiteturas com Alta Disponibilidade (HA) – Exemplo MongoDB

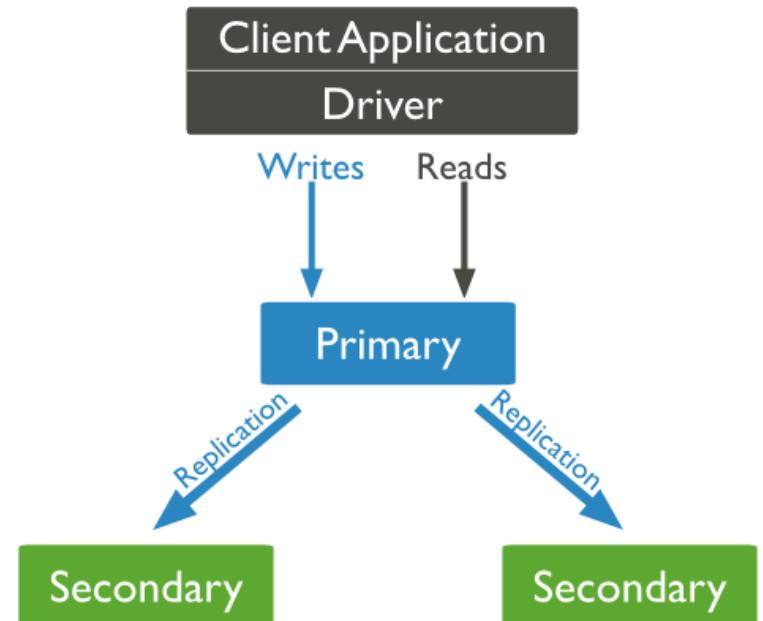
# Sistemas de Arquivos Distribuídos

Na busca de sistemas mais confiáveis, alguns meios foram desenvolvidos para oferecer mais confiança aos sistemas, entre eles está a **tolerância a falhas**. Dessa forma, é essencial para o sistema de arquivos distribuídos que **continuem a funcionar diante de falhas que aconteçam em servidores**.

A **replicação** é o método utilizado para **aumentar a disponibilidade de um serviço de arquivos**, onde os arquivos são armazenados em dois ou mais servidores e caso um deles não esteja disponível, outro servidor poderá fornecer os serviços solicitados.

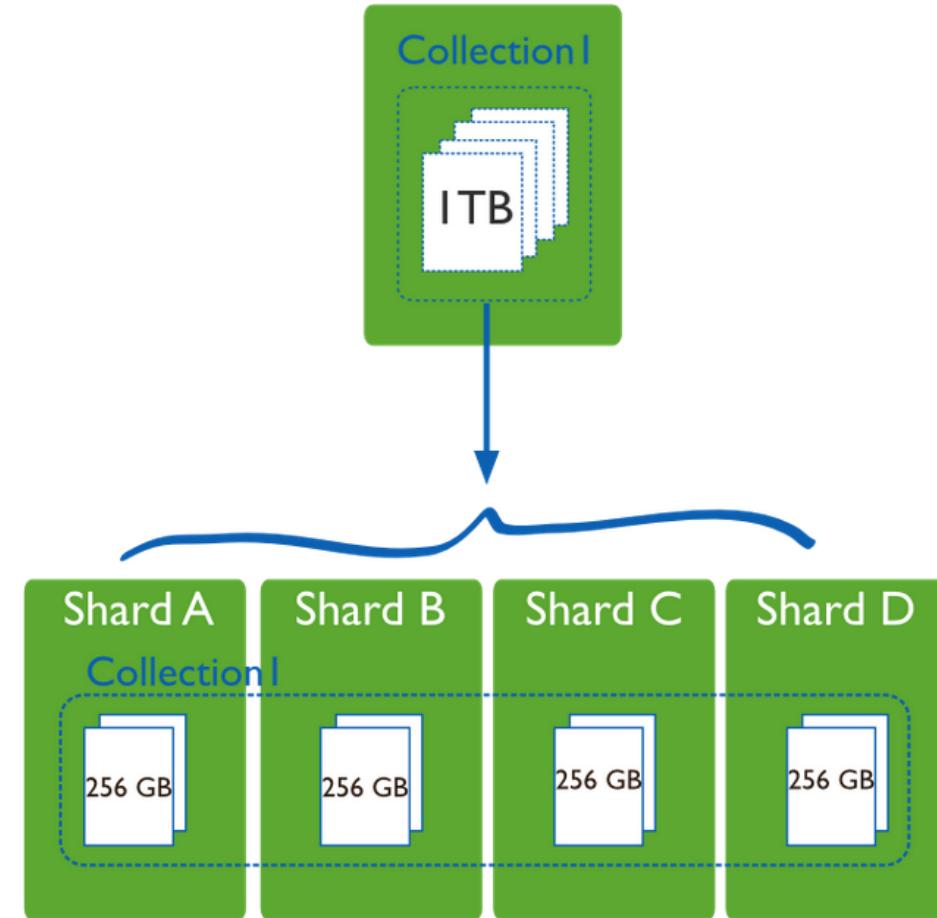
# Arquitetura Mongodb

- Um conjunto de réplicas no MongoDB é um grupo de instâncias mongod que mantém o mesmo conjunto de dados. Ele contém vários nós de suporte de dados e, opcionalmente, um nó de árbitro.
- O nó primário recebe todas as operações de leitura e gravação e registra tudo nos seus conjuntos de dados e logs.
- Os secundários replicam o log do primário e aplicam as operações a seus conjuntos de dados de forma assíncrona.
- Tendo os conjuntos de dados secundários que refletem o conjunto de dados do primário, o conjunto de réplicas pode continuar a funcionar, apesar da falha de um ou mais membros.
- Se o nó primário não estiver disponível, um secundário elegível fará uma eleição para se eleger como o novo nó primário.

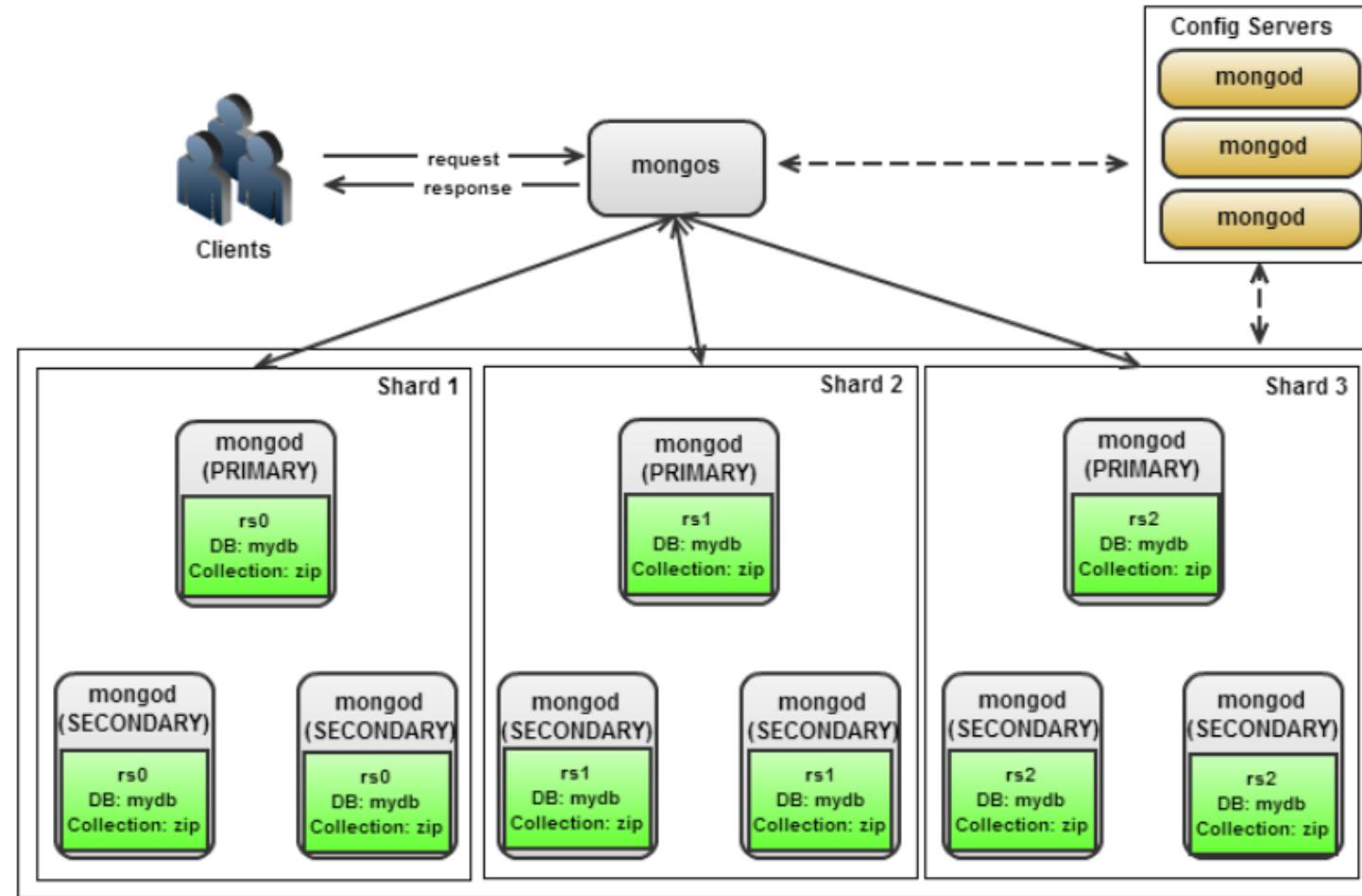


# Sharding

- Um cluster (agrupamento) no mongodb é um agrupamento fragmentado:
  - Escala leituras e gravações ao longo de vários nós.
  - Cada nó não lida com todos os dados, portanto, você pode separar os dados ao longo de todos os nós do fragmento.

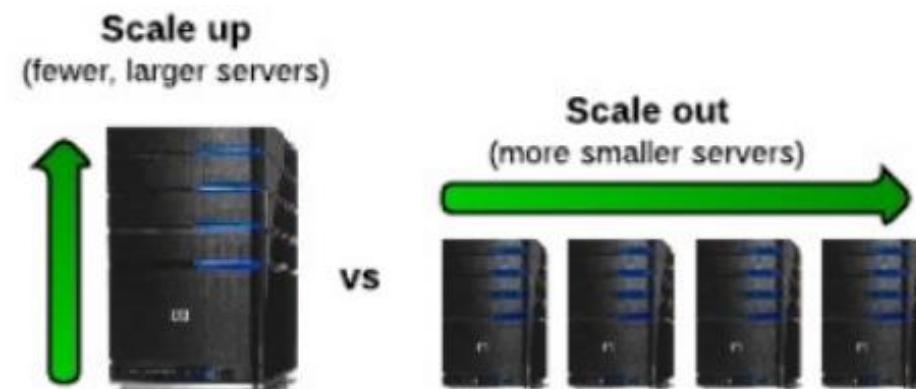


# Arquitetura usando Sharding e ReplicaSet.



# Escalabilidade

- À medida em que o volume de dados cresce, aumenta-se a necessidade de escalabilidade e melhoria do desempenho.
- Podemos escalar **verticalmente** (adicionar CPU, memória e disco) ou podemos escalar **horizontalmente** (adicionar mais nós).



# Data Ponds

# Data Pond

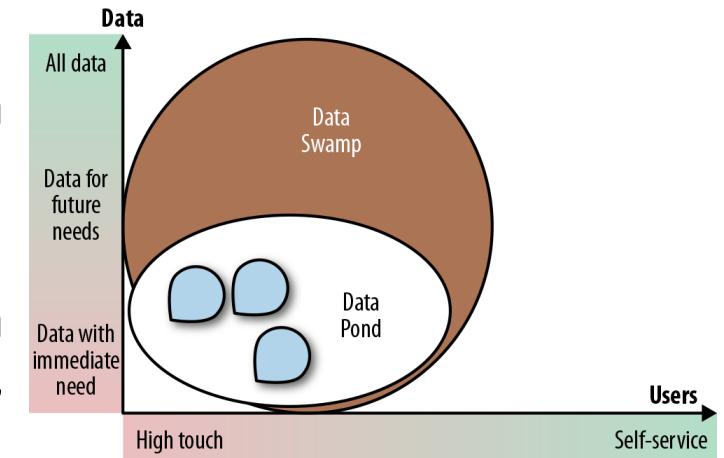
- Como Data Warehouses, um Data Pond é um conceito, não uma tecnologia, Você pode usar várias tecnologias para construir um Data Pond.
- Um Data Pond é essencialmente uma estratégia de armazenamento de dados.
- Um Data Pond é definido como um enorme – e relativamente barato – repositório de armazenamento, como o Hadoop, que pode armazenar todos os tipos de dados necessários para análise de negócios ou recuperação de dados.

# Data Pond

- Os Data Ponds são projetados para receber dados e o procedimento envolve a coleta, importação e processamento de dados para armazenamento ou uso posterior. Nos casos em que o modelo de preços para armazenamento em um Data Warehouse não é adequado para a coleta completa de dados, um Data Lake pode ser usado.
- Embora os dados possam ser armazenados em um Data Lake sem serem estruturados, faz sentido, desde o início, organizar o espaço para armazenar e armazenar dados por categoria. Assim, cada usuário poderá encontrar e utilizar os dados necessários mais rapidamente, e o risco do “lago” se transformar em um “pântano” é significativamente reduzido.

# Data Pond

- Os Data Ponds são recursos para toda a organização, não apenas para o departamento de TI. Portanto, todas as partes interessadas devem estar envolvidas no planejamento de projetos de Data Lake.
- O Data Lake é fundamental para a arquitetura de Big Data da empresa e, portanto, não pode ser implementado isoladamente.
- O Data Puddle (poça de dados) geralmente são criadas para uma pequena equipe focada ou caso de uso especializado. Essas “poças” são coleções de dados de tamanho modesto pertencentes a uma única equipe, frequentemente construídas na nuvem por unidades de negócios.



Fonte:  
<https://www.oreilly.com/library/view/the-enterprise-big/9781491931547/ch01.html>

# Apache Hadoop

- Hadoop é uma plataforma de software em Java de computação distribuída voltada para clusters e processamento de grandes volumes de dados, com atenção a tolerância a falhas.
- Foi inspirada no MapReduce (um modelo de programação paralela para processamento largamente distribuído de grandes volumes de dados proposto primeiramente pela empresa Google) e no GoogleFS (Google File System é um sistema de arquivos distribuído proprietário desenvolvido pelo Google para fornecer acesso eficiente e confiável aos dados usando grandes clusters de hardware comum).
- É disponibilizado pela Amazon e IBM em suas plataformas.



# Módulos do Framework do Apache Hadoop

- **Hadoop Common** - Contém as bibliotecas e arquivos comuns e necessários para todos os módulos Hadoop.
- **Hadoop Distributed File System (HDFS)** - Sistema de arquivos distribuído que armazena dados em máquinas dentro do cluster, sob demanda, permitindo uma largura de banda muito grande em todo o cluster.
- **Hadoop Yarn** - Trata-se de uma plataforma de gerenciamento de recursos responsável pelo gerenciamento dos recursos computacionais em cluster, assim como pelo agendamento dos recursos.
- **Hadoop MapReduce** - Modelo de programação para processamento em larga escala.

# Blocos Funcionais em uma Arquitetura de Dados

# Tipos de Sistemas Gerenciadores de Dados

- **Data Warehouses:** um Data Warehouse agrupa dados de diferentes fontes de dados relacionais em uma empresa em um repositório único, central e consistente.
- **Data Marts:** um Data Mart é uma versão focada de um Data Warehouse que contém um subconjunto menor de dados importantes e necessários para uma única equipe ou um grupo seletivo de usuários dentro de uma organização, como o departamento de RH.
- **Data Lakes:** enquanto os Data Warehouses armazenam dados processados, um Data Lake armazena dados brutos, normalmente petabytes deles. Um Data Lake pode armazenar dados estruturados e não estruturados, o que o torna exclusivo de outros repositórios de dados.

# Tipos de Arquitetura de Dados

- **Data Fabrics:** é uma arquitetura que se concentra na automação da integração de dados, engenharia de dados e governança em uma cadeia de valor de dados entre provedores de dados e consumidores de dados. Uma malha de dados é baseada na noção de “metadados ativos” que usa gráfico de conhecimento, semântica, mineração de dados e tecnologia de aprendizado de máquina (ML) para descobrir padrões em vários tipos de metadados (por exemplo, logs do sistema, social, etc.). Em seguida, aplica esse insight para automatizar e orquestrar a cadeia de valor dos dados. Por exemplo, ele pode permitir que um consumidor de dados encontre um produto de dados e, em seguida, tenha esse produto de dados provisionado para ele automaticamente.

# Tipos de Arquitetura de Dados

- **Data Meshes:** uma malha de dados é uma arquitetura de dados descentralizada que organiza os dados por domínio de negócios. Usando uma malha de dados, a organização precisa parar de pensar nos dados como um subproduto de um processo e começar a pensar neles como um produto por si só. Os produtores de dados atuam como proprietários de produtos de dados. Como especialistas no assunto, os produtores de dados podem usar sua compreensão dos principais consumidores dos dados para projetar APIs para eles. Essas APIs também podem ser acessadas de outras partes da organização, fornecendo acesso mais amplo aos dados gerenciados.

# Benefícios de Arquitetura de Dados

- **Reduzindo a redundância:** pode haver campos de dados sobrepostos em diferentes fontes, resultando em risco de inconsistência, imprecisões de dados e oportunidades perdidas de integração de dados. Uma boa arquitetura de dados pode padronizar como os dados são armazenados e potencialmente reduzir a duplicação, permitindo análises holísticas e de melhor qualidade.
- **Melhorando a qualidade dos dados:** Arquiteturas de dados bem projetadas podem resolver alguns dos desafios de Data Lakes mal gerenciados, também conhecidos como “pântanos de dados”. Um pântano de dados (DATA SWAMP) carece de qualidade de dados apropriada e práticas de governança de dados para fornecer aprendizados perspicazes.

# Benefícios de Arquitetura de Dados

- **Habilitando a integração:** os dados geralmente ficam isolados, como resultado de limitações técnicas no armazenamento de dados e barreiras organizacionais dentro da empresa. As arquiteturas de dados de hoje devem ter como objetivo facilitar a integração de dados entre domínios, para que diferentes geografias e funções de negócios tenham acesso aos dados uns dos outros.
- **Gerenciamento do ciclo de vida dos dados:** uma arquitetura de dados moderna pode abordar como os dados são gerenciados ao longo do tempo.

# Características de uma Arquitetura Moderna de Dados

- **Cloud-native e cloud-enabled**, para que a arquitetura de dados possa se beneficiar do dimensionamento elástico e da alta disponibilidade da nuvem.
- **Pipelines de dados robustos, escaláveis e portáteis**, que combinam fluxos de trabalho inteligentes, análises cognitivas e integração em tempo real em uma única estrutura.
- **Integração de dados perfeita**, usando interfaces de API padrão para conectar-se a aplicativos legados.
- **Habilitação de dados em tempo real**, incluindo validação, classificação, gerenciamento e governança.
- **Desacoplado e extensível**, portanto, não há dependências entre os serviços e os padrões abertos permitem a interoperabilidade.
- **Otimizado** para equilibrar custo e simplicidade.

# Schema On-Write x Schema On-Read

# Schema-on-Write

- Essa construção está fortemente vinculada ao gerenciamento de Banco de Dados Relacional, incluindo o esquema e a criação de tabelas e a ingestão de dados.
- O problema aqui é que os dados não podem ser carregados nas tabelas sem os esquemas e tabelas criados e configurados.



Fonte: <https://medium.com/@deepa.account/schema-on-read-and-schema-on-write-ac72524926b>

# Schema-on-Write

- Uma estrutura de banco de dados funcional não é definível sem entender a estrutura dos dados que devem ser ingeridos no banco de dados. Uma das tarefas mais demoradas ao trabalhar com um banco de dados relacional é fazer o trabalho Extract Transform Load (ETL). Os dados brutos raramente são estruturados para que sejam carregados no banco de dados sem a necessidade de transformação para se adequar ao esquema do banco de dados.
- Portanto, você não deve apenas definir o esquema do banco de dados para adequá-lo aos dados, mas também estruturar os dados para adequá-los ao esquema.



# Schema on-Read

- O esquema do banco de dados é criado quando os dados são lidos.
- As estruturas de dados não são aplicadas ou iniciadas antes que os dados sejam ingeridos no banco de dados; eles são criados durante o processo ETL.
- Isso permite que dados não estruturados sejam armazenados no banco de dados.
- A principal razão para desenvolver o princípio do esquema na leitura é o crescimento explosivo dos volumes de dados não estruturados e a alta sobrecarga envolvida durante o processo do esquema na gravação.



Fonte: <https://medium.com/@deepa.account/schema-on-read-and-schema-on-write-ac72524926b>

# Sincronização de Fluxos Independentes e Dependentes

# Orquestração de Dados

- A orquestração de dados é o processo de obter dados isolados de vários locais de armazenamento de dados, combiná-los e organizá-los e disponibilizá-los para ferramentas de análise de dados. A orquestração de dados permite que as empresas automatizem e simplifiquem a tomada de decisões baseada em dados.
- O software que executa a orquestração de dados conecta seus sistemas de armazenamento para que suas ferramentas de análise de dados possam acessar facilmente o sistema de armazenamento necessário quando necessário.

# 3 Passos para Orquestração de Dados

## Organizar

- As ferramentas de orquestração de dados primeiro precisam entender e organizar seus dados existentes e novos dados recebidos.
- Onde quer que esses dados estejam, as ferramentas de orquestração de dados precisam ser capazes de acessá-los e entender que tipo de dados existem e de onde vieram.

# 3 Passos para Orquestração de Dados

## Transformar

- As ferramentas de orquestração de dados pegam dados em diferentes formatos e os transformam para que fiquem em um formato padrão. Isso torna a análise de dados mais rápida porque você não precisa perder tempo reconciliando manualmente os dados.

# 3 Passos para Orquestração de Dados

## Ativar

- A parte mais importante da orquestração é disponibilizar os dados para as ferramentas que precisam deles. Isso se chama ativação. A ativação acontece quando as ferramentas de orquestração enviam os dados para as ferramentas que sua empresa usa para operar no dia a dia. Dessa forma, os dados de que você precisa já estão lá quando você precisa - o carregamento de dados não é necessário.

# Dependência de Dados

Dependência de dados deve ser conectada às etapas de orquestração:

1. Aguardar o tempo: a expressão Cron significa apenas — “o primeiro horário” em que um fluxo pode ser iniciado. Essa configuração fixa é realmente opcional, porque um fluxo agora pode reagir puramente a um evento
2. Aguarde os dados de entrada:
  - As tabelas de partições devem ter as partições esperadas já preenchidas.
  - As tabelas de snapshot/dimensão devem ter incremento/delta.

# Dependência de Dados

- 3. Opcionalmente, aguarde o resultado das verificações de qualidade de dados para as tabelas de entrada selecionadas.
- 4. Aguardar recursos: Se todas as 3 condições acima forem satisfeitas, o orquestrador pode verificar qual cluster/fila/pod tem capacidade suficiente (ou é o menos ocupado) e atribuir o fluxo a esse recurso de computação específico. Se todos os recursos estiverem ocupados ou temporariamente off-line, o fluxo continuará aguardando.
- 5. Inicie o fluxo no modo de execução.

